COMPUTER SCIENCE S5

STUDENT'S BOOK

FOREWORD

Dear Student,

Rwanda Basic Education Board is honoured to present to you this Computer Science book for Senior five which serves as a guide to competence-based teaching and learning to ensure consistency and coherence in the learning of Computer Science subject. The Rwandan educational philosophy is to ensure that you achieve full potential at every level of education which will prepare you to be well integrated in society and exploit employment opportunities.

The government of Rwanda emphasizes the importance of aligning teaching and learning materials with the syllabus to facilitate your learning process. Many factors influence what you learn, how well you learn and the competences you acquire. Those factors include the instructional materials available among others. Special attention to the activities that facilitate the learning process in which you can develop your ideas and make new discoveries during concrete activities carried out individually or with peers.

In competence-based curriculum, learning is considered as a process of active building and developing knowledge and meanings by the learner where concepts are mainly introduced by an activity, a situation or a scenario that helps the learner to construct knowledge, develop skills and acquire positive attitudes and values. For effective use of this textbook, your role is to:

- Work on given activities which lead to the development of skills
- Share relevant information with other learners through presentations, discussions, group work and other active learning techniques such as role play, case studies, investigation and research in the library, from the internet or from your community;
- Participate and take responsibility for your own learning;
- Draw conclusions based on the findings from the learning activities.

I wish to sincerely extend my appreciation to the people who contributed towards the development of this book, particularly REB staff who organized the whole process from its inception. Special gratitude goes to the University of Rwanda which provided experts in design and layout services, illustrations and image anti-plagiarism, lecturers and teachers who diligently worked to successful completion of this book. Any comment or contribution would be welcome for the improvement of this textbook for the next edition.

Dr. MBARUSHIMANA Nelson

Director General, REB

ACKNOWLEDGEMENT

I wish to express my appreciation to all the people who played a major role in development of this textbook. It would not have been successful without active participation of different education stakeholders.

I owe gratitude to different Universities and schools in Rwanda that allowed their staff to work with REB in the in-house textbooks production project. I wish to extend my sincere gratitude to lecturers, teachers and all other individuals whose efforts in one way or the other contributed to the success of writing of this textbook.

Special acknowledgement goes to the University of Rwanda which provided experts in design and layout services, illustrations and image anti-plagiarism..

Finally, my word of gratitude goes to the Rwanda Education Board staff particularly those from the Curriculum, Teaching and Learning Resources department (CTLR) who were involved in the whole process of in-house textbook writing.

Murungi Joan,

Head of Curriculum, Teaching and Learning Resources Department

TABLE OF CONTENTS

UNIT 1
FUNDAMENTALS OF LAPTOP AND PORTABLE DEVICES

UNIT1: FUNDAMENTALS OF LAPTOP AND PORTABLE DEVICES

Key Unit Competency: To be able to identify, use and maintain safely laptops and other portable devices.

INTRODUCTORY ACTIVITY

Apply the following scenario:
You are in computer lab with different devices on table.

Figure 1.1. Mobile devices

Answer the following questions:

1. Name the devices you see on your table
2. Describe parts of each device.
3. Describe the common use and functions of these devices

1.1. PORTABLE DEVICES

Learning Activity 1.1

1) You are sited in your computer lab, observe well each device and then answer following questions:

a. Describe the portable devices available in lab;
b. Discuss the similarities and differences between available devices;
c. Enumerate other portable devices which are not in your lab.

2) In supermarket A, a laptop costs 350,000 Rwf while in supermarket B, laptop of same quality costs 300,000Rwf. Select where you will prefer to buy your laptop and explain why.

1.1.1. Portable devices definition

A *portable device* is a device that can easily be carried. It is a small form of a computing *device* that is designed to be held and used in the hands.

A portable device may also be called a handheld device or mobile device. Portable devices are primarily battery powered devices with base computing resources in the form of a processor, memory, storage and network access. Portable devices are becoming an increasingly important part of personal computing as the capabilities of devices like laptops, tablets and smartphones continue to improve.

11.2. Types of portable devices

a. Laptops

A laptop computer is a portable personal computer powered by a battery, or an Alternating Current (A.C) cord plugged into an electrical outlet, which is also used to charge the battery. Laptops have an attached keyboard and a touchpad, trackball, or isometric joystick used for navigation. Sometimes, a wireless mouse can also be used on a laptop. A wireless mouse is a computer mouse that needs no wires to send signals from the mouse to a computer. Wireless mouse technology predominantly uses radio frequencies (RF) to send signals from the mouse to the computer. Like other radio technologies, this requires a transmitter and a receiver. The mouse transmits radio signals to a receiver, which is itself connected to the computer hardware, normally via a wire. This kind of wireless mouse is very reliable, and capable of transmitting the mouse's movements to the receiver from across a room. A laptop also has a thin display screen that is attached and can be folded flat for transport. Laptop computer is also known as Notebook computer or simply Notebook.

The memory and storage capacity of laptop computer is almost equivalent to the PC or desktop computer. It also has the hard disk, DVD/CD optical disk drive, different connectors etc.

Figure 1.2. Laptop

b. Smartphones

A smart phone is a handheld personal computer with a mobile operating system and an integrated mobile broadband network connection for voice, SMS, and Internet data communication. Most if not all smartphones also support Wireless Fidelity (Wi-Fi). Smartphones are typically pocket-sized, as opposed to tablets, which are much larger. A smartphone is a cellular phone that performs most of the functions of a computer.

A smart phone supports all features of PDA plus mobile phone features

Smartphone characteristics/features

- Media players
- Internet access
- Global Positioning System(GPS)
- Wireless Fidelity (Wi-Fi)
- Camera
- Sensor

c. Mobile phones (cell phones)

A mobile phone or cell phone (is a long-range, electronic device used for mobile telecommunications (mobile telephony, text messaging or data transmission) over a cellular network. Today, this type of portable device is being dominated by smart phone as technology improves.

Figure 1.3: Mobile phones or cell phones

d. PDAs

A Personal Digital Assistant (PDA), also known as a palmtop computer, or Personal Data Assistant, is a mobile device that functions as a personal information manager. Newer PDAs commonly have color screens, audio capabilities and web browsers that enable them to be used as mobile phones (smartphones) or portable media players.

Many PDAs can access the Internet, intranets (communication within LAN) or extranets via Wi-Fi, or Wireless Wide Area Networks. Many PDAs employ touch screen technology.

e. Tablets

A tablet computer, or simply tablet, is a mobile computer with a large screen, circuitry and battery in a single unit with no built in keyboard. Some tablets include detachable keyboards have been sold since the mid-1990s. Such tablets include iPad, HP Touchpad, Android)

Tablets are very similar to smartphones. They have an Internet connection via a wireless connection including 3G/4G. Applications can be downloaded and installed. Sensors including speech recognition.

Figure 1.4: IPad

Application Activity 1.1.

6. Describe the difference between PDA and Smartphone
7. Compare a smart phone and Tablet
8. According to the comparison between laptop, Smart phone, Cell phone, PDA and Tablet, which one is most useful? Explain why.

1.2 LAPTOP COMPUTER FUNDAMENTALS

Learning Activity 1.2.

1) In the computer lab observe laptop computers and make groups and answer the following questions:

a) Discuss common uses of laptop computer in schools and society in general

b) Describe the external components of a laptop

c) Describe the internal components of a laptop

d) Investigate why laptop has Fn key on keyboard while desktop keyboard does not have it

e) With your laptops connected to Internet, search for steps to be followed while applying for National ID at www.irembo.gov.rw site

f) In the computer lab, identify the following features of a laptop: AC power port, ethernet port VGA port, vents, HDMI port, USB ports, camera.

g) Open a laptop and identify all its internal components

1.2.1. Common uses

The first laptops were used primarily by business people who needed to access and enter data when they were away from the office. The use of laptops was limited due to expense, weight, and limited capabilities compared to less expensive desktops.

Some common uses for laptops

Laptop is used in:

- Education field (example: Laptops are used by teachers in preparation of teaching and learning materials)
- Medicine field (example: Laptops are used by doctors to keep patient records)
- Business field (Electronic commerce uses Laptops for example in marketing and selling products or good and services.)
- Accessing online services (example: National ID application at www.irembo.gov.rw)
- Entertainment field (example: Playing games and watching movies while traveling)
- Communication field (example: when you send and receive an email in a public place)

- Security field (example: Laptop can be used to fight against hackers: people who access personal information without authorization of the owner. Notice that accessing of personal information without permission is punishable crime).

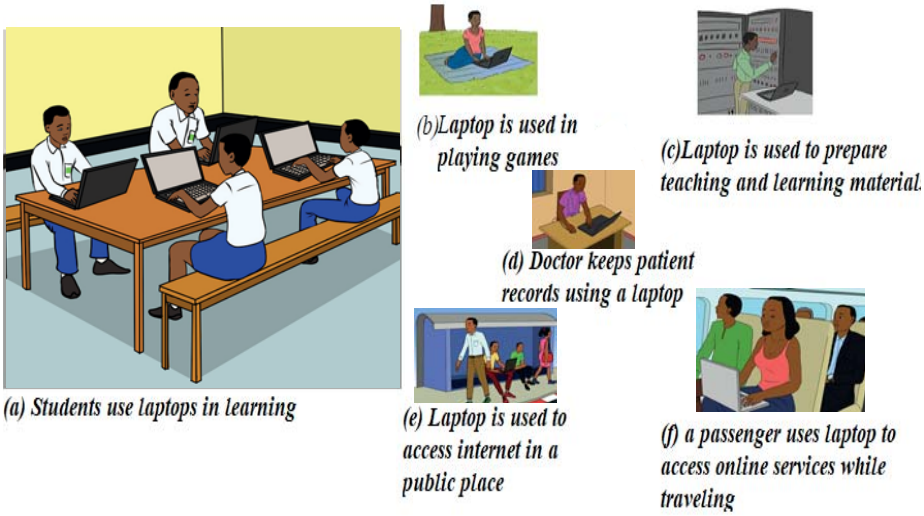


Figure 1.5. Graphical representation of some common uses of laptop

1.2.2. Laptop features

The most significant feature of a laptop is its compact size. The design of the laptop places the keyboard, screen, and internal components into a small portable case.

Common Characteristics/features of Laptops:

- Small and portable;
- An integrated display screen in the lid;
- An integrated keyboard in the base;
- They run on AC power or a rechargeable battery;
- Laptops support hot-swappable drives and peripherals.

1.2.3. Evolution (History) of laptops

In 1983, Gavilan SC is the first machine marketed as a "laptop"

In 1990, Intel introduces 80386SL processor which uses low power; quickly incorporated into many laptops.

In 1991, Apple PowerBook introduces First laptop Ethernet port. Modem speeds reach 14,400 bits/sec.

In 1996, First Pentium laptop with USB, first "three spindle" machine, with floppy,

hard disk, CD-ROM.

In 1999, Laptop manufacturers use PCI slot to enable Wi-Fi connectivity. Apple brands WiFi as AirPort, incorporates a slot into all of its laptops.

In 2005, Laptop computers outsell desktops for the first time.

In 2010, HDMI ports become common on laptops. Apple iPad was introduced.

The components of a laptop

Components may be located in different places for different models.

a. The external components of a laptop

The Laptop and the desktop computers use the same types of ports so that peripherals can be interchangeable. Ports, connections, and drives are located on the front, back, and sides of the laptop due to the compact design. There are status indicators, ports, slots, connectors, bays, jacks, vents, and a keyhole on the exterior of the laptop.

As port a laptop has an AC power port, an ethernet port, a VGA port a HDMI port, a USB port a sound in (microphone) port and a sound out (speaker) port. Beside these ports a laptop has a battery latch at its bottom

On the user view, which is normally visible when a laptop is being used one can see the following components:

1) Web camera
2) LCD
3) Power button
4) Keyboard
5) Touch pad
6) Right Click button
7) Left Click button

Figure 1.5. Laptop- User view

Note:

- The location of the power button may depend on the mark of the laptop
- Some laptops may not have an in-buitl CD/DVD disk drive while in other types, the CD/DVD disk drive can be located on left or right side. For those with no such drive an external drive that can be connected to the laptop through the USB port is often used.
- A desktop monitor can be connected to a laptop. In such cases, using the function key can allow one to toggle the screen from the laptop to the desktop monitor or view both at the same time.

b. The internal components of a laptop

The central circuit board (motherboard) makes up a complex electronic system. A motherboard provides the electrical connections by which the other components of the system communicate. It includes many components such as Central Processing Unit (CPU), Random Access memory (RAM), etc. Laptop motherboards and other internal components installation vary by manufacturer and are proprietary.

Internally and at the top a laptop after removing te cover there is access to the motherboard, speakers, cooling fans, wireless card, CMOS battery and audio board while under the bottom cover there is access to the hard drive and RAM module but this configuration may change from laptop mark to another

Application Activity 1.2

1) Suppose that your computer laptop is not connected to Internet. If there is no wireless connection,

a) which of the following cables can be used to have the connection?

i. VGA (Video Graphic Array) cable;

ii. Network cable;

iii. AC Power cable.

b) Describe the role of the remaining cables.

2) Choose from a laptop in the computer lab one document in PowerPoint and project it by using the Projector of your school.

3) Connect a desktop monitor to a laptop computer and then do the following:

i. Display the contents from the laptop to both screens;

ii. Display the contents from the laptop to only desktop monitor;

Differentiate HDMI port from USB port.

Open a laptop and identify all its internal parts

1.3. Compare and contrast laptop and desktop components

Both desktop and laptop computers can perform most of the same functions. However, these two kinds of computers are built very differently and the parts are not interchangeable. Few components can be shared between desktops and laptops like the expansion devices that are attached to laptop devices with USB ports, parallel ports, and PC Cards.

Figure 1.6: Laptop vs Desktop

Laptop components are proprietary. So you may not be able to use components made by one laptop manufacturer to repair a laptop made by another manufacturer. Therefore, that technicians may have to obtain certification for each laptop manufacturer they support. The following table summarizes the specifications between a laptop and desktop computers.

Feature/Component	Laptop	Desktop
Motherboard form factor	Specialized components (specific form factors)	Standardized components (Universal form factors)
Processor	Processors use less power	Processors use great electricity quantity
	Processors create less heat	Processors create great heat quantity
	Processors don't require large cooling devices	Processors require large cooling devices
Power management	laptop takes electricity from the battery	Desktops remain plugged into a power source
Expansion capability	Expansion devices are attached to laptop devices with USB ports, parallel ports, and PC Cards.	A desktop attaches these devices with USB ports and parallel ports.
RAM slot type	SODIMM	DIMM
Physical Size of Hard Disk	2.5-inch drives	3.5-inch drives

Table 1.1. comparison of laptop and desktop

Expansion devices are attached to laptops and desktops differently. Laptops have limited space so the expansion bays on laptops are designed to allow different types of drives to fit into the same bay. Drives are hot-swappable and are inserted or removed as needed.

As both desktop and laptop computers are electronic devices, they manage power. The optimization of power management can be configured differently through the Control Panel feature of the running operating system. This configuration of power management can also mark the difference between a laptop and a desktop.

Application Activity 1.3

Go through the steps for power option Edit plan. To do this in Windows 10 go through these windows: "Control panel"window, "Hardware and Sound"window, "Power Options" window, Create a Power Plan and Edit Plan Setting. Then:

1. Set your laptop so that it can turn off the display:
   i. After 3 minutes when powered by battery
   ii. After 7 minutes when plugged in
3. Discuss different reasons why there is no need of large cooling devices in a computer lab containing laptops.
4. Discuss the similarities and differences between a laptop and desktop expansion components
5. Open a laptop and a desktop and identify the difference between their motherboards

1.4. SMARTPHONES, PDAs, TABLETS

Learning Activity 1.4.

All companies deal with large amount of data. Using different portable devices, users can collect and access data from the outside of the organization.

1. Identify the portable devices other than laptops that are used in your school to allow teacher and student to access materials on Internet.
2. Discuss the common uses and features of those identified devices in above question

The Smartphones, PDAs, tablets are new types of small personal computing devices that have been introduced in the mid-1990s and these are referred to as handheld computers. This type of computer is named handheld computer because it can fit in one hand while you can operate it with the other hand. Because of its reduced size, the screen of handheld computer is quite small.

1.4.1. Common uses and standard services of Smartphones, PDAs and tablets

Like the laptops, these computers are used by mobile employees whose jobs require them to move from place to place.

They are used in different fields like the following:

- Communication (example: - when you send and receive an email)
  - Voice over Internet Protocol (VoIP) – voice/video calls.
  - Instant text message ex: chatting using WhatsApp
  - Streaming music/video content
  - Communication via social networking applications)
- Entertainment (example: when you play games and watch movies while traveling)
- Business (Electronic commerce uses these computers for example in marketing and selling products or goods and services.)
- Education field (Smartphones, PDAs and tablets are teaching and learning tools. They are used in performing calculations, note taking etc.)
- Smartphones, PDAs and tablets have storage which can store information like contact information (names, addresses, phone numbers, e-mail addresses)
- Smartphones, PDAs and tablets can remind you of appointments (clock, alarm functions)

1.4.2. Common features of Smartphones, PDAs and tablets

Smartphones, PDAs, tablets have operating systems which allows them to run multiple applications. Their Operating systems have features like contacts, email, calendar, appointments, tasks, games, media etc.

Examples of mobile operating systems:

a. Apple's iOS
b. Android OS
c. Windows Phone OS
d. BlackBerry OS

Figure 1.7: Apple's iOS, Android OS, Windows Phone OS and BlackBerry OS Icons

Apart from the operating system, tablets have in common a wi-fi, capability to have their memories expanded, a high resolution display with touch technology and a QUERT keyboard

Application Activity 1.4.

Using a smartphone or a tablet or IPad do the following:

1. Download and install one of these applications from Play Store: WhatsApp, Google Chrome and Facebook.
2. Send an email message to one of the classmates and ask him/her to reply.
3. Search the application called "Polaris Office" in Play Store install it and state its function?

1.4.3. Standards Service of Smartphone, PDA and Tablet

Some of the services that can be found in smartphones, PDA, and tablets include;

- Appointment calendar
- MP3 player
- Web browser
- E-mail access, in addition to text messaging
- Mini-keyboards or onscreen keyboards
- Bluetooth
- Character recognition (allowing for handwritten input)
- Synchronization of information with desktop or laptop computers
- Voice recording
- Digital camera
- Video recording
- Location tracking through GPS(Global Positioning System)
- Wi-Fi
- Microsoft Office (MS) compatible applications (Native with Pocket PC operating systems; Palm operating systems may require third-party software)

1.5. LAPTOP AND PORTABLE DEVICES CLEANING PROCEDURES

Learning Activity 1.5

Observe the figure below and answer the followed questions

1. Discuss what would happen if a drink was spilled into the keyboard of a laptop.
2. How can you prevent this laptop to be damaged?
3. What can you do if the laptop and water are in contact?
4. What can you consider while choosing cleaning materials and products?
5. Describe different laptop parts which can be cleaned.
6. How can you keep waste products after cleaning a laptop?

As laptops are mobile, they are used in different types of environments. Some environments can be hazardous to a laptop. Even eating or drinking around a laptop creates a potentially hazardous condition.

It is important to keep a laptop clean and to ensure that it is being used in the most optimal environment possible. Proper routine cleaning is the easiest, least expensive way to protect and to extend the life of a laptop.

It is very important to use the right products, cleaning materials and procedures when cleaning a laptop. Always read all warning labels on the cleaning products.

Before starting to clean any part of a laptop, turn it off, disconnect all attached devices, disconnect it from the electrical outlet and remove its battery. To remove the battery move the battery lock to the unlocked position, hold the release lever in the unlock position and remove the battery.

To clean laptop keyboards blow compressed air between the keys then slide a soft cloth over the laptop keyboard in order to remove the dirt. The cleaning can be done using compressed air whereby this air is blown into the laptop through the vents. Use tweezers to remove any debris.

To clean LCD screens slide a soft cloth slightly moistered with water or LCD cleaner over the laptop display. A special attention should be taken on using products specifically designed for cleaning LCD displays.

To clean touch pads slide a soft cloth over surface of the touch pad gently with an approved cleaner. Never use a wet cloth and use a soft cloth with only approved cleaning solution.

Application Activity1.5

1. Clean a keyboard and a screen of a laptopin the computer lab.
2. Discuss different Laptop Keyboard, LCD screen and touchpad Cleaning materials, products and Procedures.

END OF UNIT ASSESSMENT ACTIVITY

1. Discuss the advantages and disadvantages of each type of computer below:

Laptop computer

Tablet

Smartphone

2. Describe what is used for standalone and networked laptop computer.
3. Compare smartphone features and cell phone features
4. Is a Laptop hardware maintenance difficult comparing to a desktop computer maintenance? Explain.
5. Give a clear difference between VGA and HDMI connectors.
6. Explain how you can take care your laptop.
7. Exchange a track of music from one device to another. What do you use? Are there other

UNIT 2
COMPLEX DATA STRUCTURE AND ALGORITHM

UNIT2: COMPLEX DATA STRUCTURE AND ALGORITHM

INTRODUCTORY ACTIVITY

Read the following scenario and after answer questions:

In a bank, a Teller has the role of serving the clients. He/she has to organize the banknotes of same values together and coins of same values together that he/she has to serve to clients. When a client arrives in the bank, he/she chooses a line to join in the front of a Teller and the line is formed according to the arrival of clients.

Mr. KARENZI wants to pay 78450 Rwf as his school fees to the bank and he must organize banknotes and coins before deposit money to facilitate the teller to not mix the banknotes and coins

Answer the following questions:

1. Observe the above figure and describe what you see;
2. Where a new arrived client will line up?
3. Using the above figure who is going to be served first by the teller 1 or 2;
4. Discuss how KARENZI is going to organize bank notes and coins before depositing;
5. Discuss how the teller is going to keep money deposited by Karenzi

2.1. Introduction to Data Structures

Data Structure is a way of collecting and organizing data in such a way that we can perform operations on these data in an effective way. Data Structures is about rendering data elements in terms of some relationship, for better organization and storage.

Example: Library is composed of elements (books) to access a particular book requires knowledge of the arrangement of the books.

2.1.1. Basic types of Data Structures

Anything that can store data can be called as a data structure, hence Integer, Float, Boolean, Char etc., all are data structures. They are known as **Primitive Data Structures**. And we also have some complex Data Structures, which are used to store large and connected data and are called **non primitive** data structure or **Abstract Data Structure**.

Example: Array, Linked List, Tree, Graph, Stack, Queue etc.

All these data structures allow us to perform different operations on data. We select these data structures based on which type of operation is required.

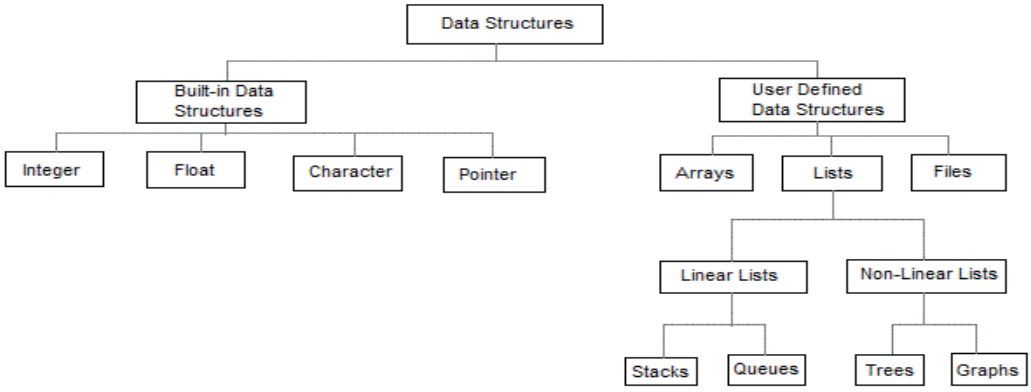


Figure 2.1. Abstract data structure

2.1.2. Importance of data structures

The data structures have the following importance for programming:

- Data structures study how data are stored in a computer so that operations can be implemented efficiently
- Data structures are especially important when there is a large amount of information to deal with.
- Data structures are conceptual and concrete ways to organize data for efficient storage and manipulation

2.1.3. Data Structure Operations

Beside the simple operations performed on data (unary operators, binary operators and precedence operators) seen in Senior Four, the data appearing in our data structures are processed by means of other operations. The following are some of the most frequently used operations.

Traversing: Accessing each record exactly once so that certain items in the record may be processed.

- **Searching**: Finding the location of the record with a given key value or finding the locations of all records which satisfy one or more conditions
- **Inserting:** Adding a new record to the structure.
- **Deleting:** Removing a record from the structure.
- **Sorting:** Arranging the records in some logical order (e.g., in numerical order according to some NUMBER key, such as social security number or account number).
- **Merging:** Combining the records of two different sorted lists into a single sorted list.

It is important for every Computer Science student to understand the concept of *Information* and how it is organized or how it can be utilized.

If we arrange some data in appropriate sequence, then it forms a Structure and gives us a meaning. This meaning is called information. The basic unit of information in computer Science is a bit, binary digit. So we found two things in information: One is **Data** and the other is **Structure**

2.2 Array of elements in data structure

LEARNING ACTIVITY 2.1.

Study the following table

	Mathematics	Physics	Computer
Niyibizi			
Urakaza			
Shyaka			
kwizera		95	

	Mathematics	Physics	Computer
Niyibizi			
Urakaza			
Shyaka			
kwizera		95	

Answer the following questions.

1. Suggest a name for the above table?
2. How many rows and columns in the above table?
3. Identify the location of the value 95 in the above table?

Arrays hold a series of data elements, usually of the same size and data type. Individual elements are accessed by their position in the array. The position is given by an **index**, which is also called a **subscript**. The index usually uses a consecutive range of integers, (as opposed to an associative array) but the index can have any ordinal set of values

Some arrays are *multi-dimensional,* meaning they are indexed by a fixed number of integers, for example by a tuple of four integers. Generally, one- and two-dimensional arrays are the most common. Most programming languages have a built-in *array* data type.

In computer programming, a group of homogeneous elements of a specific data type is known as **an array,** one of the simplest data structures.

2.2.1. Arrays dimensions:

A dimension of array is a direction in which you can vary the specification of an array›s elements. An array can be of one or many dimensions.

a. One-dimensional array.

Example: a [3]

It is an array called a with 3 elements. The elements of the array are logically represented by the name of the array with in brackets its index or positions. For the one dimensional array a with 3 elements, elements are written like a[0], a[1] and a[2]. The indexes start always by 0. Graphically, it is

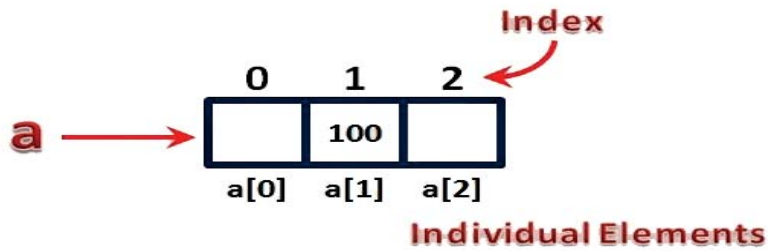


Figure 2.2.: One-dimensional array

b. Multi-dimensional arrays

Example:

Two dimensional array ex: a[2][7]as Integer

Three dimensional array ex: a[3][2][5]as Integer

2.2.2. Two dimensional array

A Two dimensional Array is a collection of a fixed number of elements (components) of the same type arranged in two dimensions.

The two dimensional array is also called a matrix or a table. The intersection of a column and a row is called a cell. The numbering of rows and columns starts by 0

Example:
The array a[3][4] is an array of 3 rows and 4 columns. In matrix representation, it looks like the following:

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Figure 2.3.: Two-dimensional array

a[i][j] means that it is an element located at the intersection of row i and column j.

Each element is identified by its row and column

2.2.3. Two dimensional array declaration

The syntax for declaring a two-dimensional array is:

SET Array name=Array [row size][column size] of Data type

Where row size and column size are expressions yielding positive integer values, the two expressions row size and column size specify the number of rows and the number of columns, respectively, in the array.

Examples:

1. SET a=Array[2][7] of Integer
2. SET b=Array[4][3] of Integer

Notice that the number of elements **=row size*column size**

The examples above defines respectively arrays named a and b with 14 (2*7) elements and 12 (4*3) elements. The expression marks [0] [0] will access the first element of the matrix marks and marks [1] [6] will access the last row and last column of array a.

If the array marks if filled with the elements 80, 75, 76, 75, 55, 70, 55, 70, 65, 85, 35 and 59, we get the following table.

2.2.4. Two dimensional array initialization

When a two-dimension array has been declared, it needs to be assigned values for its elements. This action of assigning values to array elements is called initialization. Two-Dimensional array is initialized by specifying bracketed values for each row. For the case of array of integers, the default values of each element is 0 while for others the default values for other types is not known.

For the case of an array of 3 rows and 4 columns, if the name is A and the type of elements is integer, the initialization is:

int A[3][4]={{A[0][0], A[0][1], A[0][2], A[0][3]}, /* initializers for row indexed by 0 */

{A[1][0], A[1][1], A[1][2], A[1][3]}, /* initializers for row indexed by 1*/

{A[2][0], A[2][1], A[2][2], A[2][3]} /* initializers for row indexed by 2 */

};

The nested braces, which indicate the intended row, are optional. The following initialization is equivalent to the previous initialization.

Int A[3][4]={A[0][0], A[0][1], A[0][2], A[0][3], A[1][0], A[1][1], A[1][2], A[1][3], A[2][0], A[2][1], A[2][2], A[2][3]};

An example of marks for 3 students in 4 courses. The array will look like this:

	col 0	col 1	col 2	col 3
row 0	18	16	14	19
row 1	10	20	12	15
row 2	15	17	18	16

Table 2.1.: Array initialization

If the table is called marks, marks [1][3]=15; where the row position is 1 and the column position is 3.

The expression marks [0] [0] will access the first element of the matrix marks and marks [2] [3] will access the last row and last column.

Syntax to initializing two-dimensional arrays

The two dimensional array marks which will hold 12 elements can be initialized like:

BEGIN
SET marks=Array[3] [4] of Integer
marks[0][0]=18

marks[0][1]=16
marks[0][2]=14
marks[0][3]=19
marks[1][0]=10
marks[1][1]=20
marks[1][2]=12
marks[1][3]=15
marks[2][0]=15
marks[2][1]=17
marks[2][2]=18
marks[2][3]=16
END

APPLICATION ACTIVITY 2.1.

1. Four secondary schools in RUHANGO District are displaying the number of students per year that they registered in 2015, 2016, 2017 and 2018.
   a. How many rows and columns will be in the table that contains the displayed data?
   b. Declare an array called population to contain the needed data.
   c. Initialize the array population with numbers of your choice.
2. Given below the case of array a:

a[0][0]	1	a[1][0]	54
a[0][1]	2	a[1][1]	23
a[0][2]	3	a[1][2]	11
a[0][3]	13	a[1][3]	45
a[0][4]	17	a[1][4]	31
a[0][5]	34	a[1][5]	65
a[0][6]	19	a[1][6]	26

   a. Give the size of the above array;
   b. Declare array a;
   c. Initialize the above array.

2.2.5. Accessing two dimensional array elements

LEARNING ACTIVITY 2.2.

What do the following statements mean:

WRITE marks[1][2] ;

What are the meanings of the above statements

- **Reading (storing) two dimensional array elements**

  There exist two ways of accessing the elements of an array. The elements' values of the array can come from the input devices or from a given file so that these elements are written into the array. The function used is READ or other versions of it, depending on the programming languages used for programs writing.

  For example READ marks[1][2] stores a value in second row, third column location of the array marks.

  To store multiple values into array you must use a FOR loop.

Syntax:

FOR i=initial value TO upperlimit DO
FOR j=initial value TO upperlimit DO
READ array name[i][j]
i=i+1
j=j+1
END FOR
END FOR

Where i refers to the row position and j refers to the column position.

Initial value=lower limit

Example: Write an algorithm of a program that allows the user to enter (store) two dimensional array elements

BEGIN
SET marks=Array [3] [8] of Integer
Use variable i as integer
Use variable j as integer
FOR i=0 TO 2 DO
FOR j=0 TO 7 DO
READ marks[i] [j]
i=i+1
j=j+1

END FOR
END FOR
END

Writing (displaying) two dimensional array elements

To access two dimensional array elements' values, we can also take the elements' values of an array residing in the memory of the computer and write them in any output device or a given file.

To write (display) elements from an array in an output device or other file, use a WRITE function or other versions of it depending on the programming languages used for programs writing together with the array name and location (index) of the element. In that case the elements of a two dimensional array can be displayed by the following syntax:

WRITE Array name [i][j]

Where i refers to the row position and j refers to the column position.

To display a single value of an array you must provide the array name and index to write operation example:

Example for writing on screen a two dimensional array element

WRITE a[1][2], i=1 and j=2).

Example: Write an algorithm of a program that displays stored two dimensional array marks' elements

BEGIN
SET a=Array[3][8] of integer
Use variable i As integer
Use variable j As integer
WRITE "stored two dimensional array elements are:"
FOR i=0 to 2DO
FOR j=0 to 7DO
WRITE a[i][j]
i=i+1
j= j+1
END FOR
END FOR
END

Algorithm for reading and printing values in arrays

Example: Write an algorithm that allows the user to enter(store) two dimensional array elements and displays those stored elements.

```
BEGIN
SET Scores=Array[3][8] of Integer
Use variable i  As integer
Use variable j As integer
WRITE "Enter  two dimensional array
  elements:"
FOR i=0 TO 2 DO
 FOR j=0 TO 7 DO
READ Scores[i][j]
i=i+1
j=j+1
END FOR
END FOR
WRITE "stored  two dimensional array
  elements are:"
FOR  i=0 to 2 DO
FOR j=0 to 7 DO
WRITE  a[i][j]
i=i+1
j= j+1
END FOR
END FOR
END
```

APPLICATION ACTIVITY 2.2.

1. Write an algorithm to do the following operations on a two dimensional array A of size m x n.
   - Declare the array A;
   - Read the element 'values into matrix of size m x n
   - Output the elements' values of matrix of size m x n on the screen;
   - Calculate the sum of all elements' values of matrix of size m x n;
   - Write a programme in C++ Language where you apply the above questions
2. Write an algorithm that allows a user to input data through input device and display the smallest element' s value on the screen.
3. Write a programme in C++ language that implements the above algorithm.
4. Consider a 2 by 3 array T
   a. Write a declaration of T
   b. How many rows does T have?
   c. How many columns does T have?
   d. How many elements does T have?
   e. Write name of elements in second row
   f. Write the name of element in third column
   g. Write a single statement that sets the elements of T in row 1and column 2 to zero
   h. Write a statement that inputs the values of T from keyboard
   i. Write a statement that displays the elements of first row of T

2.3. ABSTRACT DATA STRUCTURE

LEARNING ACTIVITY 2.3

1. Observe and analyze the above figure
2. Explain different procedures that you can use to arrange the above books.

2.3.1. Definition of Abstract Data type in data structure.

Abstract Data structure is a mathematical model of a data structure that specifies the type of data stored, the operations supported on them and the types of parameters of the operations with no implementation. Each program language has its own way to implement abstract data type.

2.3.2 Importance of abstract data type in data structures

- Data structures study how data are stored in a computer so that operations can be implemented efficiently;
- Abstract Data structures are especially important when you have a large amount of information;
- Conceptual and concrete ways to organize data for efficient storage and manipulation

2.3.3. Abstract Data type Operations

The data appearing in our data structures are processed by means of certain operations. Following are some of the most frequently used operations.

- **Searching:** Finding the location of the record with a given key value or finding the locations of all records which satisfy one or more conditions;
- **Sorting:** Arranging the records in some logical order .

2.4. List

2.4.1. Introduction

A list is a collection of data in which each element of the list contains the location of the next element.

In a list, each element contains two parts: data and link to the next element. The data parts of the list hold the value information and data to be processed. The link part of the list is used to chain the data together and contains a pointer (an address) that identifies the next element in the list.

In the list you can do the following operations:

- LIST: Creates an empty list;
- INSERT: inserts an element in the list;
- DELETE: deletes an element from the list;
- RETRIEVE: retrieves an element from the list;
- TRAVERSE: traverses the list sequentially;
- EMPTY: checks the status of the list.

Example of a list:

A bank has 10 branches created one after another at different dates. The Bank Inspector who wants to visit all of them can follow the order of their creation. He/she will be informed only about the name and location of the first branch. When he reaches that branch, the Branch Manager will tell him/her the address of the next branch, and so one.

After visiting all branches, the Bank Inspector will realize that he/she has a list of the bank branches.

Graphically, it will be represented like a list where each branch is considered as a node containing the names of the branch and the address of the next branch.

Difference between Arrays and Lists

A list is an ordered set of data. It is often used to store objects that are to be processed sequentially.

An array is an indexed set of variables, such as dancer [1], dancer [2], dancer [3] It is like a set of boxes that hold things. A list is a set of items. An array is a set of variables that each store an item.

In a list, the missing spot is filled in when something is deleted. While in an array, an empty variable is left behind when something is deleted.Simply a list is a sequence of data, and linked list is a sequence of data linked with each other.

2.4.2 Linked list

This is a dynamic table/ data structure used to hold a sequence. Its characteristics are the following:

- Items forming a sequence are not necessarily held in contiguous data location or in the order in which they occur in the sequence.
- Each item in the list is called a node and contains an information field and a new address field called a link or pointer field.
- Information field holds actual data for list item; link field contains the address of the next item.
- The link field in the last item indicates that there are not further items. E.g. has a value of 0.
- Associated with the list is a pointer variable which points to the first node in the list.
- We use linked list in linking files, for example, in a hard disk.
- It can grow or shrink in size during execution of a program.
- It can be made just as long as required
- It does not waste memory space.

Linked list is classified into three types as shown below:

- Single linked lists
- Double linked lists
- Circular linked list

1. Single linked lists

Single linked list is a sequence of elements in which every element has link to its next element in the sequence. In any single linked list, the individual element is called as "Node". Every "Node" contains two fields, data and next. The data field is used to store actual value of that node and next field is used to store the address of the next node in the sequence

Figure 2.6: Node representation

In a single linked list, the address of the first node is always stored in a reference node known as "front" (Sometimes it is also known as "head").Always next part (reference

part) of the last node must be NULL

Figure 2.7: Single Linked List representation

Algorithm in Single Linked List

Creating and read an element in a NODE

BEGIN
struct node{
int info
struct node next
} head, current, temp
head=null
temp=null
current=null
Begin sub

Step 1 Read the element into x
Step 2 Create a temp node in the memory
temp =(struct node)sizeof (node)
Step 3 Assign the values in temp node as follows
temp -> info =x
temp ->next=null
End sub
END

Inserting at Specific location in the list (After a Node)

We can use the following steps to insert a new node after a node in the single linked list

- **Step 1:** Create a **newNode** with given value.
- **Step 2:** Check whether list is **Empty** (**head** == **NULL**)
- **Step 3:** If it is **Empty** then, set **newNode => next** = **NULL** and **head** = **newNode**.
- **Step 4:** If it is **Not Empty** then, define a node pointer **temp** and initialize with **head**.
- **Step 5:** Keep moving the **temp** to its next node until it reaches to the node

after which we want to insert the newNode (until **temp1 => data** is equal to **location**, here location is the node value after which we want to insert the newNode).

- **Step 6:** Every time check whether **temp** is reached to last node or not. If it is reached to last node then display '**Given node is not found in the list!!! Insertion not possible!!!'** and termin ate the function. Otherwise move the **temp** to next node.
- **Step 7:** Finally, Set '**newNode => next** = **temp => next**' and '**temp => next** = **newNode**'

Deleting a Specific Node from the list

We can use the following steps to delete a specific node from a single linked list

- **Step 1:** Check whether list is Empty (head == NULL)
- **Step 2:** If it is Empty then, display 'List is Empty!!! Deletion is not possible' and terminate the function.
- **Step 3:** If it is Not Empty then, define two Node pointers 'temp1' and 'temp2' and initialize 'temp1' with head.
- **Step 4:** Keep moving the temp1 until it reaches to the exact node to be deleted or to the last node. And every time set 'temp2 = temp1' before moving the 'temp1' to its next node.
- **Step 5:** If it is reached to the last node then display 'Given node not found in the list! Deletion not possible!!!' And terminate the function.
- **Step 6:** If it is reached to the exact node which we want to delete, then check whether list is having only one node or not
- **Step 7:** If list has only one node and that is the node to be deleted, then set head = NULL and delete temp1 (free (temp1)).
- **Step 8:** If list contains multiple nodes, then check whether temp1 is the first node in the list (temp1 == head).
- **Step 9:** If temp1 is the first node then move the head to the next node (head = head => next) and delete temp1.
- **Step 10:** If temp1 is not first node then check whether it is last node in the list (temp1=> next == NULL).
- **Step 11:** If temp1 is last node then set temp2 => next = NULL and delete temp1 (free (temp1)).
- **Step 12:** If temp1 is not first node and not last node then set temp2 => next = temp1=> next and delete temp1 (free (temp1)).

Displaying a Single Linked List

We can use the following steps to display the elements of a single linked list

- **Step 1:** Check whether list is Empty (head == NULL)
- **Step 2:** If it is Empty then, display 'List is Empty!!!' and terminate the function.
- **Step 3:** If it is Not Empty then, define a Node pointer 'temp' and initialize with head.
- **Step 4:** Keep displaying temp=> data with an arrow (->) until temp reaches to the last node
- **Step 5:** Finally display temp => data with arrow pointing to NULL (temp => data -> NULL).

2. Double linked lists

Double linked list is a sequence of elements in which every element has links to its previous element and next element in the sequence. So, we can traverse forward by using next field and can traverse backward by using previous field. Every node in a double linked list contains three fields and they are shown in the following figure

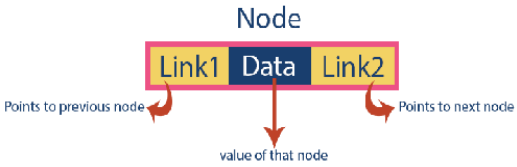


Figure 2.8: Representation of double linked list

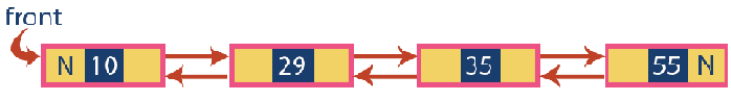


Figure 2.9: Double linked list representation

Algorithm in Double Linked List

Inserting at Specific location in the list (After a Node)

We can use the following steps to insert a new node after a node in the double linked list

- **Step 1:** Create a newNode with given value.
- **Step 2:** Check whether list is Empty (head == NULL)
- **Step 3:** If it is Empty then, assign NULL to newNode => previous&newNode => next and newNode to head.
- **Step 4:** If it is not Empty then, define two node pointers temp1&temp2 and

initialize temp1 with head.

- **Step 5:** Keep moving the temp1 to its next node until it reaches to the node after which we want to insert the newNode (until temp1 => data is equal to location, here location is the node value after which we want to insert the newNode).
- **Step 6:** Every time check whether temp1 is reached to the last node. If it is reached to the last node then display 'Given node is not found in the list!!! Insertion not possible!!!' and terminate the function. Otherwise move the temp1 to next node.
- **Step 7:** Assign temp1 => next to temp2, newNode to temp1=> next, temp1 to newNode => previous, temp2 to newNode => next and newNode to temp2 => previous.

Deleting a Specific Node from the list

We can use the following steps to delete a specific node from the double linked list

- **Step 1:** Check whether list is Empty (head == NULL)
- **Step 2:** If it is Empty then, display 'List is Empty!!! Deletion is not possible' and terminate the function.
- **Step 3:** If it is not Empty, then define a Node pointer 'temp' and initialize with head.
- **Step 4:** Keep moving the temp until it reaches to the exact node to be deleted or to the last node.
- **Step 5:** If it is reached to the last node, then display 'Given node not found in the list! Deletion not possible!!!' and terminate the fuction.
- **Step 6:** If it is reached to the exact node which we want to delete, then check whether list is having only one node or not
- **Step 7:** If list has only one node and that is the node which is to be deleted then set head to NULL and delete temp (free(temp)).
- **Step 8:** If list contains multiple nodes, then check whether temp is the first node in the list (temp == head).
- **Step 9:** If temp is the first node, then move the head to the next node (head = head => next), set head of previous to NULL (head => previous = NULL) and delete temp.
- **Step 10:** If temp is not the first node, then check whether it is the last node in the list (temp => next == NULL).
- **Step 11:** If temp is the last node then set temp of previous of next to NULL (temp => previous => next = NULL) and delete temp (free(temp)).

- **Step 12:** If temp is not the first node and not the last node, then set temp of previous of next to temp of next (temp => previous => next = temp => next), temp of next of previous to temp of previous (temp => next => previous = temp => previous) and delete temp (free(temp)).

Displaying a Double Linked List

We can use the following steps to display the elements of a double linked list

- **Step 1:** Check whether list is **Empty** (**head** == **NULL**)
- **Step 2:** If it is **Empty**, then display **'List is Empty!!!'** and terminate the function.
- **Step 3:** If it is not Empty, then define a Node pointer **'temp'** and initialize with **head**.
- **Step 4:** Display **'NULL <= '**.
- **Step 5:** Keep displaying **temp → data** with an arrow (⇔) until **temp** reaches to the last node
- **Step 6:** Finally, display **temp → data** with arrow pointing to **NULL** (**temp → data ---> NULL**).

3. Circular Linked List

In circular linked list, every node points to its next node in the sequence but the last node points to the first node in the list

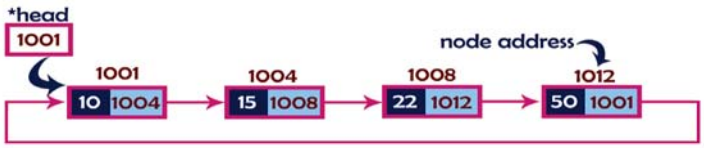


Figure 2.10: Circular Linked List representation

Algorithm in Circular Linked List

Inserting at Specific location in the list (After a Node)

We can use the following steps to insert a new node after a node in the circular linked list...

- **Step 1:** Create a **newNode** with given value.
- **Step 2:** Check whether list is **Empty** (**head** == **NULL**)
- **Step 3:** If it is **Empty** then, set **head** = **newNode** and **newNode** → **next** = **head**.
- **Step 4:** If it is **Not Empty** then, define a node pointer **temp** and initialize with **head**.

- **Step 5:** Keep moving the **temp** to its next node until it reaches to the node after which we want to insert the newNode (until **temp1 → data** is equal to **location**, here location is the node value after which we want to insert the newNode).
- **Step 6:** Every time check whether **temp** is reached to the last node or not. If it is reached to last node then display **'Given node is not found in the list!!! Insertion not possible!!!'** and terminate the function. Otherwise move the **temp** to next node.
- **Step 7:** If **temp** is reached to the exact node after which we want to insert the newNode then check whether it is last node (temp → next == head).
- **Step 8:** If **temp** is last node then set **temp → next** = **newNode** and **newNode → next** = **head**.
- **Step 8:** If **temp** is not last node then set **newNode → next** = **temp → next** and **temp → next** = **newNode**.

Deleting a Specific Node from the list

We can use the following steps to delete a specific node from the circular linked list...

- **Step 1:** Check whether list is **Empty** (**head** == **NULL**)
- **Step 2:** If it is **Empty** then, display **'List is Empty!!! Deletion is not possible'** and terminate the function.
- **Step 3:** If it is **Not Empty** then, define two Node pointers **'temp1'** and '**temp2'** and initialize '**temp1**' with **head**.
- **Step 4:** Keep moving the **temp1** until it reaches to the exact node to be deleted or to the last node. And every time set '**temp2 = temp1**' before moving the '**temp1**' to its next node.
- **Step 5:** If it is reached to the last node then display **'Given node not found in the list! Deletion not possible!!!'**. And terminate the function.
- **Step 6:** If it is reached to the exact node which we want to delete, then check whether list is having only one node (**temp1 → next** == **head**)
- **Step 7:** If list has only one node and that is the node to be deleted then set **head** = **NULL** and delete **temp1** (**free(temp1)**).
- **Step 8:** If list contains multiple nodes then check whether **temp1** is the first node in the list (**temp1 == head**).
- **Step 9:** If **temp1** is the first node then set **temp2** = **head** and keep moving **temp2** to its next node until **temp2** reaches to the last node. Then set **head = head → next**, **temp2 → nex**t = **head** and delete **temp1**.
- **Step 10:** If **temp1** is not first node then check whether it is last node in the list

(**temp1 → next == head**).

- **Step 11:** If **temp1** is last node then set **temp2 → next** = **head** and delete **temp1** (**free (temp1)**).
- **Step 12:** If **temp1** is not first node and not last node then set **temp2 → next** = **temp1 → next** and delete **temp1** (**free (temp1)**).

Displaying a circular Linked List

We can use the following steps to display the elements of a circular linked list

- **Step 1:** Check whether list is **Empty** (**head** == **NULL**)
- **Step 2:** If it is **Empty**, then display **'List is Empty!!!'** and terminate the function.
- **Step 3:** If it is **Not Empty** then, define a Node pointer **'temp'** and initialize with **head**.
- **Step 4:** Keep displaying **temp → data** with an arrow (**--->**) until **temp** reaches to the last node
- **Step 5:** Finally display **temp → data** with arrow pointing to **head → data**.

2.5. Queue

A queue is a linear list in which data can only be inserted at one end, called the REAR, and deleted from the other end, called the FRONT. These restrictions ensure that the data is processed through the queue in the order in which it is received. In an other words, a queue is a structure in which whatever goes fist comes out first (**first in, first out(FIFO)** structure).

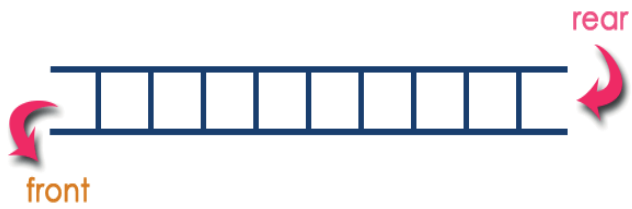


Figure 2.11: Queue representation

2.5.1 Types of queues

There exist two types of queues:

- Linear queue
- Circular queue

a. Linear Queue

Queue data structure is a linear data structure in which the operations

are performed based on FIFO principle. **Queue Operations using Array** before we implement actual operations, first follow the below steps to **create an empty queue**.

- **Step 1:** Include all the **header files** which are used in the program and define a constant **'SIZE'** with specific value.
- **Step 2:** Declare all the **user defined functions** which are used in queue implementation.
- **Step 3:** Create a one dimensional array with above defined SIZE (**int queue[SIZE]**)
- **Step 4:** Define two integer variables **'front'** and '**rear**' and initialize both with **'-1'**. (**int front = -1, rear = -1**)
- **Step 5:** Then implement main method by displaying menu of operations list and make suitable function calls to perform operation selected by the user on queue.

enQueue(value) - Inserting value into the queue

- Step 1: Check whether queue is FULL. (rear == SIZE-1)
- Step 2: If it is FULL, then display "Queue is FULL!!! Insertion is not possible!!!"" and terminate the function.
- Step 3: If it is NOT FULL, then increment rear value by one (rear++) and set queue [rear] = value.

deQueue() - Deleting a value from the Queue

- Step 1: Check whether queue is EMPTY. (front == rear)
- Step 2: If it is EMPTY, then display "Queue is EMPTY!!! Deletion is not possible!!!" and terminate the function.
- Step 3: If it is NOT EMPTY, then increment the front value by one (front ++). Then display queue[front] as deleted element. Then check whether both front and rear are equal (front == rear), if it TRUE, then set both front and rear to '-1' (front = rear = -1).

display() - Displays the elements of a Queue

We can use the following steps to display the elements of a queue

- **Step 1:** Check whether **queue** is **EMPTY**. (**front == rear**)
- **Step 2:** If it is **EMPTY**, then display **"Queue is EMPTY!!!"** and terminate the function.
- **Step 3:** If it is **NOT EMPTY**, then define an integer variable '**I'** and set '**i** =

front+1'.

- **Step 3:** Display '**queue[i]'** value and increment '**I**' value by one (**i++**). Repeat the same until '**I**' value is equal to **rear** (**i**<= **rear**)

b. Circular Queue

Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle.

Figure 2.12: Circular Queue representation

Circular Queue Operation

To implement a circular queue data structure using array, we first **create it**.

- **Step 1:** Include all the **header files** which are used in the program and define a constant **'SIZE'** with specific value.
- **Step 2:** Declare all **user defined functions** used in circular queue implementation.
- **Step 3:** Create a one dimensional array with above defined SIZE (**int cQueue[SIZE]**)
- **Step 4:** Define two integer variables **'front'** and '**rear**' and initialize both with '**-1'**. (**int front = -1, rear = -1**)
- **Step 5:** Implement main method by displaying menu of operations list and make suitable function calls to perform operation selected by the user on circular queue.

enQueue(value) - Inserting value into the Circular Queue

- **Step 1:** Check whether **queue** is **FULL**. (**(rear == SIZE-1 && front == 0) || (front == rear+1)**)
- **Step 2:** If it is **FULL**, then display **"Queue is FULL!!! Insertion is not possible!!!"** and terminate the function.
- **Step 3:** If it is **NOT FULL**, then check **rear == SIZE - 1 && front != 0** if it is **TRUE**, then set **rear = -1**.
- **Step 4:** Increment **rear** value by one (**rear++**), set **queue[rear]** = **value** and check ‹**front == -1**' if it is **TRUE**, then set **front = 0**.

deQueue() - Deleting a value from the Circular Queue

- **Step 1:** Check whether **queue** is **EMPTY**. (**front == -1 && rear == -1**)
- **Step 2:** If it is **EMPTY**, then display **"Queue is EMPTY!!! Deletion is not possible!!!"** and terminate the function.
- **Step 3:** If it is **NOT EMPTY**, then display **queue[front]** as deleted element and increment the **front** value by one (**front ++**). Then check whether **front == SIZE**, if it is **TRUE**, then set '**front = 0'**. Then check whether both '**front – 1'** and **rear** are equal (**front -1** == **rear**), if it **TRUE**, then set both **front** and **rear** to '**-1**' (**front** = **rear** = **-1**).

display () - Displays the elements of a Circular Queue

- **Step 1:** Check whether **queue** is **EMPTY**. (**front == -1**)
- **Step 2:** If it is **EMPTY**, then display **"Queue is EMPTY!!!"** and terminate the function.
- **Step 3:** If it is **NOT EMPTY**, then define an integer variable '**i**' and set '**i** = **front**'.
- **Step 4:** Check whether '**front <= rear**', if it is **TRUE**, then display '**queue[i]**' value and increment '**I**' value by one (**i++**). Repeat the same until '**i <= rear'** becomes **FALSE**.
- **Step 5:** If '**front <= rear**' is **FALSE**, then display '**queue[i]'** value and increment '**I**' value by one (**i++**). Repeat the same until '**i <= SIZE – 1**' becomes **FALSE**.
- **Step 6:** Set **i** to **0**.
- **Step 7:** Again display '**cQueue[i]**' value and increment **i** value by one (**i++**). Repeat the same until '**i <= rear**' becomes **FALSE**.

2.5.2 Queue operations

There are three main operations related to queues.

- Enqueue: the enqueue operation inserts an items at the rear of the queue
- Dequeue: the dequeue operation deletes the item at the front of the queue
- Display: show elements in the array

2.6 Stack

A stack is a restricted linear list in which all additions and deletions are made at one end, the top. If we insert a series of data items into a stack and then remove them, the order of the data is reversed. Data input as 5, 10, 15, 20, for example would be removed as 20, 15, 10, and 5. This reversing attribute is why stacks are known as Last in, First out (LIFO) data structures.

We use many different types of stacks in our daily lives. We often talk of a stack of coins, stack of books on a table and stack of plates in a kitchen. Any situation in which

we can only add or remove an object at the top is a stack. If we want to remove an object other than the one at the top, we must first remove all objects above it. The following charts illustrates cases of stack

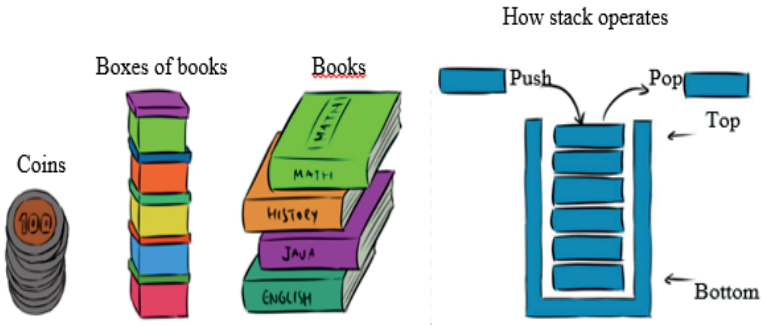


Figure 2.13: cases of stack

A Stack is a Last in First out (LIFO) dynamic table or data structure. It has the following characteristics:

- List of the same kind of elements;
- Addition and deletion of elements occur only at one end, called the top of the stack;
- Computers use stacks to implement method calls;
- Stacks are also used to convert recursive algorithms into non recursive algorithm.

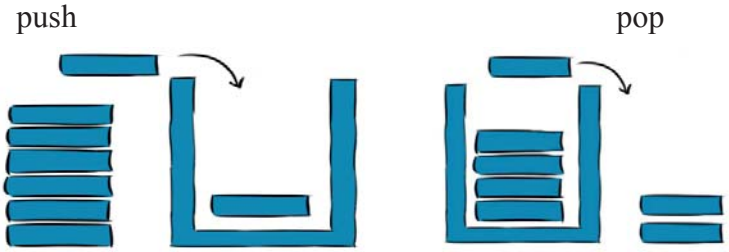


Figure 2.14: stacks' operations (Push and Pop)

2.6.1 Operations performed on stacks

The different operations performed on stacks are as follows:

- **Push**: adds an element to the stack
- **Pop**: removes an element from the stack
- **Peek**: display at top element of the stack

Stack Operations using Array

Before implementing actual operations, first follow the below steps to **create an empty stack**.

Step 1: Include all the **header files** which are used in the program and define a constant **'SIZE'** with specific value.

- **Step 2:** Declare all the **functions** used in stack implementation.
- **Step 3:** Create a one dimensional array with fixed size (**int stack[SIZE]**)
- **Step 4:** Define a integer variable '**top'** and initialize with **'-1'**. (**int top = -1**)
- **Step 5:** In main method display menu with list of operations and make suitable function calls to perform operation selected by the user on the stack.

push(value) - Inserting value into the stack

- **Step 1:** Check whether **stack** is **FULL**. (**top == SIZE-1**)
- **Step 2:** If it is **FULL**, then display **"Stack is FULL!!! Insertion is not possible!!!"** and terminate the function.
- **Step 3:** If it is **NOT FULL**, then increment **top** value by one (**top++**) and set stack [top] to value (**stack[top] = value**).

pop() - Delete a value from the Stack

- **Step 1:** Check whether **stack** is **EMPTY**. (**top == -1**)
- **Step 2:** If it is **EMPTY**, then display **"Stack is EMPTY!!! Deletion is not possible!!!'"** and terminate the function.
- **Step 3:** If it is **NOT EMPTY**, then delete **stack [top]** and decrement **top** value by one(**top-**).

Display() - Displays the elements of a Stack

- **Step 1:** Check whether **stack** is **EMPTY**. (**top == -1**)
- **Step 2:** If it is **EMPTY**, then display **"Stack is EMPTY!!!"** and terminate the function.
- **Step 3:** If it is **NOT EMPTY**, then define a variable '**I**' and initialize with top. Display **stack[i]** value and decrement **i** value by one (**i--**).
- **Step 3:** Repeat above step until **i** value becomes '0'.

2.6.2 Stack exceptions

Adding an element to a full stack and removing an element from an empty stack would generate errors or exceptions:

- **Stack overflow exception:** it occurs if you try to add an item to stack or queue which is already full.
- **Stack underflow exception:** it occurs if you try to remove an item from an empty queue or stack.

Notec 1: Stack and Queue data structures can be implement in two ways: By using array or by using linked list.

Notice 2: When stack or Queue is implemented using array, that stack or Queue can organize only a limited number of elements. When stack or Queue is implemented using linked list, that stack or Queue can organize an unlimited number of elements.

2.7 Tree

In linear data structure, data is organized in sequential order and in non-linear data structure; data is organized in random order. Tree is a very popular data structure used in wide range of applications. A tree data structure can be defined as follows.

In a tree data structure, if we have **N** number of nodes then we can have a maximum of **N-1** number of links.

Tree data structure is a collection of data (Node) which is organized in hierarchical structure and this is a recursive definition.

Example

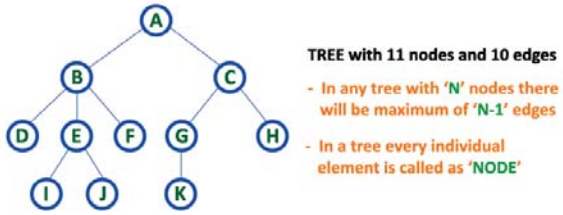


Figure 2.15.: Tree representation

2.7.1 Tree Terminology

In a tree data structure, we use the following terminology:

a. Root

In a tree data structure, the first node is called as **Root Node**. Every tree must have root node..

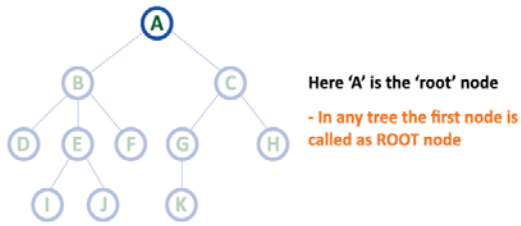


Figure 2.16: Tree Terminology representation

b. Edge

In a tree data structure, the connecting link between any two nodes is called an **Edge**. In a tree with '**N**' number of nodes there will be a maximum of '**N-1**' number of edges.

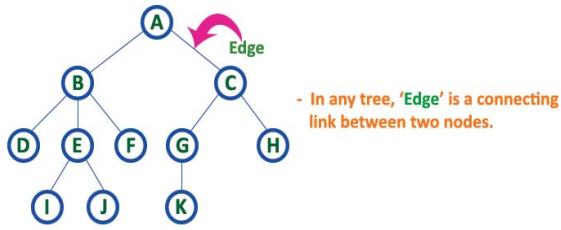


Figure 2.17: Edge representation

c. Parent

In a tree data structure, the node which is predecessor of any node is called a **PARENT NODE**. In simple words, the node which has branch from it to any other node is called as parent node.

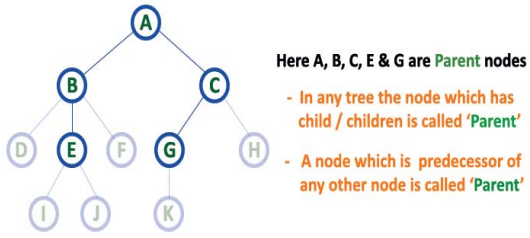


Figure 2.18.: Parent representation

d. Child

In a tree data structure, the node which is descendant of any node is called a **CHILD Node**. In a tree, any parent node can have any number of child nodes. In a tree, all the nodes except root are child nodes.

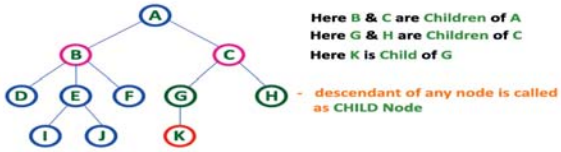


Figure 2.19.: Child representation

e. Siblings

In a tree data structure, nodes which belong to same Parent are called **SIBLINGS**.

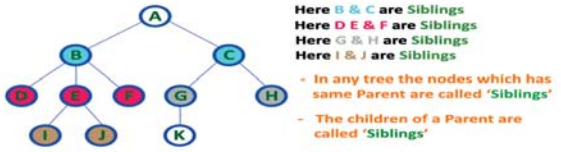


Figure 2.20.siblings

f. Leaf

In a tree data structure, the node which does not have a child is called a LEAF Node. In a tree data structure, the leaf nodes are also called a External Nodes or Terminal node.

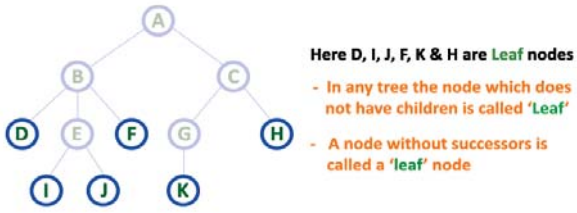


Figure 2.21: Leaf representation

g. Internal Nodes

In a tree data structure, the node which has at least one child is called **INTERNAL Node**. In a tree data structure, nodes other than leaf nodes are called **Internal Nodes**. **The root node is also said to be Internal Node** if the tree has more than one node. Internal nodes are also called as ‹**Non-Terminal**' nodes.

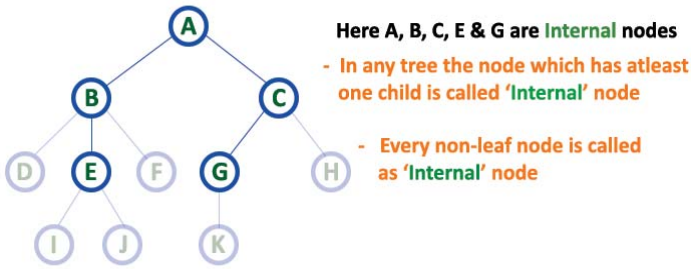


Figure 2.22: Internal node representation

h. Degree

In a tree data structure, the total number of children of a node is called as **DEGREE** of that Node. In simple words, the Degree of a node is total number of children it has. The highest degree of a node among all the nodes in a tree is called as ‹**Degree of Tree**'

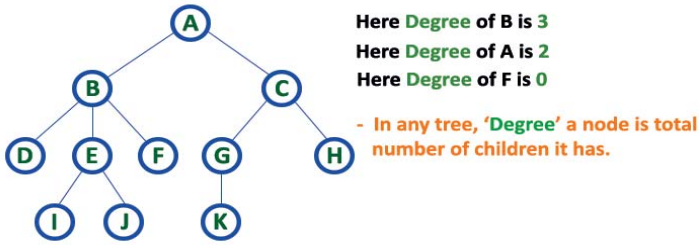


Figure 2.23: Degree representation

i. Level

In a tree data structure, the root node is said to be at Level 0 and the children of root node are at Level 1 and the children of the nodes which are at Level 1 will be at Level 2 and so on... In simple words, in a tree each step from top to bottom is called as a Level and the Level count starts with '0' and incremented by one at each level (Step).

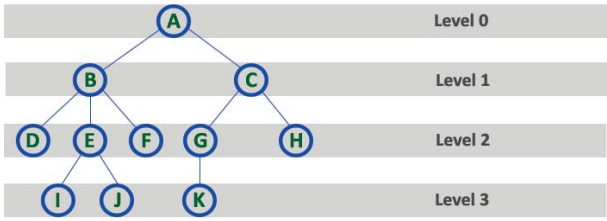


Figure 2.24: level representation

j. Height

In a tree data structure, the total number of edges from leaf node to a particular node in the longest path is called as **HEIGHT** of that Node. In a tree, height of the root

node is said to be **height of the tree**. In a tree, **height of all leaf nodes is '0'.**

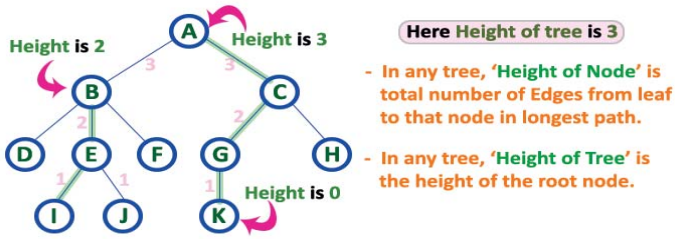


Figure 2.25.: Height representation

k. Depth

In a tree data structure, the total number of edges from root node to a particular node is called **DEPTH** of that Node. In a tree, the total number of edges from root node to a leaf node in the longest path is said to be **Depth of the tree**. In a tree, **depth of the root node is '0'.**

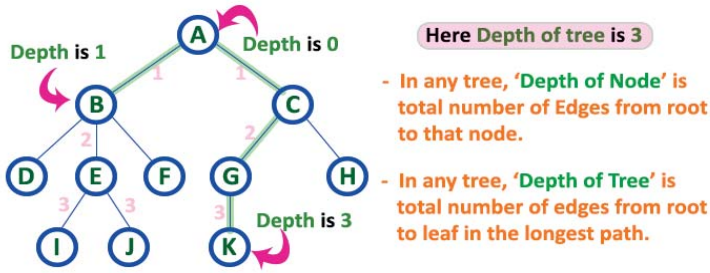


Figure 2.26.: Depth representation
Source: http://mitpolytechnic.ac.in/downlaods/09_knowledge-bank/02_computer/SEM-3/DS/Chap5_6.pdf

l. Path

In a tree data structure, the sequence of Nodes and Edges from one node to another node is called as **PATH** between those two Nodes. **Length of a Path** is total number of nodes in that path. In below example **the path A - B - E - J has length 4**.

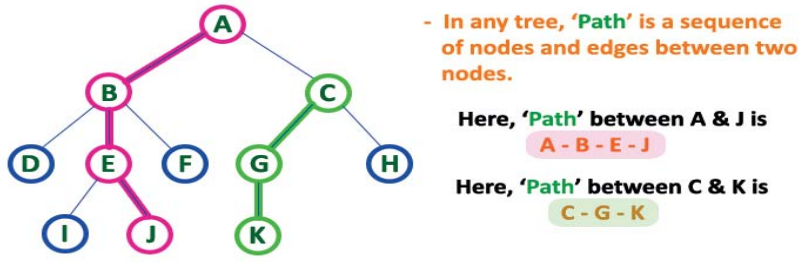


Figure 2.27: path representation

m. Sub Tree

In a tree data structure, each child from a node forms a sub tree recursively. Every child node will form a sub-tree on its parent node.

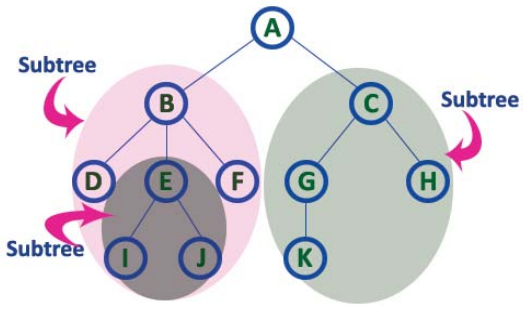


Figure 2.28: Sub tree representation

2.7.2 Binary Tree Representations

A binary tree data structure is represented using two methods. Those methods are as follows:

- Linked List of Binary Tree representation
- Array Representation

Let **T** be a Binary Tree. There are two ways of representing **T** in the memory as follow Linked

a. Linked List of Binary Tree representation

Consider a Binary Tree **T**. **T** will be maintained in memory by means of a linked list representation, which uses three parallel arrays; **INFO**, **LEFT**, and **RIGHT** pointer variable **ROOT** as follows. In Binary Tree, each node **N** of **T** will correspond to a location **k** such that:

1. **LEFT [k]** contains the location of the left child of node **N**.
2. **INFO [k]** contains the data at the node **N**.
3. **RIGHT [k]** contains the location of right child of node **N**.

- **Representation of a node:**

LEFT [k]	INFO [k]	RIGHT [k]

In this representation of binary tree, root will contain the location of the root **R** of **T**. If any one of the sub tree is empty, then the corresponding pointer will contain the null value if the tree **T** itself is empty, the **ROOT** will contain the null value.

Example

Consider the binary tree **T** in the figure of **Binary Tree**

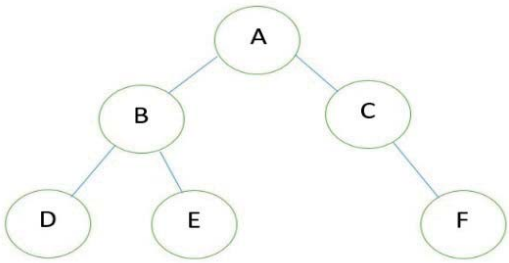


Figure 2.29: Binary Tree

Linked Representation of the Binary Tree

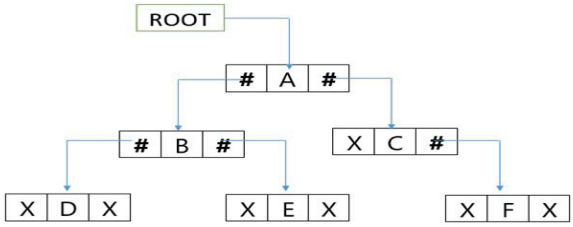


Figure 2.30.: Linked Representation of the Binary Tree

b. Array Representation of Binary Tree

Let us consider that we have a tree **T**. let our tree **T** is a binary tree that us complete binary tree. Then there is an efficient way of representing **T** in the memory called the sequential representation or array representation of **T**. This representation uses only a linear array TREE as follows:

1. The root **N** of **T** is stored in **TREE [1]**.
2. If a node occupies **TREE [k]** then its left child is stored in **TREE [2 * k]** and its right child is stored into **TREE [2 * k + 1]**.

For Example:

Consider the following Tree:

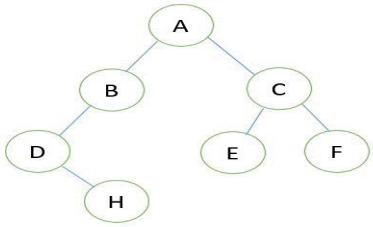


Figure 2.31: Array Representation of Binary Tree

Its sequential representation is as follow:

A	B	C	D	-	E	F	-	H		

2.7.3 Binary Tree Traversals

When we wanted to display a binary tree, we need to follow some order in which all the nodes of that binary tree must be displayed. In any binary tree displaying order of nodes depends on the traversal method.

Displaying (or) visiting order of nodes in a binary tree is called as Binary Tree Traversal.

There are three types of binary tree traversals.

- In - Order Traversal
- Pre - Order Traversal
- Post - Order Traversal

Traversal is a process to visit all the nodes of a tree and may print their values too. Because, all nodes are connected via edges (links) we always start from the root (head) node. That is, we cannot randomly access a node in a tree.

Generally, we traverse a tree to search or locate a given item or key in the tree or to print all the values it contains.

a. In-order Traversal

In this traversal method, the left sub tree is visited first, then the root and later the right sub-tree. We should always remember that every node may represent a sub tree itself.

If a binary tree is traversed **in-order**, the output will produce sorted key values in an ascending order.

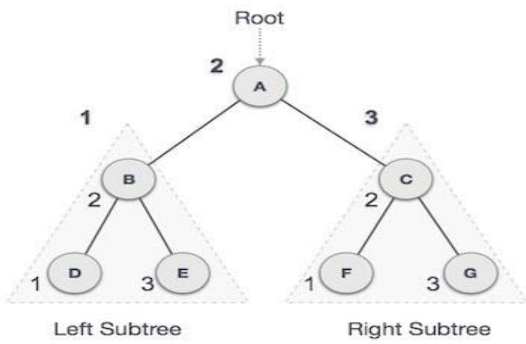


Figure 2.32: In-order Traversal

We start from **A**, and following in-order traversal, we move to its left subtree **B**. **B** is also traversed in-order. The process goes on until all the nodes are visited. The output of inorder traversal of this tree will be −

D → B → E → A → F → C → G

Algorithm:

Until all nodes are traversed –

Step 1 – Recursively traverse left subtree.

Step 2 – Visit root node.

Step 3 – Recursively traverse right subtree.

b. Pre-order Traversal

In this traversal method, the root node is visited first, then the left subtree and finally the right subtree.

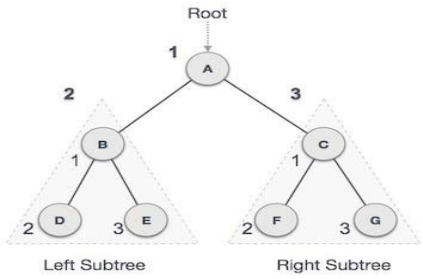


Figure 2.33: Pre-order Traversal

We start from **A**, and following pre-order traversal, we first visit **A** itself and then move to its left subtree **B**. **B** is also traversed pre-order. The process goes on until all the nodes are visited. The output of pre-order traversal of this tree will be −

A → B → D → E → C → F → G

Algorithm:

Until all nodes are traversed –

Step 1 – Visit root node.

Step 2 – Recursively traverse left subtree.

Step 3 – Recursively traverse right subtree.

c. Post-order Traversal

In this traversal method, the root node is visited last, hence the name. First we traverse the left sub tree, then the right sub tree and finally the root node.

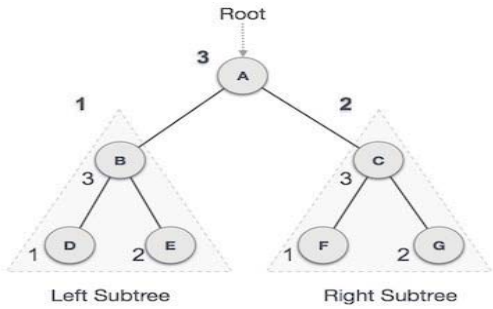


Figure 2.34.: Post-order Traversal

We start from **A**, and following Post-order traversal, we first visit the left sub tree **B**. **B** is also traversed post-order. The process goes on until all the nodes are visited. The output of post-order traversal of this tree will be −

D → E → B → F → G → C → A

Algorithm:

Until all nodes are traversed –

Step 1 – Recursively traverse left subtree.

Step 2 – Recursively traverse right subtree.

Step 3 – Visit root node.

Binary Search tree exhibits a special behavior. A node's left child must have a value less than its parent's value and the node's right child must have a value greater than its parent value.

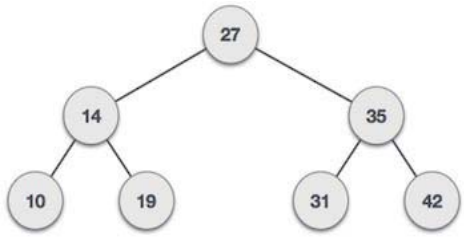


Figure 2.35: Tree Example

2.7.4. Basic Operations on binary tree

A Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties

- The left sub-tree of a node has a key less than or equal to its parent node's key.
- The right sub-tree of a node has a key greater than to its parent node's key.

Thus, BST divides all its sub-trees into two segments; the left sub-tree and the right

sub-tree and can be defined as

left_subtree (keys) ≤ node (key) ≤ right_subtree (keys)

Representation

BST is a collection of nodes arranged in a way where they maintain BST properties. Each node has a key and an associated value. While searching, the desired key is compared to the keys in BST and if found, the associated value is retrieved.

Following is a pictorial representation of BST

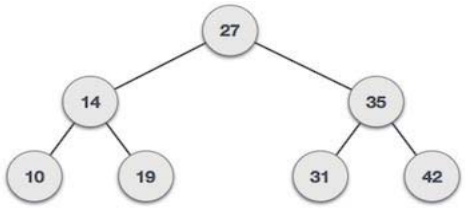


Figure 2.36.: ***Binary Search Tree***

We observe that the root node key (27) has all less-valued keys on the left sub-tree and the higher valued keys on the right sub-tree.

Basic Operations

Following are the basic operations of a tree –

- Search – Searches an element in a tree.
- Insert – Inserts an element in a tree.
- Pre-order Traversal – Traverses a tree in a pre-order manner.
- In-order Traversal – Traverses a tree in an in-order manner.
- Post-order Traversal – Traverses a tree in a post-order manner.

Node

Define a node having some data, references to its left and right child nodes.

```
struct node {
int data
struct node leftChild
struct node rightChild
};
```

Search Operation

Whenever an element is to be searched, start searching from the root node. Then if the data is less than the key value, search for the element in the left subtree. Otherwise, search for the element in the right subtree. Follow the same algorithm for each node.

Algorithm

```
struct node search(int data){
struct node current = root
  print "Visiting elements: "
while(current->data != data){
if(current != NULL){
      print current->data
//go to left tree
if(current->data > data){
        current = current->leftChild
//else go to right tree
else
        current = current->rightChild
end if//not found
if(current == NULL)
return NULL
      }
}
}
return current
}
```

Insert Operation

Whenever an element is to be inserted, first locate its proper location. Start searching from the root node, then if the data is less than the key value, search for the empty location in the left subtree and insert the data. Otherwise, search for the empty location in the right subtree and insert the data.

Algorithm

```
void insert(int data){
struct node tempNode =(struct node)
struct node current
struct node parent
```

```
  tempNode->data = data
  tempNode->leftChild = NULL
  tempNode->rightChild = NULL //if tree is empty
if(root == NULL){
    root = tempNode
}else{
    current = root
    parent = NULL
while(1){
      parent = current
//go to left of the tree
if(data < parent->data){
        current = current->leftChild
//insert to the left
if(current == NULL){
          parent->leftChild = tempNode
return
}
}//go to right of the tree
else{
        current = current->rightChild
//insert to the right
if(current == NULL){
          parent->rightChild = tempNode
return
}}}}}
```

2.8 Searching and sorting using complex data structure.

LEARNING ACTIVITY 2.4

Array A[10] is declared and initialized as: int A[10] = {10, 8, 2, 7, 3, 4, 9, 1, 6, 5};

a. Write down the steps to do the followings:

b. Search the value "7"

Definition

Searching is the process of finding the location of a given element in a set of elements.

Searching means to find whether a particular value is present in an array or not. If the value is present in the array, then searching is said to be successful and the searching process gives the location of that value in the array.

There are two simple approaches to searching:

Linear search: This method traverses a list sequentially to locate the search key.

Binary search: This method works on sorted lists by progressively making better guesses to find the location of a search key.

2.8.1 Linear search Algorithms

The linear search is also called sequential search. It is a simple method used for searching an array for a particular value. It works by comparing the value to be searched with every element of the array one by one in a sequence until a match is found.

Notice that the linear search is mostly used to search in an unordered list of elements

Linear Search Algorithm

linear search (a, n, item, loc)

Note:

- "a" is an array of the size n.
- "n" is the size of the array "a"
- "item" is the element to find in the array "a".
- "loc" is the index of an element in the array "a".

Begin
for i=0 to (n-1) by 1 do
if (a[i] = item) then
set loc=i
Print loc
exit
endif
setloc +1
endfor
end

2.8.2 Binary Search

Learning activity 2.5

Consider an array A[11] that is declared and initialized as:

int A[11] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

The value to be searched is VAL= 9

Analyze the 1st step and do the remaining one.

The algorithm proceeds in the following manner:

- BEG = 0, END = 10, MID = (0 + 10)/2 = 5

  Now, compare VAL = 9 and A[MID] = A[5] = 5
- A [5] is less than VAL. Therefore, we search for the value in the second half of the array. So, we assign the value of A[MID] to BEG.

- **Binary Search**

You search an array by repeatedly dividing the search interval in half. While dividing you begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty. This kind of search is also called the dichotomy method or bisection method.

Schematically, it looks like the following:

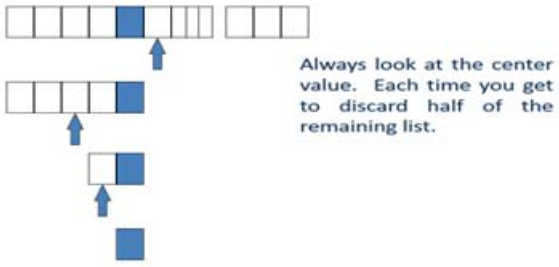


Figure 2.37: Process of Binary search or dichotomy method

- **The full binary search Algorithm is given down here.**

Binary search (a, n, item, loc)

Note:

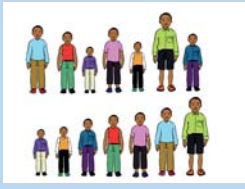
- "a " is an array of the size n.
- "n" is the size of the array "a"
- " item " is the element to find in the array "a".
- "loc" is the index of the element in the array "a".

```
Begin
set beg=0
set end=n-1
set mid=(beg+end)/2
while((beg<=end) and(a[mid]!=item) do
if(item<a[mid]) then
set end=mid-1
else
set beg=mid+1
endif
set mid=(beg+end)/2
endwhile
if(beg>end) then
set loc=-1
else
set loc=mid
endif
end
```

2.9 Sorting using complex data structure

Learning activity 2.5

Analyze the pictures below and answer the followings questions.



1. Describe the arrangement of these people in their rows
2. Compare the two rows of person

2.9.1. Definition:

Sorting is the process of arranging elements in some logical order either in ascending or descending order.

Methods for sorting

There are various methods for sorting in this section that we are going to discuss: Bubble sort, Insertion sort and Selection sort.

2.9.2. Bubble sort

Bubble Sort is a simple-minded algorithm based on the idea that we look at the list, and wherever we find two consecutive elements out of order, we swap them. This is done as follows: Repeatedly traverse the unsorted part of the array by comparing consecutive elements, and interchange them when they are out of order.

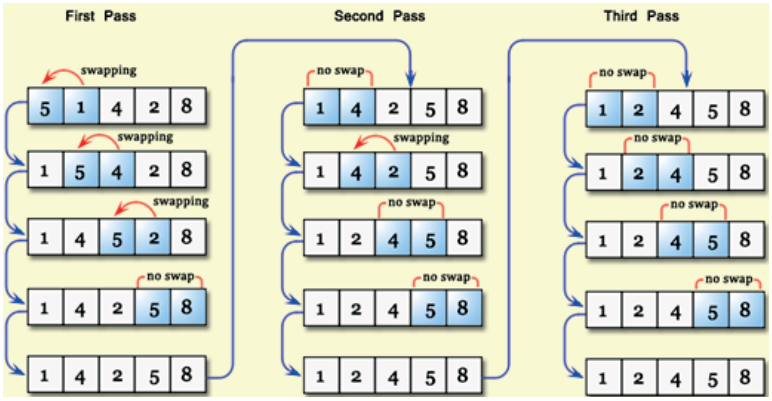


Figure 2.38.: Process of \|Bubble Sort

Source: https://www.jianshu.com/p/5bbfc8a191f5

Bubble Sort **Algorithm**

Begin
for k=1 to (n-1) by 1 do
for j=0 to (n-k-1) by 1 do
if(a[j]>a[j+1]) then
set temp=a[j]
set a[j]=a[j+1]
set a[j]=temp
endif
endfor
endfor
end

2.9.3. Insertion sort

The Insertion Sort is a comparison-based algorithm that builds a final sorted array one element at a time. It iterates through an input array and removes one element per iteration, finds the place if the element belongs in the array, and then places it there.

The illustration is given here below:

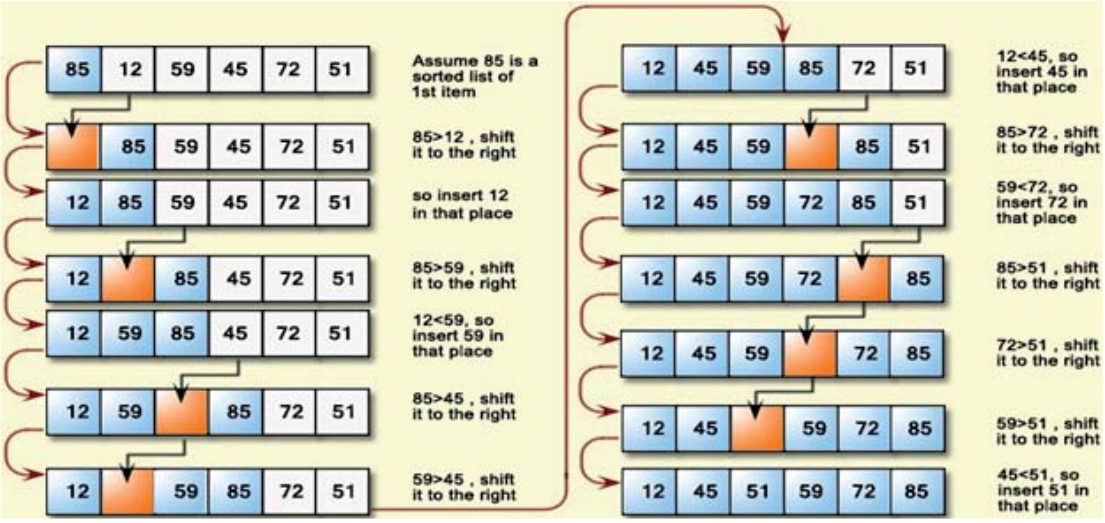


Figure 2.39.: process of insertion sort

Source: https://www.jianshu.com/p/5bbfc8a191f5

The Insertion Sort traverses the array and inserts each element into the sorted part of the list where it belongs. It involves pushing down the larger elements in the sorted part.

Insertion Sort Algorithm

Begin
For I = 1 to N-1
J = I
Do while (J > 0) and (A(J) < A(J - 1)
Temp = A(J)
A(J) = A(J - 1)
A(J - 1) = Temp
J = J - 1
End-Do
End-For
End

2.9.4. Selection sort

The Selection Sort is a simplicity sorting algorithm. Here are basic steps of selection sort algorithm:

i. Find the minimum element in the list
ii. Swap it with the element in the first position of the list
iii. Repeat the steps above for all remainder elements of the list starting at the second position.

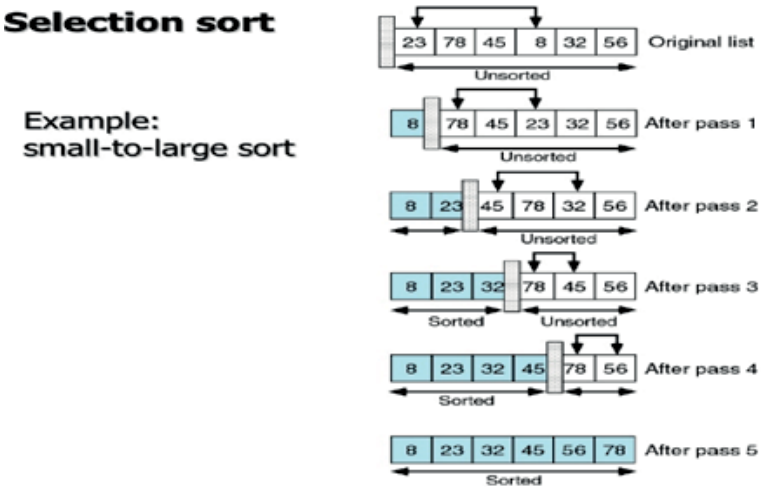


Figure 2.40.: process of selection sort

The idea of Selection Sort is that we repeatedly find the smallest element in the unsorted part of the array and swap it with the first element in the unsorted part of the array.

Selection Sort Algorithm

Begin
For I = 0 to N-1 do:
Smallsub = I
For J = I + 1 to N-1 do:
A(J) < A(Smallsub)
Smallsub = J
End-If
End-For
Temp = A(I)
A(I) = A(Smallsub)
A(Smallsub) = Temp
End-For
End

Application activity 2.4.

1. By using the bubble sort algorithm, write a program to sort an integer array of 10 elements in ascending order.
2. By using the sequential search algorithm, write an algorithm program to search for an element of an integer array of 10 elements.
3. Write a program to find the maximum and minimum value of an array.

2.10 Apply algorithm to solve complex mathematical functions

2.10.1. Understanding quadratic equation

LEARNING ACTIVITY 2.6.

Below you are given the algorithm and pseudo code to resolve quadratic equation: ax2+ b x+ c=0; where a, b and c are not equal to zero. Improve the algorithm and pseudo code to resolve quadratic equation since some of steps to resolve a quadratic equation are not given.

Algorithm:

- Start.
- Declare the required variables.
- Indicate the user to enter the coefficients of the quadratic equation
- Enter the input.
- Calculate the roots of quadratic equation using the proper formulae.
- Display the result.
- Stop. Pseudo Code
- Start.
- Input a, b, c.
- D — sqrt (b × b – 4 × a × c).
- X1 — (-b + d) / (2 × a).
- X2 — (-b - d) / (2 × a).
- Print x1, x2.
- Stop.

Quadratic equation is any mathematical equation having the form ax2+bx+c=0 where x represents an unknown, a, b, and c are constants.

The quadratic equation is a second degree polynomial equation since it only contains powers of x that are non-negative integers.

APPLICATION ACTIVITY 2.5

Draw the flow chart of the algorithm to solve quadratic equation.

2.10.2. Recursive function

Definition: A recursive function is a function that calls itself during its execution.

The Recursive function repeats itself several times, outputting the result at the end of each iteration. In this section we are going to discuss the following recursive functions:

- Summing a list of numbers;
- Factorial function;
- Fibonacci series function;
- Sorting;
- Palindrome.

- **Summing a list of numbers**

To calculate the sum of a list of numbers such as: [1, 3, 5, 7, 9], below is the pseudo code which is an iterative function that computes the sum of a list of numbers.

```
deflistsum(numList):
theSum = 0
fori in numList:
theSum = theSum + i
returntheSum
print(listsum([1,3,5,7,9]))
```

The function uses an accumulator variable (thesum) to compute a running total of all the numbers in the list by starting with 00 and adding each number in the list

APPLICATION ACTIVITY 2.6.

1. Analyze the below pseudo codes and give the general formula of the function below.

   If n = 5 sum(n) = 1+2 +3 +4 +5 =15

   If n=6 sum(n) = 1 +2 +3 +4 +5 +6= 21

2. Write the algorithm and draw the flow chart to calculate the sum of numbers in a list.

2.10.3 Factorial function

LEARNING ACTIVITY2.8

Analyze the pseudo code below and give the general formula.

```
double factorial (int n )
{
double  fac1;
fac1 = 1;
for (inti =1; i<= n ; i ++)
fac1 =fac1*i;
return ( fac1);
}
```

In mathematics, the factorial of a non-negative integer n, denoted by n! is the product of all positive integers less than or equal to n.

Notice that the value of 0! is 1, according to the convention for an empty product.

The factorial function (symbol: !) means to multiply a series of descending natural numbers. Examples:

$4! = 4 \times 3 \times 2 \times 1 = 24$

$7! = 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 5040$

$1! = 1$

APPLICATION ACTIVITY 2.7

1. Analyse the pseudo codes below and answer following questions:

Pseudo Code A

```
double factorial( int n )
{
double  fac1;
fac1 = 1 ;
for ( inti =1; i<= n ; i ++)
fac1 =fac1 *i;
return ( fac1);
 }
```

Pseudo Code B

```
double factorial( int n )
{
 if ( n ==1 ) // base case
 return 1;
  return n * factorial( n  - 1 );
}
```

i. Predict the result of each pseudo code
ii. What do you think of each concept used?
iii. Are they representing the same algorithm? If yes, which one is not time consuming and space consuming? Explain why.

2. Draw the flow chart to calculate factorial number.

2.10.4 Fibonacci series function

LEARNING ACTIVITY 2.9

F (0) = 0

F (1) = 1

F (2) = F (2-1) + F (2-2)

= F (1) + F (0) = 1 + 0 = 2

1. Compute F (3), F (4), F (5),
2. Analyze the above pseudo codes and deduct the general formula

In mathematics, the Fibonacci numbers are integer numbers in the following sequence: 0, 1, 1, 2, 3, 5, 8..., this series of integers is called the Fibonacci sequence. Fibonacci sequence is characterized by the fact that every number after the first two is the sum of the two preceding one. It means that except for the first two terms of the sequence every other term is the sum of the previous two terms.

The pseudo code of Fibonacci function

```
Fibo(n)
Begin
if n <= 1 then
Return n;
else
Return Call Fibo(n-1) + Call Fibo(n-2);
endif
End
```

APPLICATION ACTIVITY 2.8..

Write the algorithm and draw the flow chart to calculate Fibonacci Series.

2.10.5. Palindrome function

LEARNING ACTIVITY 2.10..

Investigate and find at least 5 Kinyarwanda's words which do not change when its letters are reversed.

INSTRUCTIONS:

- Extract the letters from the word.
- Reverse their order.
- Paste them back together into a word.
- Compare with the original word.

The algorithm which would get out any amount palindrome numbers on screen.

Let take: 300003 310013 320023 330033 340043 350053.

Findings: written algorithm how to check if it is a palindrome number or not.

Here is algorithm for how i check if its palindrome or not:

```
BEGIN
   int n, num, dig, rev = 0
   PRINT "Insert number"
   READ num
   n = num
  while (num != 0)
   {
      dig = num % 10
      rev = (rev * 10) + dig
      num = num / 10
   }
   if (n == rev)
   PRINT "This is palindrome", rev
   else
   PRINT "This is not palindrome " rev
    END IF
END
```

A palindrome is a word that is spelled the same from both ends.

For example: anna, ifi, isi, and malayalam are palindromes.

The table below gives two basic ways of determining whether a word is a palindrome.

Pseudo code:	Detailed Algorithm:
Input a String Initialize Len to zero , Flag to zero While String[Len] is not equal to NULL Increment Len Initialize I to zero , J to Len-1 While I is less than (Len/2)+1 If String[I] equal to String[J] Flag=0 else Flag=1 Increment I , Decrement J If Flag equal to zero Print Key Is a Palindrome else Print Key Is Not a Palindrome Stop	Step 1: Input S (string) Step 2: Len = 0 , Flag =0 Step 3: While (S[Len] != NULL) Len++ Step 4: I = 0 , J = Len-1 Step 5: While (I < (Len/2)+1) If (S[I] == S[J]) Flag=0 else Flag=1 I++ , J– Step 6: If (Flag == 0) Print Key Is a Palindrome else Print Key Is Not a Palindrome Step 7: End

APPLICATION ACTIVITY 2.9..

From the above pseudo code and detailed algorithm draw the flowchart that check whether a given string is a palindrome or not

2.10.6. Euclid's GCD(Greatest common denominator)

LEARNING ACTIVITY 2.11.

Given int a=210 and int b=45;

Find the greatest common divisor of two positive integers a and b.

Write down all steps to find the greatest common denominator.

Suggest the algorithm to find the greatest common denominator of two numbers.

Compare with the original word.

LEARNING ACTIVITY 2.12.

Analyse the following pseudo code and answer to the question below.

1. Start 2. read two numbers a,b 3. set n=b,d=a 4. r=a%b 5. while r!=0 5.1 set n=d,d=r 5.2 r=n%d 6. setgsc=d 7. lcm=(a*b)/gcd 8. displaygcd and lcm 9. stop	begin Let a, b denote the numerator and denominator, respectively. Let 'gcd' denote the Greatest Common Divisor of both a and b. *If (a==0 || b==0)* *return gcd = 1;* *else* *{* *r = (a%b);* *while (r < 0 || r > 0)* *{* *a = b;* *b = r;* *r = (a%b);* *}* *gcd = b;* *}* *return gcd;* *end*

c. What the 2 pseudocodes are giving as results?

d. Replace a by 24 and b by 16. What is the results of the two pseudo codes?

- **Greatest Common Denominator**

The greatest common denominator, or GCD, between two numbers is the greatest integer that divides both given numbers.

- **Euclidean Algorithm**

The Euclidean Algorithm is pretty straight forward: given two numbers, repeatedly replace the larger number with the greater number mod the lesser number. Keep repeating the step until one of the two numbers reaches zero. The other number will then be the greatest common denominator (or greatest common divisor), the GCD.

1. Divide m by n and let r be the remainder.
2. If r is 0, n is the answer; if r is not 0, continue to step 3.
3. Set m = n and n = r. Go back to step 1.

2.11 END UNIT ASSESSMENT

1. Examine the following:

```
Double[][] values =
 {{1.2, 9.0, 3.2},
  {9.2, 0.5, 1.5, -1.2},
  {7.3, 7.9, 4.8}} ;
```

what is in values[2][1] ?

2. You want to create a table that looks like:

12	-9	8
7	14	
-32	-1	0

3. Which of the following will work

A

```
double[][]  table =
  { 12, -9, 8,
    7, 14,
    -32, -1, 0} ;
```

```
double[][] table =
  { {12, -9, 8}
    {7, 14}
    {-32, -1, 0} };
```

B

```
double[][] table =
  { {12, -9, 8},
    {7, 14, 0},
    -32, -1, 0} };
```

```
double[][] table =
  { {12, -9, 8},
    {7, 14},
    {-32, -1, 0} };
```

C D

4. Write an algorithms that allows a user to enter 4 elements for a 2x2 array and print their product.
5. A linear list of elements in which deletion can be done from one end (front) and insertion can take place only at the other end (rear) is known as a ?
   a. Queue
   b. Stack
   c. Tree
   d. Linked list

6. Write a Fibonacci series algorithm of any given number entered through the keyboard
7. Write algorithm where the user is asked to input a number and print it's factorial
8. For each of the algorithm designed in the different activities, write their corresponding programmes in C++ languages and verify the results.

UNIT 3

INTRODUCTION TO COMPUTER NETWORK

UNIT 3: INTRODUCTION TO COMPUTER NETWORK

Key Unit Competency:

To be able to explain principles, standards and purposes of computer network

Read the following scenario and answer to the asked questions.

In the school computer labs, choose a classmate and send him/her an email that has "Computer Network "as the email subject and the email body contains names of previous units already covered in senior 5 computer science. Ask him/her to reply immediately to your email.

After receiving the reply, answer the following questions:

1. List all possible elements that make this communication possible
2. What devices are used to facilitate sending email successfully?
3. Discuss what made possible the sending and receiving of emails
4. Do you like using email?
5. How do you value the factor of sending and receiving an email
6. Describe what you see in the photo below?

3.1. Fundamental of computer Networ

3.1.1. Computer Network Definition

ACTIVITY 3.1

Observe computers in the computer lab and answer the following question?

1. Describe what you see?
2. What are transmission media used to connect computers in the computer labs?
3. Are all devices connected?

A computer network is a system of interconnected computers for the purpose of sharing data and resources.

Computers on a network are called nodes. Nodes can include hosts such as personal computers, phones, servers as well as networking hardware. Two devices are networked together when one device is able to exchange information with the other device. The computer can be connected to another in the two ways.

Wired network: Computers are connected using cable media. Most commonly Ethernet Cable, coaxial cable and optic fiber.

Wireless network: Computers are connected using wireless media. Radio waves are used in wireless mode.

3.1.2 Properties of Computer Network

- **Easy Sharing of Resources:** Computers are able to share various resources easily over a network. Shared resources can be Internet, files, printer, storage and others.
- **Performance:** It is achieved by measuring the speed of data transmission with number of network users, connectivity used and the software used. The commonly measured qualities in the network performance are **Bandwidth and Latency.**
- **Reliability:** It means that computer network provides assurance of the delivery of data to the intended recipient.
- **Scalability:** The possibility of adding new computer without affecting the network performance.
- **Security:** computer network must be secured for the benefit of the user and data protection. The security is achieved by protecting data from unauthorized access.

- **Quality of Service (QoS):** Quality of Service refers to the mechanism that **manage congested network traffic.**
- **Fault tolerant:** A fault tolerant network limits the impact of hardware or software failure and recovers quickly when a failure occurs

APPLICATION ACTIVITY 3.2

1. Investigate what happen when a shared resource is disconnected from the computer network?
2. Imagine other areas where computer network is used?

3.1.3 Advantages and Disadvantage of Computer Networking

The common advantages of computer network

Activity 3.2

1. Discuss the disadvantages of using computer network within your school?
2. In class of senior 5 MPC, Computer Science Teacher decides to share notes for students to use on our school network. The mathematics teacher decides to print notes of 100 pages and distributes to each member of senior 5 MPC class.
   i. Evaluate the cost impact for both teachers
   ii. Both teachers, whom save money in giving notes.

The common advantages of computer network

1. **Enhanced communication and availability of information:** It allows access to a vast amount of useful information.

   Example: Population data, newsletters, online businesses, contents, Applications.

2. **Allow resource sharing:** Fewer resources are needed when an organization uses a computer network.

   Example: only one printer is needed instead of buying a printer for each office.

3. **File sharing made easy:** computer network allows people to share files, which helps to save more time and effort.

   Example: A teacher can share homework to all students through school network

4. **Improve storage capacity:** since you are going to share information, files and resources to other people, you have to ensure that all data and content are properly stored in the system.

5. **Cost efficiency:** on computer network, you can share software license installed on the server and can then be used by various workstations.
6. **Security of information and resources:** users cannot see other users' files unlike on stand-alone machines.
7. **Backup of data is easy** as all the data is stored on the file server.

The common disadvantages of computer networking

1. **Lack of independence:** people rely on computer network and when the system is down, people get stack. Most of organizations depend on the computer networks.
2. **Security issues:** huge number of people use a computer network to get and share their files and resources, a certain user's security would be always at risk. Viruses can spread to other computers throughout a computer network. There is a danger of hacking, particularly with wide area networks. Security procedures are needed to prevent such abuse, Examples: The use of Antivirus and firewall.
3. **Lack of robustness:** computer network's main server breaks down, the entire system would become useless

APPLICATION ACTIVITY 3.2

1. Investigate what happen when a shared resource is disconnected from the computer network?
2. Imagine other areas where computer network is used?

3.1.4 Types of computer networks

Activity 3.3

With two or more Bluetooth enabled devices available in the classroom:

1. Turn on the Bluetooth feature on both devices
2. Share and receive files such images and documents between the devices
3. Turn off Bluetooth feature in the two devices.

A computer network is classified basing on the size of the area covered, the number of users connected and the number & types of services available.

The followings are type of computer networks:

1. Personal Area Network (PAN)

A personal area network (PAN) is a network that connects devices, such as mice, keyboards, printers, Smartphone, and tablets within the range of an individual person. PAN has connectivity range up to 10 meters. PAN may include wireless computer keyboard and mouse, Bluetooth enabled headphones, wireless printers and TV remotes. All of these devices are dedicated to a single host and are most often connected with Bluetooth technology.

Bluetooth is a wireless technology that enables devices to communicate over short distances. A Bluetooth device can connect up to seven other Bluetooth devices.

2. Local Area Network (LAN)

Traditionally, a LAN is defined as a network that covers a small geographical area. However, the distinguishing characteristic for LANs today is that they are typically owned by an individual, such as in a home or small business, or wholly managed by an IT department, such as in a school or corporation. This individual or group enforces the security and access control policies of the network. LAN uses Ethernet IEEE 802.3 as its standard.

A Wireless LAN (WLAN) is a LAN that uses radio waves to transmit data between wireless devices. In a traditional LAN, devices are connected together using copper cabling. In some environments, installing copper cabling might not be practical, desirable, or even possible. In these situations, wireless devices are used to transmit and receive data using radio waves. As with LANs, on a WLAN, you can share resources, such as files, printers, and Internet access. WLAN uses Ethernet IEEE 802.11 as its standard.

3. Metropolitan Area Network (MAN)

A metropolitan area network (MAN) is a network that spans across a large organization like campus or a city. The network consists of various buildings interconnected through wireless or fiber optic backbones.

A backbone is the part of the computer network infrastructure that interconnects different LAN networks and provides a path for exchange of data between these different networks

The communication links and equipment are typically owned by a network service provider who sells the service to the users. A MAN can acts as a high-speed network to enable sharing of regional resources.

4. Wide Area Network (WAN)

A WAN connects multiple networks that are in geographically separated locations.

The distinguishing characteristic of a WAN is that it is owned by a service provider. Individuals and organizations contract for WAN services. WAN network provides connectivity to MANs and LANs. The most common example of a WAN is the Internetwork or Internet in short. The Internet is a large WAN that is composed of millions of interconnected networks.

Example: Kigali and Nairobi networks are connected through the Internet

APPLICATION ACTIVITY 3.3

1. Discuss the role ISP services for LANs?
2. Do an observation in the school computer lab and categorize the computer network type available at your school? Justify your answer.
3. Using clear examples, Distinguish Personal Area Network and Local Area Network?
4. Using an arrow match the following in Group A with their corresponding in Group B

Group A	Group B
a. Radio waves	1. Qualities of Computer network Performance
b. Millions of interconnected network	2. Disadvantages of Computer network
c. Sharing resources	3. Wireless
d. Hacking attacks	4. WAN
e. Bandwidth and Latency	5. Advantages of Computer network

3.2 Computer Network Concept and Technology

ACTIVITY 3.4

You are browsing a website on the internet while the speed is okay and suddenly you find that the page is loading slowly. Investigate the reason behind slowness?

3.2.1 Network metrics

Network metrics are defined as standards of measurement by which efficiency, performance, progress, or quality of a plan, process can be assessed. Network metric are used by a router to make routing decisions.

When data is sent over a computer network, it is broken up into small chunks called **packets.** Each packet contains source and destination address information. Packets are sent across a network one bit at a time.

a. Bandwidth

Bandwidth is the amount of data that can be transmitted in a fixed amount of time. Bandwidth is measured in the number of bits that can be sent every second.

The following are examples of bandwidth measurements:

- b/s - bits per second
- kb/s - kilobits per second
- Mb/s - megabits per second
- Gb/s - gigabits per second

b. Latency

Latency is the time between requesting data and receiving data. More simply put, the time it takes to establish a connection between your computer and the server hosting the website you requested. The important thing to take away here is that latency is not speed. Data is delayed by network devices and cable length. Network devices add latency when processing and forwarding data.

c. Throughput:

Throughput is the actual rate at which information is transferred. It measures the amount of completed work against time consumed and may be used to measure the performance of a processor, memory and/or network communications.

d. Error rate:

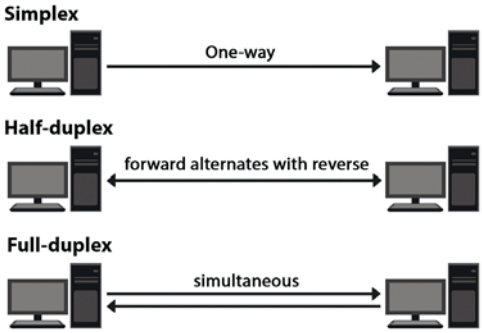
Error rate is the number of corrupted bits expressed as percentage or fraction of the total sent.

e. Jitter:

Jitter is variation in packet delay at the receiver of information. For measuring the capacity of all of these metrics above, we focus on capacity of message and this one change into different form depending on the network devices where this message is located.

3.2.2 Data Transmission model

The term transmission mode is used to define the direction of signal flow between two linked devices. The data that is transmitted over the network can flow using one of three modes: simplex, half-duplex and full-duplex.



Picture 3.1. Transmission modes: simplex, half-duplex and full-duplex

1. **Simplex:** it is a single one-way transmission. In a **simplex transmission mode**, the communication between sender and receiver occurs only in one direction. That means only the sender can transmit data, and receiver can receive that data. The receiver cannot transmit any information back to the sender.

   The keyboard to monitor transmission is an example of simplex transmission mode. Other examples are communication between a computer and a printer, listening to the radio and the signal that is sent from a TV station to your home TV.
2. **Half-Duplex:** data flows in one direction at a time. In half-duplex, the channel of communications allows alternating transmission in two directions, but not in both directions simultaneously. An Example of Half-duplex is the Talkie- Walkie used by the police.
3. **Full-Duplex:** data flows in both directions at the same time. In a **full duplex transmission mode**, the communication between sender and receiver can occur simultaneously. Sender and receiver both can transmit and receive simultaneously at the same time.

Basis for Comparison	Simplex	Half Duplex	Full Duplex
Direction of Communication	Communication is unidirectional.	Communication is two-directional but, one at a time.	Communication is two directional and done simultaneously.
Send/Receive	A sender can send data but, cannot receive.	A sender can send as well as receive the data but one at a time.	A sender can send as well as receive the data simultaneously.

Performance	The half-duplex and full duplex produces better performance than the Simplex.	The full duplex mode produces higher performance than half duplex.	Full duplex has better performance as it doubles the utilization of bandwidth.
Example	Keyboard and monitor.	Walkie-Talkies, internet chart	Telephone.

Table 3.1: comparison of network transmission modes

A telephone conversation is an example of full-duplex communication. Both people can talk and be heard at the same time.

APPLICATION ACTIVITY 3.4

1. Examine 2 reasons that cause latency in computer network.
2. What is your school download bandwidth and upload bandwidth?
3. Contrast upload bandwidth and download bandwidth?
4. Categorize following devices according to their transmission modes
   a. TV
   b. Radio
   c. Walkie -Talkie
   d. Phone
   e. Computer

3.2.3. Internetwork (Internet, Intranet, Extranet)

ACTIVITY 3.5

Using computer network, an organization's employees can access data when they are inside the organization buildingand when they are physically outside. Investigate technologies that can be used to allow employees inside and outside to access organization's data.

A network of networks is called an internetwork, or simply the **Internet**. The Internet, extranets, and intranets all rely on Transport Control Protocol / Internet Protocol (TCP/IP). However, they are different in terms of the levels of access they allow to various users inside and outside the organization and the size of the network.

1. **An Intranet** is a private computer network that uses Internet Protocol to securely share any part of an organization's information or operational systems within that organization. Only users inside the organization are only allowed to access it.
2. **An Extranet** is a private network that uses Internet protocols, network

connectivity. An extranet can be viewed as part of a company's intranet that is extended to users outside the company, the connectivity is made possible by the Internet.

3. **The Internet** is a global system of interconnected computer networks that use the standard Internet Protocol suite (TCP/IP) to serve billions of users worldwide.

The difference between the Internet and Extranets is that while the extranet allows limited access to non-members of an organization, the Internet generally allows everyone to access all network resources.

APPLICATION ACTIVITY 3.5

General supply Ltd. sell various services online and offline. It has various partners such as suppliers, sellers, buyers and clients in different Districts of Rwanda. Each partner wants to access the General supply data from its servers located at the main office in Ruhango District.

Examine in details the appropriate technology for each user to access the General supply Ltd. data to avoid the user location problem

3.3 Local Area Networks (LANs)

ACTIVITY 3.6

By definition, computer network is said to be a group of computers connected together and share resources. Using your school environment, observe followings:

1. The range covered by your school network?
2. The number of users on your school network?
3. Resources shared on your school network?

3.3.1 Definition

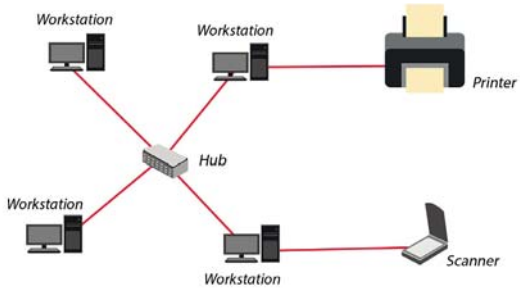
Local Area Networks (LANs) are networks usually confined to a geographic area, such as a single building or a college campus. LANs can be small, linking as few as three computers, but often link hundreds of computers used by thousands of people.

3.3.2 LAN Categories

1. Peer-to-Peer Network

A peer-to-peer (P2P) network is created when two or more computers are connected and share resources without going through a separate server computer. In a peer-to-peer network, there is no hierarchy among the computers, nor are there any dedicated servers.

Each device on the network, also called a client, has equivalent capabilities and responsibilities. A user is responsible for its own resources and can decide which data and devices to be shared with other computers. Because individual users are responsible for the resources on their own computers, the network has no central point of control and no central administration. Peer-to-peer networks work best in environments with ten or fewer computers.



Picture 3.2: Peer-to-Peer network

Advantages of peer to peer network are as follows:

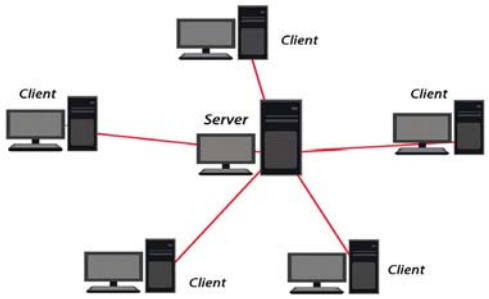
- The main advantage of peer to peer network is that it is easier to set up
- In peer-to-peer networks all nodes are act as server as well as client therefore no need of dedicated server
- The peer to peer network is less expensive
- Peer to peer network is easier to set up and use this means that you can spend less time in the configuration and implementation of peer to peer network.

Disadvantages of peer to peer network

- A computer can be accessed anytime
- Network security has to be applied to each computer separately
- Backup has to be performed on each computer separately
- No centralized server is available to manage and control the access of data.

2. Client-Server Network

In a client-server network, the client requests information or services from the server and the server provides the requested information or service to the client. Servers on a client-server network commonly perform some of the processing work for client machines.



Picture 3.3: Client-Server Network

In a client-server network, resources are controlled by a centralized network administration. The network administrator implements data backups and security measures. The network administrator also controls user access to the server resources.

Note: In a home or small business, a single server can run multiple types of server software, it may be necessary for one computer to act as a file server, a web server, and an email server. A client computer can also run multiple types of client software. There must be client software for every service required. With multiple client software installed, a client can connect to multiple servers at the same time.

Advantages of Client-Server Network

- **Centralization of control:** Access, resources and integrity of the data are controlled by the dedicated server so that a program or unauthorized client cannot damage the system.
- **Scalability:** You can increase the capacity of clients and servers separately. Any element can be increased (or enhanced) at any time, you can add new nodes to the network (clients or servers).
- **Easy maintenance:** Distribute the roles and responsibilities to several standalone computers, you can replace, repair, upgrade, or even move a server, while customers will not be affected by that change (or minimally affect).

Disadvantage of Client Server Networks

- There is a reliance on the central server, if it fails, no work can be done
- A network manager is required and this costs money
- The server costs money, as does the network operating system
- Servers are powerful, thus expensive
- Lots of network traffic.

Advantages of Client Server Networks over Peer to Peer Networks

- **Centralization:** Unlike Peer to Peer, where there is no central administration, Client Server Networks have a centralized control.
- **Proper Management :** All the files are stored at the same place. Therefore, management of files becomes easy and it is easier to find files.
- **Back-up and Recovery possible:** As all the data is stored on server it is easy to make a back-up of it. Also, in case of some break-down if data is lost, it can be recovered easily and efficiently. While in peer to peer network, we have to take back-up at every workstation.
- **Upgrade and Scalability in Client-server set-up:** Changes can be made easily by just upgrading the server. Also new resources and systems can be added by making necessary changes in server.
- **Accessibility:** From various platforms in the network, server can be accessed remotely.
- Security: Rules defining security and access rights can be defined at the time of set-up of server.

Disadvantages of client server network over the Peer to peer network

- Congestion in Network: Too many requests from the clients may lead to congestion, which rarely takes place in Peer to Peer network. Overload can lead to breaking-down of servers. In peer-to-peer, the total bandwidth of the network increases as the number of peers increase.
- Client-Server network is not as robust as a Peer to Peer network and if the server fails, the whole network goes down.
- Cost: It is very expensive to install and manage client server network.

APPLICATION ACTIVITY 3.6

Respond the following questions by True or False and justify your answer

- Peer to Peer network is only implemented by computers which run same operating system.
- In Server-Client network, Server respond to service only requested by the Client.
- Limit number of hosts in the Peer to Peer network are 10 computers.
- Client requests only the service available on the server in Server-Client network.
- Peer to Peer network is more secure than Server-Client network.
- There is a dedicated computer in Peer to Peer network.
- In Client- Server network, both client and server must run the same Operating System.

3.4. Physical Components

ACTIVITY 3.7

In the computer lab observe devices that are directly participating in the network. Examine the role of each device you have identified.

3.4.1. Definition of network device

A network host or a node is a computer or any other device that is directly connected to a computer network. A network host may offer information resources, services, and applications to users.

i. Network Interface Card (NIC)

Picture 3.4: Network Interface Card
Source: http://www.alpagut.com/ImageGallery.aspx?PID=1196&t=1

Network Interface card is a component that allows the computer to communicate across a network. This component is frequently built into the motherboard of today's computers, but it can also be a separate card for use in a PCI slot.

Network Interface cards can be either wired or wireless. However, some cards do support both wireless and wired networking

APPLICATION ACTIVITIES 3.7

NIC Card installation in a desktop computer:

Step 1: Shut down your Computer and disconnect it from the power source

Step 2: Remove the case cover. Then remove the cover of the available slot.

Step 3: Install NIC in proper slot and secure it

Step4: Replace the case cover.

Note: A wireless NIC has an antenna connected to the back of the card or attached with a cable so that it can be positioned for the best signal reception. You must connect and position the antenna.

Picture 3. 5: Wireless Network Interface Card

Source: https://www.ictlounge.com/html/network_hardware.htm

ii. Media Access Control Address

Media Access Control Address (MAC address) of a device is a unique identifier assigned to network interfaces for communications at the data link layer of a network segment. MAC address is a physical address of Network Interface Card. In other words MAC addresses are linked to the hardware of network adapters. A MAC address is given to a network adapter when it is manufactured.

Example of a MAC address: *00:0a:95:9d:68:16*.

iii. Modem

A modem is a hardware device that enables a computer to send and receive data over a telephone line or a cable or satellite connection. Modem is used to transmit digital information via analog systems. The word "modem" is derived from the term "**Mo**dulator - **Dem**odulator."

The essential functions of a modem are:

- **Modulate:** an analog carrier signal to carry digital information, it means to convert the analog signal to digital signal.
- **Demodulate:** a similar signal so as to decode the digital information from the analog carrier signal and it means to convert the same signal back to the analog signal then transmitted through telephone line.

They are two types of modems:

Internal modem which are circuit boards that plug into a computer's motherboard and external modems which are discrete units housed in a separate case. Typically, an external modem is connected to the telephone line and the computer via cables or USB.

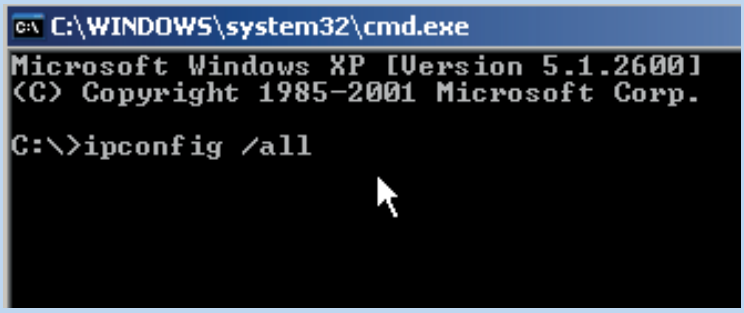
APPLICATION ACTIVITY 3.8

Find the Mac address of your computer

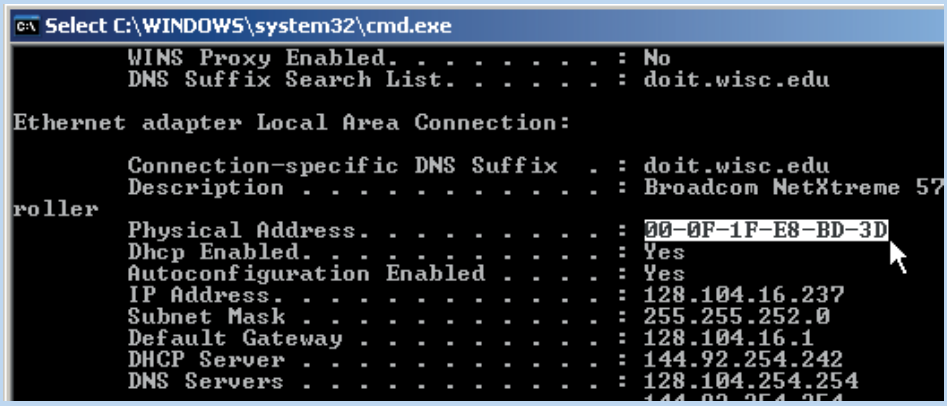
Step1: Click the **Run** button in the windows 10 **Start Menu**

Step 2: Type **cmd** in the **Open** prompt of the Run menu and click **OK** to launch a command prompt window

Step 3: Type **ipconfig /all** at the command prompt to check the network card settings.



Step4: MAC address is listed by ipconfig under **Physical Address**



iv. RJ 45 Connector and Port

A registered jack (RJ) is a standardized physical network interface for connecting telecommunications or data equipment. The physical connectors that registered jacks use are mainly of the modular connector and 50-pin miniature ribbon connector types

The RJ45 port is the network port on a computer. This socket has many names. It is also known as the Ethernet port, the network adapter, the network jack or the RJ45 jack.

3.5. Network Devices

ACTIVITY 3.8

Observe the connection between network devices that make your school network. Then answer following questions:

1. Examine how computers are connected together in order to exchange data?
2. Investigate how data / message is forwarded from the Computer A (as source) to Computer B (as destination)?
3. The equipment used to connect devices within a LAN has evolved from hub to bridge to switch. Contrast both devices?

3.5.1 Hub

A hub is a network hardware device for connecting multiple devices together and making them act as a single **network segment.**

Definition:A network segment is a portion of a computer network that is separated from the rest of the network by a network device.

Hub receives message on one port and then send it out to all other ports, this means that when hub receives message, the received message is regenerated or duplicated and sent to all computers connected to the hub, each computer on the network receives the message, if it is not the destination message is destroyed, if it is the destination it reads the message.

Note: Hubs are used less often today because of the effectiveness and low cost of switch. Hub do not segment network traffic. When one device sends traffic, the hub floods that traffic to all other devices connected to hub. The devices are sharing the bandwidth.

3.5.2 Switches

Switch filters and segments network traffic by sending messages only to the device to which it is sent. This means that when a switch receives a message do not duplicated it, it sends it directly to the destination computer. This provides higher dedicated bandwidth to each device on the network.

A switch maintains a switching table, the switching table contains a list of all MAC addresses of computers on the network and a list of switch port which are used to reach a computer with a given MAC address.

When message arrives that is destined for a particular MAC address, the switch uses the switching table to determine which port to use to reach the MAC address. Then

message is forwarded to the destination.

3.5.3 Bridge

A network bridge is a computer networking device that creates a single aggregate network from multiple communication networks or network segments. This function is called **network bridging**

Bridges keep a record of all the devices on each segment. A bridge can then filter network traffic between LAN segments. This helps reduce the amount of traffic between devices

3.5.4 Access point

Picture 3.6: Linksys access point ***Source: https://en.wikipedia.org/wiki/Wireless_access_point***

The Access Point is connected to a switch using UTP cable, therefore it can provide access to the rest of the network. Instead of providing copper cabling to every network host, only the wireless access point is connected to the network with copper cabling and spread radio waves to the rest of network.

The range (radius of coverage) for Access Point indoors is 98.4 ft (30 m) and too much greater distances outdoors depending on the technology used.

APPLICATION ACTIVITY 3.9

1. Justify why outdoor range is better that indoor range?
2. Discuss how a switch perform micro-segmentation?
3. Analysis why Switch is much preferred than a HUB
4. Examine the difference between Access Point and Bridge?
5. Identify the purpose of switching table in the switch?
6. Using the internet search for the diagram representation of hubs, switches and bridges in a network

3.6 Network Transmission Medium

ACTIVITY 3.9

Draw a computer network of 10 computers

This is the physical mean of communication between network computers.

Data transmission media are the physical materials used to transmit data between computers. Packet of data can be transmitted on network as **electrical signals** in **electric wire**, **light signal** in fiber optic cables or as **electromagnetic waves** through space.

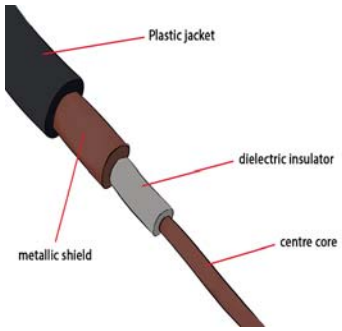
There are two main types of data communication media used in network:

1. **Bounded or Guided media:** which transmit signals by sending electricity or light over a cable wire. Common examples of bounded media are twisted cables, Coaxial cables and fiber optic cables.
2. **Unbounded media**: which transmit data through the air, radio waves, layer or infrared signal and satellite based microwaves, etc.

3.6.1 Guided transmission media

A wide variety of networking cables are **Coaxial, Fiber optic** and **twisted-pair** cables which use electrical signals over copper to transmit data while Fiber-optic cables use light signals to transmit data. These cables differ in bandwidth, size, and cost.

i. Coaxial cable



Picture 3.7: Coaxial cable parts

It is used by both cable television companies and satellite communication systems and it carries data in the form of electrical signals. There are several types of coaxial cable:

- **Thicknet or 10BASE5** - used in networks and operated at 10 Mb/s with a maximum length of 1640.4 ft. (500 m.)

- **Thinnet 10BASE2** - used in networks and operated at 10 Mb/s with a maximum length of 607 ft. (185 m.)

b. Twisted-pair copper cabling

Twisted-pair is a type of copper cabling used for telephone communications and most Ethernet networks. The pair is twisted to provide protection against crosstalk, which is the noise generated by adjacent pairs of wires in the cable.

The use of this cable is limited by two factors:

a. Attenuation: the strength of signal reduces as the distance increases i.e the cables loose signals strength when they exceed their maximum length stated in network specification.

b. Crosstalk: this refers to interference generated by cables when they are too close to each other. Signals from one line get mixed with signals from another.

3.6.2. Types of twisted pair cables

There are two types of twisted pair cables: unshielded and shielded cables

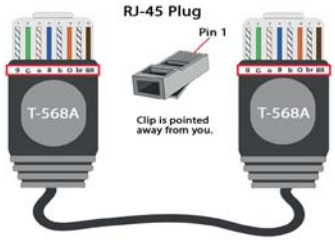
a. Unshielded twisted-pair (UTP)

Unshielded twisted-pair (UTP) cabling is the most common variety of twisted-pair cabling.

UTP cable consists of four pairs of color-coded wires that are twisted together and then encased in a flexible plastic sheath that protects from minor physical damage. The twisting of wires helps protect against crosstalk. However, UTP does not protect against electromagnetic interference (EMI) or radio frequency interference (RFI). EMI and RFI can be caused by a variety of sources including electric motors and fluorescent lights.

There are two different wiring schemes called T568A and T568B. Using the T568A and T568B wiring schemes, two types of cables can be created: a **straight-through** cable and a **crossover cable.**

1. **A straight-through cable** is the most common cable type. It maps a wire to the same pins on both ends of the cable. The order of connections (the pin out) for each color is the exact same on both ends.



Picture 3.8: Wiring Schemes T568A

Two devices directly connected and using different pins for Transmit and Receive are known as unlike devices. They require a straight-through cable to exchange data. For example, connecting a PC to a switch requires a straight-through cable.

2. **A crossover cable** uses both wiring schemes. T568A on one end of the cable and T568B on the other end of the same cable.

Devices that are directly connected and use the same pins for transmit and receive, are known as like devices. They require the use of a crossover cable to exchange data. For example, connecting a PC to another PC requires a crossover cable.

Note: The most used cable standards are below:

- 100Base-TX: known as Fast Ethernet, it uses category 5, 5E, or 6 UTP cable and it wires u to 100 meters long.
- 1000Base-T means that the speed of the network is up to 1000 Mbps, baseband signaling is sed, T stands for twisted-pair as UTP cable used

b. **Shielded twisted-pair (STP)**

The STP type similar to UTP cable except that there is a metal foil or braided metal-mesh cover that encases each pair of insulated wires. The extra covering in shielded twisted pair wiring protect the transmission line from electromagnetic interference (EMI) leaking into or out of cable. However this can make cables quite bulky and harder to install.

The basic difference between UTP and STP is that UTP is a cable with wires that are twisted together to reduce noise and crosstalk. On other hand, STP is a twisted pair cable confined in foil or mesh shield that guards the cable against electromagnetic interference.

APPLICATION ACTIVITY 3.10

Requirements:

1. Unshielded twisted pair (UTP) patch cable
2. Modular connector (8P8C plug, aka RJ45)
3. Crimping tool
4. Cable tester (optional, but recommended

Step 1: Strip the cable jacket about 1.5 inch down from the end.

Step 2: Spread the four pairs of twisted wire apart.

Step 3: Untwist the wire pairs and neatly align them in the T568B orientation.

Step 4: Cut the wires as straight as possible, about 0.5 inch above the end of the jacket.

Step 5: Carefully insert the wires all the way into the modular connector, making sure that each wire passes through the appropriate guides inside the connector.

Step 5: Carefully insert the wires all the way into the modular connector, making sure that each wire passes through the appropriate guides inside the connector.

Step 6: Push the connector inside the crimping tool and squeeze the crimper all the way down.

Step 7: Repeat steps 1-6 for the other end of the cable.

Step 8: To make sure you›ve successfully terminated each end of the cable, use a cable tester to test each pin.

APPLICATION ACTIVITY 3.11

Step 1: Open the Network and Sharing Center:

Step 2: Select the Control Panel from the Start menu.

Step 3: Select Open Network and Sharing Center and click Network and Sharing Center

3.6.3 Fiber optic cables

An optical fiber cable, also known as a fiber optic cable, is an assembly similar to an electrical cable, but containing one or more optical fibers that are used to carry light signals

Advantages of fiber optic

a. Since they transmit light rather than electronic signal, the problem of electrical interference is eliminated , therefore they are not affected by radio interference or cross talk
b. They have become the standard for connecting networks between buildings due to their immunity the effects of moisture and lighting.
c. Fiber optic can transmit signal over much longer distance than coaxial cable and twisted cables
d. They can carry information at high rate speed between
e. The distance can be up to 2000 meters without repeater
f. Security: it is difficult to tap into optical fiber line. If this is happen it will be noticed immediately.
g. Low transmission loss
h. Data can be transmitted digitally
i. They are more resistant to adverse weather conditions.

3.6.4 Unguided transmission media

Unguided media transport electromagnetic waves without using a physical conductor. This type of communication is often referred to as wireless communication. The mediums used in wireless communications are air, vacuum and even water. Air is the most commonly used medium.

a. Wireless Transmission

Wireless transmission can be categorized into three broad groups namely radio waves, microwaves, infrared

1. Radio waves

Radio waves are normally **omnidirectional.** When an antenna transmits radio waves, they are propagated in all directions. This means that the sending and receiving antennas do not have to be aligned. The omnidirectional characteristics of radio waves make them useful for multicasting, in which there is one sender but many receivers. Our FM radio stations, cordless phones and televisions are examples of multicasting.

2. Infrared

Infrared is used in devices such as the mouse, wireless keyboard and printers. Some manufacturers provide a special port called the IrDA port that allows a wireless keyboard to communicate with a PC.

Infrared signals have frequencies between 300 GHz to 400 THz. They are used for short-range communication. Infrared signals have high frequencies and cannot penetrate walls. Due to its short-range communication system, the use of an infrared communication system in one room will not be affected by the use of another system in the next room. This is why using an infrared TV remote control in our home will not interfere with the use of our neighbor's infrared TV remote control.

The disadvantages of using infrared

- Infrared signals cannot be used for long distance communication.
- Infrared waves can not be used outside of a building because sun's rays contain infrared waves that can interfere with communication.

3. Bluetooth

Bluetooth technology is designed to serve as a new way of connecting devices. Bluetooth technology has an advantage of being low investment and low energy consumption demanding.

Th difference between Bluetooth and infrared is that Bluetooth allows communication when there is a barrier or a wall.

4. Wi-Fi (Wireless Fidelity)

Wi-Fi (Wireless Fidelity) is a standard that certifies that wireless devices in Wireless LAN to work together. It supports IEEE802.11b Ethernet standard.

Wi-Fi uses radio signals to transmit high speed data over the wireless network, the Access Point is used to connect devices and it acts as the central device for Wireless LAN. Wireless devices can be: smartphones, PDAs, IPad, laptop and notebook.

APPLICATION ACTIVITY 3.11

General steps to connect to a WI-FI network near you

The following steps run through the general steps that anyone needs to take to get connected to the internet via WI-FI. (Do this exercise and discover the corresponding windows for every step. What is the meaning of different icons forms that a wireless network take?)

Step 1: Locate yourself in a property or public space that has a wireless router.

Step 2: Make sure that the device you›re going to use is:

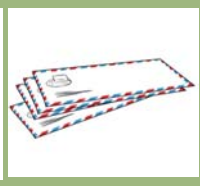
- Capable of connecting to the internet and
- Capable of connecting to WI-FI.

Step 3: Find out the name of the WI-FI network that the router in your location is transmitting.

Step 4: Once you know the name of the WI-FI network, use your chosen device to find it.

3.7. IP Address

3.7. IP Address



The fingerprint and addressed letters are ways of identifying and addressing a person. A person's fingerprints usually do not change. They provide a way to physically identify people. The mailing address of a person can change, as it relates to where the person lives or picks up mail. Do network devices have addresses? Justify your answer.

3.7.1 Understand IP Addresses

An IP address is an address used in order to uniquely identify a device on a computer network.

IP Versions

There are two versions of IP addressknown as IP version 4 and IP version 6

a. IP version 4

IP version 4 is the common used IP address**,**

a.1 Structure of IP address

- An IP address is simply a series of 32 binary bits. Because it is difficult for human to read a binary IP address, the 32 bits are grouped into fout-8-bits bytes called octets.
- When a computer is configured with an IP address it is entered as doted decimal number such as 192.168.1.5
- When a host receives an IP address , it looks at all 32 bits as they are received by the NIC
- Each octet is made up with 8 bits an each bit has a value. The four groups of 8 bits have same set of value.
- Parts of the IP address

The logical 32 bits IP address is made up two parts which are network part or **network address** and the second one **is host address** .

a.2 Classes of IP version 4

There are five classes of available IP ranges: Class A, Class B, Class C, Class D and Class E, while only A, B, and C are commonly used classes D and E are reserved.

Here is a listing of these addresses:

Class A

In class A the first bit of the first byte is always 0

0xxxxxxx where x stand for any value (either o or 1)

Minimum: 00000000
Maximum: 01111111

The valid range in class A =

The range of IPs 127.x.x.x is reserved for the **loopback or localhost,** for example, **127.0.0.1** is the common loopback address. Range **255.255.255.255**broadcasts to

all hosts on the local network.

The first byte in class A represent Network address while the three remaining represent the host address thus class A=Network. Host-Host-Host

Class B

In class B the first bit of first byte is always 1 and the second bit of first byte is always 0

10xxxxxx

Minimum: 10000000=128

Maximum: 10111111=191

Valid range[128-191]

The two first byte in class B represent Network address while the two remaining represent the host address thus class B=Network-Network-Host-Host

Class C

The first bit of first byte is always 1, the second bit of first byte is always 1 and the third bit of first byte is always 0 ***i.e*** **1100000**

Minimum: 11000000= 192

Maximum: 11101111= 223

Valid range: 192-223

The three first bytes in class C represent Network address while the last byte remaining represent the host address thus class C=Network. Network -Network -Host

a.3 Subnet Masks

A subnet mask help to know which portion of the address identifies the network and which portion of the address identifies the host. Class A, B, and C networks have default masks, also known as natural masks, as shown here:

Class A: 255.0.0.0

Class B: 255.255.0.0

Class C: 255.255.255.0

IP Address Class	Total number of bits for network ID / host ID	First octet of IP address	Number of network ID bits used to identify class	Usable number of network ID bits	Number of possible network IDs	Number of host IDs per network ID
Class A	8 / 24	0xxx xxxx	1	8-1 = 7	27-2 = 126	224-2 = 16,277,214
Class B	16 / 16	10xx xxxx	2	16-2 = 14	214 = 16,384	216-2 = 65,534
Class C	24 / 8	110x xxxx	3	24-3 = 21	221 = 2,097,152	28-2 = 254
Class D	Reserved for Multicasting					
Class E	Reserved for Multicasting					

Table 3.1: classes of IP addresses

Devices that are attached to a network have two addresses that are similar to a person's fingerprints and a person's mailing address. These two types of addresses are the Media Access Control (MAC) address and the IP address. The MAC address is hard coded onto the network interface card (NIC) by the manufacturer and it does not change. The IP address is assigned to a host to the network and may change.

b. Internet Protocol Version 6 (IPv6)

IPv6 (**Internet Protocol Version 6**) also called IPng (**I***nternet* **P***rotocol* **n***ext* **g***eneration*) is the newest version of the Internet Protocol reviewed to replace the current version of IPv4. IPv6 is designed to allow the Internet to grow steadily, both in terms of the number of hosts connected and the total amount of data traffic transmitted.

IPv6 shows a 128-bit address in eight 16-bitblocks separated by colons

Example: 3ffe:1900:4545:3:200:f8ff:fe21:67cf.

FE80:0000:0000:0000:0202:B3FF:FE1E:8329

b.1 Features of IPv6

- Supports source and destination addresses that are 128 bits (16 bytes) long.
- No more NAT (Network Address Translation)

- Auto-configuration
- No more private address collisions. Requires IPSec support to provide two security headers which can be used separately
- Those security headers are Authentication Headers(AH) and Encrypting security payload(ESP)
- Uses Flow Label field to identify packet flow for Quality of Service handling by router.
- Allows the host to send fragments packets but not routers.
- Does not require manual configuration or DHCP.
- Moves optional data to IPv6 extension headers.
- Uses Multicast Neighbor Solicitation messages to resolve IP addresses to link-layer addresses.
- Uses Multicast Listener Discovery (MLD) messages to manage membership in local subnet groups.
- Uses ICMPv6 Router Solicitation and Router Advertisement messages to determine the IP address of the best default gateway

Why there is no different classes in IP v6?

In IPv4 we have class A, B, C,D and E. In IPv6 we have only global prefix and interface id

IP v6 has three types of addresses, which can be categorized by type and scope:

Unicast addresses: a packet is delivered to one interface.

Multicast addresses: a packet is delivered to multiple interfaces.

Anycast addresses:a packet is delivered to the nearest of multiple interfaces (in terms of routing distance).

3.7.2. IP address assignment1. Static IP assignment

The manual configuration of a host in a network, allows to assign the static IP address and the following information has to be specified:

- **IP address** - identifies the computer on the network
- **Subnet mask** - is used to identify the network on which the computer is connected
- **Default gateway** - identifies the device that the computer uses to access the Internet or another network

- **Optional values** - such as the preferred Domain Name System (DNS) server address and the alternate DNS server address

Note that host on the same network should have the same Network ID

APPLICATION ACTIVITY 3.12

Go through all these steps to assign to your computer a static IP address

STEP 1: Go to Network Connections.

STEP 2: Go to the network connection your computer is currently using, then select Properties.

STEP 3: Go to Internet Protocol Version 4 (IPv4) from the list and click the Properties button

STEP 4: Choose Use the following IP address option

NOTE: When assigning a static IP address, make sure to set a value that will not conflict with others on your network. You may refer to your router documentation for the range of IP address that it assigns to its devices.

STEP 5: Next to the IP address field, enter the IP address value you want to assign to the computer.

NOTE: If the DHCP range of your router is from say 192.168.1.100-200, you can use any range from 192.168.1.2-99 for example.

STEP 6: In the Subnet mask field, enter "255.255.255.0."

STEP 7: In the Default gateway field, enter the IP address of your router or gateway device.

NOTE: Usually, the default gateway address for most routers is 192.168.1.1. Again, you may refer to your device documentation to determine the correct IP address of your router.

STEP 8: For the Preferred DNS server and Alternate DNS server fields, you can take a look for the exact DNS values that your router is getting from your ISP, which can be usually found by accessing your router's configuration page.

STEP 9: Click OK on all windows to save your changes.

3. Dynamic IP configuration

A DHCP (Dynamic Host Configuration Protocol) server automatically assigns IP address to host. It simplifies the addressing process. DHCP server can automatically assign to a host an IP addres, Subnet mask, Default gateway and Optional values, such as a DNS server address

APPLICATION ACTIVITY 3.13

Go through all the steps to assign an IP address using DHCP

Step 1: Go to network connections

Step 2: Select "View network connections".

Step 3: Locate the "Network connection" you want to view

Step 4: After locating connection, right –click on it and select "Properties"

Step 5: Double-click "Internet Protocol Version (TCP/IPv4)"

Step 6: Select the followings:

- Obtain an IP address automatically
- Obtain DNS server addresses automatically
- OK

Step 7: Select "OK" and restart your computer

3.7.3 Some common protocols

A protocol is defined a set of rules and procedures that control communication between computers or other network devices on a network.

a. Internet control message protocol (ICMP)

Internet Control Message Protocol (ICMP) is used by devices on a network to send control and error messages to computers and servers. There are several different uses for ICMP, such as announcing network errors, announcing network congestion, and troubleshooting.

b. HTTP (Hypertext Transfer Protocol)

e.g. http://www.reb.rw/

http:stands for hypertext transfer protocol used to connect computer to the server. It provides a standard web browser to communicate with a server FTP (File Transfer Protocol)

This protocol allows you to transfer files between two computers on internet.

c. TCP and UDP

Transmission Control Protocol and User Datagram Protocolare most common transport layer both protocols that manage the communication of multiple applications. The difference between the two are the specification function that each protocol implements.

d. SMTP (Simple mail Transfer protocol)

The SMTP stands for Simple mail Transfer protocol is a TCP/IP used in sending and receiving e-mails. However, since it is limited in its ability to queue message at the receiving end, it is used with one of two other protocols POP or IMAP that let users to save messages in server mailbox and download them from the server once needed.

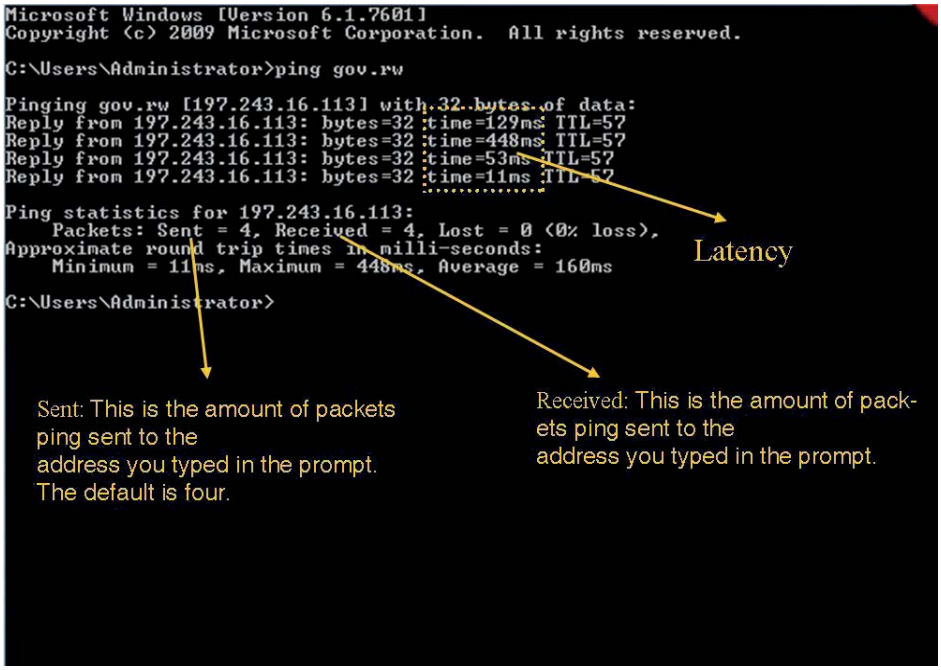
e. POP (Post Office Protocol)

Post Office Protocol is a protocol designed to allow single user computer to retrieve electronic mail from a POP server via TCP/IP.

3.7.4 Commands used to verify computer connection on a network

3.7.4.1 Ping

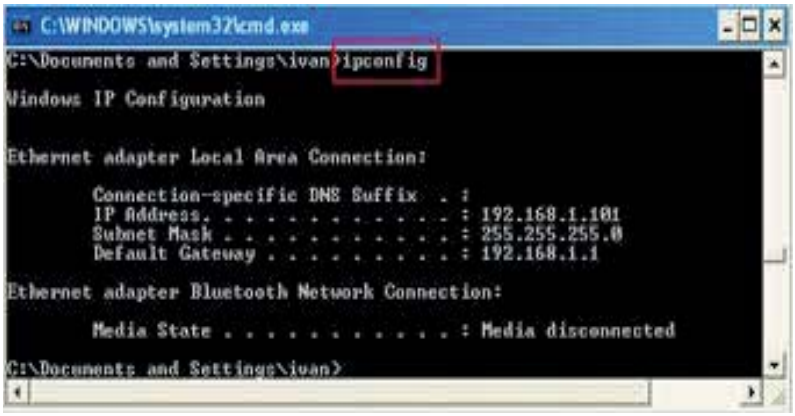
The command Ping is commonly used to test connections between computers. Ping is a simple but highly useful command-line utility used to determine whether a specific IP address is accessible.



Picture 3.9. Ping command screenshot

3.7.4.2 ipconfig

The ipconfig command is used to find out your current local IP address, default gateway, TCP/IP settings and more.



Picture 3.10: Checking the IP address of a computer on the network

All Ipconfig commands explained:

i. ipconfig/all: it displays all current IP information for all adapters.
ii. ipconfig /release: Used to release current IP information and obtain a new IP Address from the DHCP server.
iii. ipconfig /renew: it is used to renew IP Address if you have it set to obtain IP Address automatically.

3.8. Data and device sharing

ACTIVITY 3.10(b)

1. Share a folder with your classmate on your school computer network
2. Identify how resource are shared on computer network

3.8.1 Workgroup

A workgroup is a collection of computers and servers on a LAN that are designed to communicate and exchange data with one another. Each individual workstation controls its user accounts, security information, and access to data and resources.

APPLICATION ACTIVITY 3.14

Go through the steps below in order to create a **WorkGroup**

Step 1: Open the Start menu, do a search for **HomeGroup** and press **Enter.**

Step 2: Click Create a **HomeGroup.**

Step 3: Click Next.

Step 4: Choose what to share on the network. By default Windows sets Pictures, Videos, Music, and Printers & Devices as Shared. However, the Documents folder is marked as not shared.

Note: The sharing options you choose will only apply to devices connecting with different accounts. If you sign-in with the same user account on another computer, you will have access to all files regardless of what you chose to share.

Step 5: Once you've decided what content to share, click **Next.**

Step 6: The wizard will complete the setup, and you'll be presented with a HomeGroup password, which is needed to allow other computers to access files and printers. Write down or click the link to print the password.

APPLICATION ACTIVITY 3.15

Go through these steps to add computers to a HomeGroup

Step 1: Open the Start menu, do a search for HomeGroup and press Enter.

Step 2: Click the Join now button.

Note: If you don't see the Join now button, there is something wrong, make sure you're connected to the network and restart your computer.

Step 3: Click Next.

Step 4: Select the content you want to share on the network by using the drop down menu for each folder and click Next.

Step 5: Enter your HomeGroup password and click Next.

Note: If you're signed into another computer, but you're using your Microsoft Account, then you won't be prompted to enter a password.

Step 7: Click Finish to complete the task.

3.8.2 Network Resource Sharing

First determine which resources will be shared over the network and the type of permissions users will have to the resources. Any device which is connected to a host /node is called a **Network Peripheral** in other word network peripheral is any device that is indirectly connected to the network. Examples of network peripheral are: printer, scanner

APPLICATION ACTIVITY 3.16

Go through the steps below to add new folders to shared HomeGroup libraries

Step 1: Use the **Windows key + E** keyboard shortcut to open File Explorer.

Step 2: On the left pane, expand your computer's libraries on HomeGroup.

Step 3: Right-click **Documents**.

Step 4: Click **Properties**.

Step 5: Click **Add**.

Step 6: Select the folder you want to share and click **Include folder**.

Step 7: Click **Apply**.

Step 8: Click **OK**.

Application activity 3.17

Go through these steps to add new folder via Windows network share to your HomeGroup

Step 1: Right-click the folder you want to share.

Step 2: Select **Share with** and click the **Homegroup (view)** option.

Users can access the newly shared folder by:

Step 3: Using the **Windows key + E** keyboard shortcut to open File Explorer.

Step 4: Clicking **Network** on the left pane.

Step 5: Double-clicking the computer name with shared content, and browsing the folder location.

By default, every HomeGroup folder shared on the network are set with read only permissions. It has to be this way to prevent accidental deletions and modification to your files by other users.

3.9. Network Topology

ACTIVITY 3.11

Draw a Peer-to-peer network and assign each host an IP address. Network must have 1 switch, 8 computers, one printer which has NIC card and a printer.

The network topology describes the configuration of network, the physical and logical arrangement of nodes that form a network. Network topologies are classified as physical, logical and signals topologies.

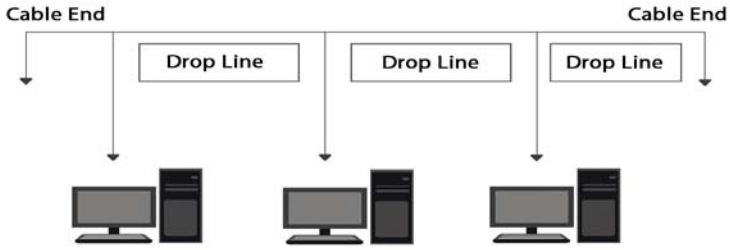
Physical topology describes the mapping of network nodes and physical connection between them. Signal topology describes the paths which signals take while they pass over network that mapping of the paths taken by data as they travel over network. A logical topology is the way data signals pass from one device to another.

There exist different types of network topologies which are Bus topology, Ring topology, Star topology, Mesh topology, Tree topology and Hybrid topology,

3.9.1 BUS Topology

Bus topology is a network type in which every computer and network device is connected to single cable. When it has exactly two endpoints, then it is called **Linear Bus topology**.

It is the most used and employed in LAN architecture. All devices are connected to a central cable, called the bus or backbone. This topology is relatively inexpensive and easy to install for small networks.



Picture 3.9 BUS Topology

Features of Bus Topology

1. It transmits data only in one direction.
2. Every device is connected to a single cable

Advantages of Bus Topology	Disadvantages of Bus Topology
1. It is cost effective. 2. Cable required is least compared to other network topology. 3. Used in small networks. 4. It is easy to understand. 5. Easy to expand joining two cables together.	1. Cables fails then whole network fails. 2. If network traffic is heavy or nodes are more the performance of the network decreases. 3. Cable has a limited length. 4. It is slower than the ring topology.

Table 3.2 advantages and disadvantages of Bus topology

3.9.2 RING Topology

It is called ring topology because it forms a ring as each computer is connected to another computer, with the last one connected to the first. Exactly two neighbors for each device. In a Ring topology each device is connected directly to two other devices, one on either side of it, to form a closed loop. This topology is relatively expensive and difficult to install, but it offers high bandwidth and can span large distances

Picture 3.10: Ring Topology

A ring topology is a computer network configuration in which computer connections create a circular data path. Each networked computer is connected to two others like points on a circle.

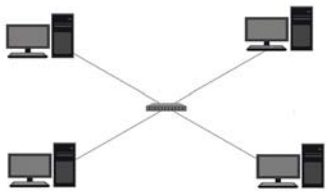
In a ring network, data travel from one device to the next until they reach their destination. Most ring topologies allow data to travel only in one direction called a unidirectional ring network. Others permit data to move in both directions called bidirectional.

Advantages of Ring Topology	Disadvantages of Ring Topology
1. Transmitting network is not affected by high traffic or by adding more nodes, as only the nodes having tokens can transmit data. 2. Cheap to install and expand	1. Troubleshooting is difficult in ring topology. 2. Adding or removing the computers disturbs the network activity. 3. Failure of one computer disturbs the whole network.

Table 3.3 advantages and disadvantages of Ring topology

3.9.3 STAR Topology

In this type of topology all the computers are connected to a single hub or a switch through a cable. This hub or switch acts as the central device and all others nodes are connected to the central device.



Picture 3.11: Star Topology

So, in a Star topology all devices are connected directly to a central computer or server. Such networks are relatively easy to install and manage

Features of Star Topology

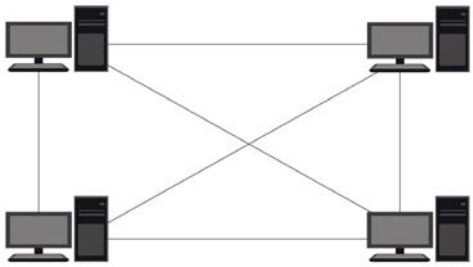
1. Every node has its own dedicated connection to the hub.
2. Hub acts as a repeater for data flow.
3. Can be used with twisted pair or coaxial cable.

Advantages of Star Topology	Disadvantages of Star Topology
1. Fast performance with few nodes and low network traffic. 2. Hub can be upgraded easily. 3. Easy to troubleshoot. 4. Easy to setup and modify 5. Only that node is affected which has failed, rest of the nodes can work smoothly.	1. Cost of installation is high. 2. Expensive to use. 3. If the hub fails then the whole network is stopped because all the nodes depend on the hub. 4. Performance is based on the hub that is it depends on its capacity

Table 3.4 advantages and disadvantages of star topology

3.9.4 MESH Topology

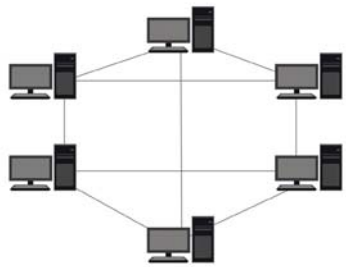
- It is a point-to-point connection to other nodes or devices. All the network nodes are connected to each other. A Mesh topology can be either a full mesh or a partial mesh. In the former, each computer is connected directly to each of the others.



Picture 3.12.Mesh Topology

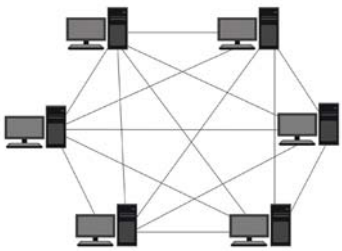
Types of Mesh Topology

1. **Partial Mesh Topology:** In this topology some of the systems are connected in the same fashion as mesh topology but some devices are only connected to two or three devices.



Picture 3.13 Partial mesh topology

2. **Full Mesh Topology:** all devices are connected to each other which is very expensive but provides the best redundancy as a failure of a single does not affect the network connectivity.



Picture 3.14 Full mesh topology

Features of Mesh Topology

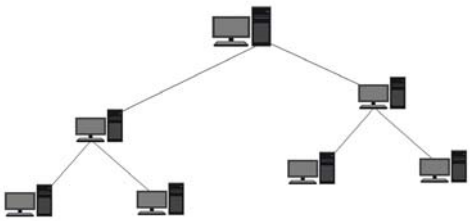
1. Fully connected.
2. Robust.
3. Not flexible.

Advantages of Mesh Topology	Disadvantages of Mesh Topology
1. Each connection can carry its own data load. 2. It is robust. 3. Fault is diagnosed easily. 4. Provides security and privacy.	1. Installation and configuration is difficult. 2. Cabling cost is more. 3. Bulk wiring is required

Table 3.5. advantages and disadvantages of Mesh topology

3.9.5 Tree Topology

It has a root node and all other nodes are connected to it forming a hierarchy. It is also called hierarchical topology. It should at least have three levels of hierarchy.



Picture 3.15 Tree Topology

Features of Tree Topology

1. Ideal if workstations are located in groups.
2. Used in Wide Area Network.

Advantages of Tree Topology	Disadvantages of Tree Topology
1. Extension of bus and star topologies. 2. Expansion of nodes is possible and easy. 3. Easily managed and maintained. 4. Error detection is easily done.	1. Heavily cabled. 2. Costly. 3. If more nodes are added maintenance is difficult. 4. Central hub fails, network fails.

Table 3.6. advantages and disadvantages of Tree topology

3.9.6 Hybrid Topology

Hybrid, as the name suggests, is mixture of two different things. Similarly in this type of topology that integrate two or more different topologies to form a resultant topology which has good points (as well as weaknesses) of all the constituent basic topologies rather than having characteristics of one specific topology. This combination of topologies is done according to the requirements of the organization.

For example, if there exists a ring topology in one office department while there is a bus topology in another department, connecting these two will result in hybrid topology. However, connecting two similar topologies cannot be termed as Hybrid topology.

Advantages of hybrid Topology	**Disadvantages of hybrid Topology**
1. **Reliable:** Unlike other networks, fault detection and troubleshooting is easy in this type of topology. 2. **Scalable:** It's easy to increase the size of network by adding new components, without disturbing existing architecture. 3. **Flexible:** Hybrid Network can be designed according to the requirements of the organization and by optimizing the available resources. 4. **Effective:** Hybrid topology is the combination of two or more topologies, so we can design it in such a way that strengths of constituent topologies are maximized while there weaknesses are neutralized.	a. **Complexity of Design:** One of the biggest drawbacks of hybrid topology is its design. b. **Costly Infrastructure:** As hybrid architectures are usually larger in scale, they require a lot of cables; cooling systems, sophisticate network devices, etc.

T*able 3.8: advantages and disadvantages of Hybrid topology*

Notice that the cost of technology, network devices, and transmission mediums to be used in the computer network has to be considered while choosing the network topology to use.

APPLICATION ACTIVITY 3.18

Using clear example, compare computer network devices, network peripherals and computer peripherals?

1. In an organization there are 20 computers distributed into different offices and all offices share one printer .The managements wants to build a computer network that connects all the computers and printer.
   i. Identify a topology to use in this situation
   ii. Draw arrangement of computers using identified topology
2. Draw a hybrid topology

UNIT 4

INTRODUCTION TO DATABASE

UNIT 4: INTRODUCTION TO DATABASE

Key Unit Competency: To be able to identify concepts of database and differentiate database models

INTRODUCTORY ACTIVITY

Observe the following diagram showing the collection of files for the school. Answer to the questions below:

a. Describe the possible ways that can be used to organize these files?
b. Where will you keep them?
c. Is it easy to retrieve and update some information?
d. What are challenges do you expect in these responsibilities
e. Suggest possible answers to address these challenges.

4.1. Definitions of key terms

Activity 4.1 1.

1. Interview someone from the administration office at your school who is dealing with student's information on daily basis. In your interview, include the following questions:
   a. How do you collect data of students?
   b. What are the data do you need from the student?
   c. Where are you keeping that data?
   d. How do you call the set of all information related to students?
   e. Have you full information about students
   f. With whom else do you share that information?
2. After you have done the interview, write up findings in the form of report to be presented to the class.

i. Data

Data is commonly referred to as 'raw' data – a collection of text, numbers and symbols, images with **no meaning.** Data therefore has to be processed, or provided with a context, before it can have meaning.

ii. Information

Information is the result of processing data, usually by computer. This results in facts, which enables the processed data to be used in context and have meaning. Information is data that **has meaning**.

iii. Database

Database is an **organized collection of related data**. It is considered to be **organized** because the data is stored in categories that are accessible in a logical manner. A database is a collection of one or more **relations**, where each relation is a table made of rows and columns.

Note: An information system is a combination of computer hardware and software that is designed to create, store, process and present information. The heart of all information systems is a database.

In general **data management** consists of following tasks:

Data capture, Data classification, Data storage, Data arranging, Data retrieval, Data maintenance, Data verification, Data coding, Data editing, Data transcription, Data transmission.

Application Activity 4.1

Using the following exam results of S5MCE students:

1. KANEZA has 39/40 in Math, 37/40 in C^{++}, 20/20 in ICT and 15/20 in English,
2. CYUSA has 40/40 in C^{++}, 35/40 in Math, 18/20 in ICT, and 17/20 in English
3. KEZA has 35/40 ni Math, 38/40 in C^{++}, 19/20 in ICT and 16/20 in English,
4. NTWARI has 35/40 in Math, 35/40 in C^{++}, 16/20 in ICT and 15/20 in English,
5. MUTESI has 37/40 in Math, 37/40 in C^{++}, 20/20 in ICT and 20/20 in English

Using spreadsheet:

a. Organize the students' marks in a table
b. Calculate total marks
c. Calculate the Calculate the percentage
d. Arrange their marks by descending order

4.2. Different area where database can be applied

Databases touch all aspects of our lives such as:

1. **Human resources:** to track information about employees
2. **Banking:** to keep customer information, accounts, and loans, and banking transactions.
3. **Airlines:** to keep for reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner
4. **Universities:** to keep student information, course registrations, and grades.
5. **Credit card transactions:** to keep purchases on credit cards and generation of monthly statements.
6. **Telecommunication:** to keep records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.
7. **Finance:** to keep storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.
8. **Manufacturing:** to keep management of supply chain and for tracking production of items in factories, inventories of items in warehouses / stores, and orders for items. etc.

Application activities 4.2

1. Research and identify the impact of database in society
2. Present your findings to the class

4.3. Database approaches

Activity 4.2

Mr. Mugabo is managing a shop in our village. All daily sold products are recorded manually in book note so that he can know how much money he got. Sometimes he uses to write that the payment is cash or not. However, when it is time to make a verification of the whole week so that he can buy new products for the shop, he uses to meet serious problems related to calculations, to know how much sold, how much remain and who have not reimbursed the depts.

1. How Mr. Mugabo can improve his shop management?
2. Identify approaches of database which are replacing manual database system;
3. After that replacement of systems where Mugabo will puts some useless hard copy files?

4.3.1 Traditional File Processing Systems (TFPS) approach

This is an approach which was used earlier, prior to DBMS. With this approach, users had to write their application programs to store data in form of files on the computer permanent storage device (Hard Disk). A user must have knowledge of programming languages but this is not easy for a common computer user, even an experienced programmer would find it difficult to write a program each time a new database was to be created. Each application program written by a user had to define and manage its own data.

a. Advantages of the Traditional File Processing

Compared to manual management of information, the Traditional File Processing presents the following advantages:

- **Simplicity:** the design of file processing is more simple than designing Database
- **Efficiency:** file processing cost less and can be more speed than Database
- **Customization:** you can customize file processing more easily and efficiently than Database because files are related with the application and it have all the data needed for that application.

b. Disadvantages of Traditional File Processing System

- **Separation and Isolation of Data:** In file-based approach, data is stored in separate files, hence it is difficult to access it.

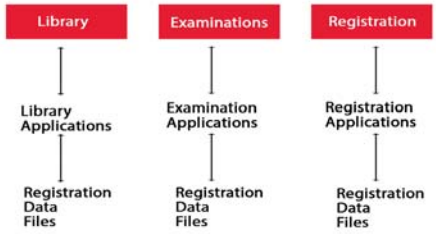


Figure 4.1: TFPS Approach

- **Duplication of Data:** Duplication of data means same data being stored more than once. This can also be termed as data redundancy.
- **Misuse of storage space:** Duplication of data leads to misuse of storage space. If the storage space is not properly it will have a direct impact on cost. The cost will increase.
- **Loss of data integrity:** Data integrity means that the data contained in the database is both accurate and **consistent (**Data inconsistency means different copies of the same data will have different values**)**.
- **Data Dependence:** In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access this file.

- **Security problems**: File based approach is not secured because different files are stored in different locations.

Application activity 4.3

Located in Nyarugenge District, Lycee de Kigali has 50 teachers and there are different departments within. There are 3 departments that need information about those teachers namely Salaries, Restaurant and Dispensary**.** The information for salaries is represented as follows:

	Name	Address	salary	Date of birth
1	MUNEZA	Huye	11000	1/3/1991
2	UMUTESI	Gisagara	11000	7/9/1992
3	MUKANTWARI	Gatsibo	12000	3/2/1990
4	NYINAWUMUNTU	Gasabo	15000	3/1/1992
5	KABALISA	Musanze	13000	3/2/1980

Information for Restaurant is represented as follows:

MUNEZA is from Huye district, Salary is 11000, born on 1/3/1991and his bill is 10000.

UMUTESI was born on 7/9/1992, in Gisagara district, Salary is 11000 and his bill is 15000

MUKANTWARI is from Gatsibo district, she was born on 3/2/1990, Salary is 12000, her bill is 12000.

NYINAWUMUNTU is from Gasabo and born on 3/1/1992 her Salary is 15000 and bill is 11000.

KABALISA is from Musanze district, born on 3/2/1980, Salary is 13000 and bill is 10000

Information for Dispensary is:

MUNEZA from, Huye , Salary is 11000, born on 1/3/1991, bill is 5000.

UMUTESI born on 7/9/1992, from Gisagara, Salary is 11000, bill is 1000

MUKANTWARI from Gatsibo born on 3/2/1990 Salary is 12000, bill is 2000

NYINAWUMUNTU from Gasabo born on 3/1/1992 Salary is 15000, bill is 1000

KABALISA from Musanze born on 3/2/1980 Salary is 13000, bill is 1200

In groups do the following:

a. Analyze how information is organized in 3 departments
b. Write down your critics about this information management in respective departments
c. What do you propose as a solution to minimize the cost of information management at Lycee de Kigali?

4.3.2 Database Management System (DBMS)

Database Management system (DBMS) is referred to as a software system that is used to store, access, manage, organize, maintain, modify and delete data from databases. Some of the most popular software include, **Microsoft Access**, **Oracle, Microsoft SQL Server, MySQL**

MySQL is, one of the most popular database management systems used when there is a need to have access to information online.

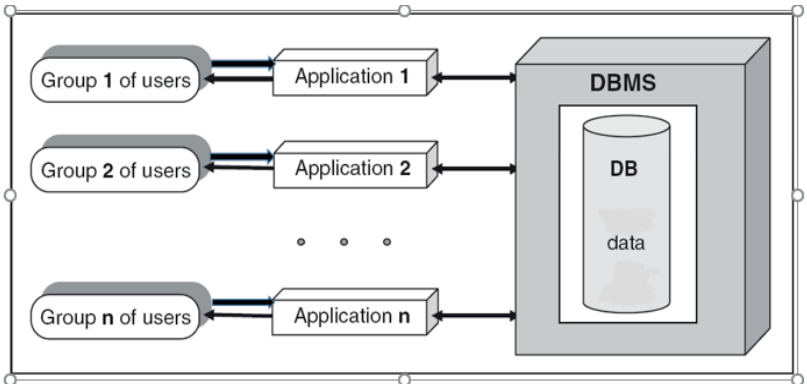


Figure 4.2: DBMS approach

a. The advantages of database management system

There are many advantages of database management system.

1. **Reduce data redundancy:** Data redundancy refers to the duplication or repetition of data. The database system is used to eliminate the problems of data redundancy and data inconsistency.
2. **Data integration:** Data integrity means that the data contained in the database is both accurate and consistent.
3. **Data Independence:** Data independence means that programs are isolated from changes in the way the data are structured and stored.
4. **Reduce data inconsistency:** Actually, data redundancy and data inconsistency are inter-related. If data redundancy is controlled, then data inconsistency will also be controlled automatically. Data inconsistency means different copies of the same data will have different values.
5. **Data sharing:** Due to the fact that data is centralized, many different users from different locations can share data.
6. **Data recovery after a crash (a break down):** DBMS allows to recover data after a crash. The crash may depend on power failure or hardware failure.
7. **Concurrent transaction control:** A transaction means a collection of operations that perform a single action in a database.
8. **Increased Data security and safety:** DBMS allows data to be highly protected against unauthorized access.

b. Disadvantages of DBMS

The disadvantages of the database approach are summarized as follows:

1. **Complexity:** The provision of the functionality that is expected of a good DBMS makes the DBMS an extremely complex piece of software. Database designers, developers, database administrators and end-users must understand this functionality to take full advantage of it.

Failure to understand the system can lead to bad design decisions, which can have serious consequences for an organization.

2. **Size:** The complexity and breadth of functionality makes the DBMS an extremely large piece of software, occupying many megabytes of disk space and requiring substantial amounts of memory to run efficiently.
3. **Performance:** Typically, a File Based system is written for a specific application, such as invoicing. As a result, performance is generally very good.

However, the DBMS is written for general purpose, to cater for many applications rather than just one. The effect is that some applications may not run as fast as they used to.

4. **Higher impact of a failure:** The centralization of resources increases the vulnerability of the system. Since all users and applications rely on the availability of the DBMS, the failure of any component can bring operations to a halt.
5. **Cost of DBMS:** The cost of DBMS varies significantly, depending on the environment and functionality provided. There is also the regular maintenance costs.
6. **Additional Hardware costs:** The disk storage requirements for the DBMS and the database may necessitate the purchase of additional storage space.

Furthermore, to achieve the required performance it may be necessary to purchase a specicalised computer (server), dedicated to running the DBMS. The procurement of additional hardware results in further expenditure.

7. **Cost of Conversion:** In some situations, the cost of the DBMS and extra hardware may be insignificant compared to the cost of converting existing applications to run on the new DBMS and hardware.

This cost also includes the cost of training staff to use these new systems and possibly the employment of specialist staff to help with conversion and running of the system.

This cost is one of the main reasons why some organizations feel tied to their

current systems and cannot switch to modern database technology.

Application activity 4.4

1. Compare Database Management System and traditional File processing system

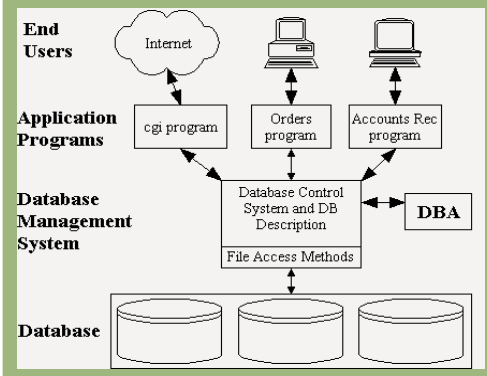
Check the DBMS existing in computers of the school computer lab. If they are more than one, discuss the similarities and differences of them.

4.4 Database access levels and users

Activity 4.3

Activity 4.3

Observe and interpret the following figure.



Source: https://sqlmentalist.com

4.4.1. Database access levels

A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained. There are three-levels that form the basis of modern database architectures:

The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database. The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users.

The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. Usually, a representational data model is used to describe the

conceptual schema when a database system is implemented. This implementation conceptual schema is often based on a conceptual schema design in a high-level data model.

The external or view level: The external view level is closest to the users. It is concerned with the way the data is viewed by individual users. A user can either be an application programmer or an end-user. The external level consists of many different external views of database. At the view level, computer users see a set of application programs that hide details of the data types.

The purpose of the three-level architecture is to separate the user application and the physical database. Different users need different views of the same data.

For example users should not have to deal directly with the physical database storage details. While the database administrator should be able to change the database storage structure or storage device without affecting other user's views.

4.4.2 Database users

When considering users of a Database system, there are three broad classes to consider:

a. Database administrator (DBA):

Responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software and hardware resources, controlling its use and monitoring efficiency of operations.

b. The database designer:

Responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.

c. The end-user:

End-users, They use the data for queries, reports and some of them update the database content.

Application Activity 4.5

From the above figure, explain different access level and right form your school hierarchy such as headmaster, DOS, teachers, students, visitors.

a. Assign each user from the hierarchy what is supposed to do. Example: Everything happening in your school is hidden from the visitors.
b. What is responsibility of DBA?

4.4.3. Data Independence

Activity 4.4

The modification of an application used to access a database does not modify the database itself.

In groups, discuss this assertion.

Data Independence: The ability to modify a scheme definition in one level without affecting a scheme definition in a higher level is called data independence.

There are two kinds of data independence:

a. Logical data independence

The ability to modify the conceptual scheme without causing application programs to be rewritten.

Immunity of external schemas to changes in the conceptual schema usually done when logical structure of database is altered.

b. Physical data independence

The ability to modify the internal scheme without having to change the conceptual or external schemas. Modifications at this level are usually to improve performance.

Application activity 4.6

1. Discuss about logical independence and physical independence
2. Compare data dependence and data independence in database 4.5.4. Data models

4.4.4. Database models

Activity 4.5

Search and summarize found information on Data Models focusing on:

a. Common data models

b. Definition of common data models

A data model is a collection of concepts and rules for the description of the structure of the database. Structure of the database means the data types, the constraints and the relationships for the description or storage of data respectively.

The most often used data models arenHierarchical model network model and Relational model

i. Hierarchical Model

The hierarchical model organizes its data using a tree structure. The root of the tree is the parent followed by child nodes. A child node cannot have more than one parent, though a parent can have many child nodes.

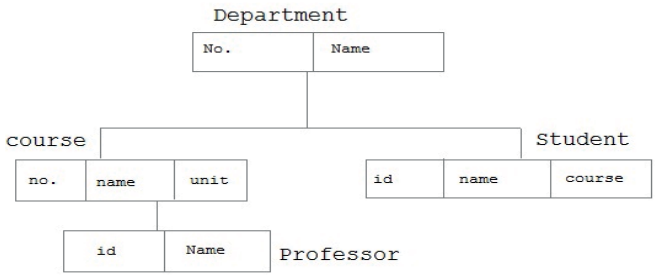


Figure 4.6: Graphical representation of hierarchical model

ii. Network Model

In the network model, entities are organized in a graph, in which some entities can be accessed through several paths.

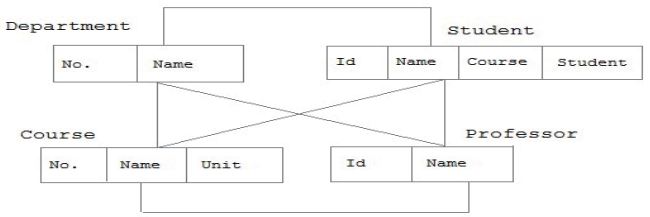


Figure 4.7: Graphical representation of Network model

iii. Relational Model

In this model, data is organized in two-dimensional tables called relations.

The tables or relation are related to each other.

A relational database is a collection of relational tables.

The following is a graphical representation of entities and their relationships in a database structure.

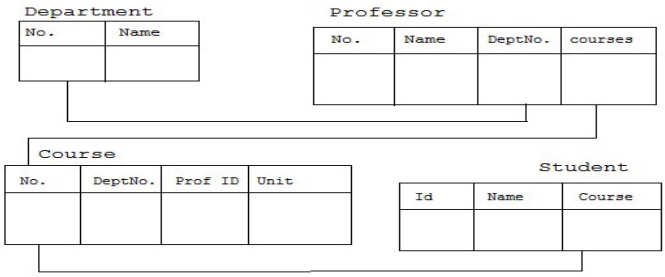


Figure 4.8: Graphical representation of hierarchical model

Notice that since more than three decades, the most used database model is the Relational models. This justifies the fact that most of the currently used DBMS are all relational. They are hence called relational Database Management Systems (RDBMS).

The database systems in their differences and similarities all serve to manage databases. The tables bellow present in summary the Database models Time frame and description, the Database Models Physical Structure, the Database models Structural Changes, the Database models Relationships, the Database models Advantages, the Database models Disadvantages, the Database models Examples and the Database models Status Today.

Database models Comparison

Db Model	Physical Structure
Hierarchical	Tree; parent-child relationships; single table acts as the "root" of the database from which other tables "branch" out; child can only have one parent, but a parent can have multiple children
Network	Network of interrelated lists; looks like several trees that share branches; children have multiple parents and parents have multiple children
Relational	Data is stored in relations (tables); relationships maintained by placing a key field value of one record as an attribute in the related record

Table 4.1: Database models Physical Structure

Db Model	Structural Changes
Hierarchical	Inflexible (once data is organized in a particular way, difficult to change); data reorganization complicated; requires careful design
Network	Inflexible (once data is organized in a particular way, difficult to change). data reorganization complicated; requires careful design

Relational	Flexible; because tables are subject-specific & key fields relate one entity to another, both the data & the database structure can be easily modified & manipulated; programs independent of data format which yields flexibility when modifications are needed

Table 4.2: Database models Structural Changes

Db Model	Relationships
Hierarchical	Linked lists using pointers stored in the parent/child records to navigate through the records. Pointers could be a disk address, the key field, or other random access technique; start at root and work down the tree to reach target data Supports one-to-one & one-to-many relationships
Network	Uses series of linked lists to implement relationships between records. Each list has an owner record & possibly many member records; a single record can either be the owner or a member of several lists of various types. Supports one-to- one, one-to-many, & many-to-many relationships
Relational	Uses key fields to link data in many different ways; supports one-to-one, one-to-many & many-to-many relationships

Table 4.3. Database models Relationships

Db Model	Advantages
Hierarchical	Easily shows one-to-one & one-to-many relationships More efficient than the flat file model b/c less need for redundant data
Network	Network model solves problem of data redundancy by representing relationships in terms of sets rather than hierarchy. Allowed complex data structures to be built; very efficient in storage & fast; better job with many-to-many relationship

Relational	More easily visualize data organization & relationships ease of design & user-friendly GUI interfaces. No duplicate data which reduces errors & improves consistency & eases database maintenance. Relationships increase data integrity; more information from same data due to file integration; new & one-time requests easily accommodated. Increased productivity; application code generators; referential integrity controls; transactional integrity ensures that incomplete transactions do not occur. Database users don't corrupt each other's work; scalable (can spread load across multiple CPUs or servers). Best for data collection & querying; allows built-in validation; define & store schema (DB structure), Load initial data, provide a variety of access methods, add, modify & delete data, provide multiple views of the data, provide security features, facilitate backup and recovery

Table 4.4. Database models Advantages

Db Model	Disadvantages
Hierarchical	No support for many-to-many relationships; user must know how the tree is structured to find anything; cannot add a record to a child table until it has been incorporated into the parent table; still creates data duplication
Network	More difficult to navigate and visualize compared to the hierarchical model. Model difficult to implement and maintain. Most implementations were used by programmers rather than end-users; had to have an understanding of how the database was structured in order to retrieve, insert, update or delete records. Required a lot of programming to use successfully

Relational	Initially, had slow performance in 70's & 80's, but today's more powerful machines speed up performance Wide price range, can be expensive High impact of failure due to centralization and non-duplication of data. Sophisticated design and programming required for some products; additional user training may be needed due to added system capabilities. Backup and recovery are more difficult due to concurrency and complexity; security is more critical and complicated due to shared usage of central database.

Table 4.5. Database models Disadvantages

Db Model	Examples
Hierarchical	IMS (Information Management System) by IBM
Network	Satellite communications, airline reservations;
Relational	Oracle, DB2, MS Access, SQL Server, MySQL (free),

Table 4.6. Database models Examples

Db Model	Status Today
Hierarchical	Limited usage (i.e. Windows file structure)
Network	Limited usage (i.e. still used in satellite communications & airline reservation systems
Relational	Most popular DBMS in use today as a result of technical development efforts to ensure that advances such as object orientation, web serving, etc. appear quickly & reliably

Table 4.7. Database models Status Today

Application Activity 4.7

In groups, do a research on Relational Database Management Systems. Present the report in class using presentation software.

END OF UNIT ASSESSMENT

1. What is the purpose of managing information?
2. Give the difference between File Processing System and Database Management System approach.
3. Discuss data independence and explain its importance in database environment.
4. Discuss the uses of databases in business environment.

UNIT 5
DATABASE DESIGN

UNIT 5: DATABASE DESIGN

INTRODUCTORY ACTIVITY

Consider the Company AMAHORO Ltd with departments like Accounting and Finance, Human Resource and ICT. Each department has its own employees who are each identified by Id, First Name, Surname, Salary, telephone number and Address.

Suppose that this Company uses the file approach to keep information of its employees, where each department has its own file to store necessary information. The payroll file is managed in Accounting and Finance departments and a record of an employee's salary is described by Employee Id, First name, Surname and Salary Amount.

Assume that a new column for the date of recruitment is to be added to the payroll and there is a request in Accounting and Finance departments to retrieve all the employees' details whose salaries are greater than 100,000 FRW. One employee finds that his/her first name was not correctly written on the payroll.

Since each department has own separate file, it is difficulty to the Human Resource Manager to access all the files and make any modification any time he/she wants.

1. Discuss whether the approach described above is well designed?
2. Describe the difficulties each department may encounter?
3. Discuss and Suggest the CEO of this Company the best and efficient approach to be used for overcoming all difficulties.

5.1 INTRODUCTION

Sometimes ago, organizations usually stored information in a way which was with many risks of being lost, damaged, corrupted and not well organized. It was kept inTraditional File Processing (TFP) which presented many disadvantages.

Nowadays, well-designed database arises to overcome those problems and saves the time in the long run for the user. It is facilitated by a Database Management Systems (DBMS). The design of such database presents 3 main levels called "Database design levels". These levels help to make easy the understanding and the management of a database. Later on, it will not be very difficult for the software developer to use that database.

ACTIVITY 5.1

Read the following scenario and give answers to the asked questions

Your school needs to keep information securely of its teachers, students, combinations and courses. Each teacher is characterized by its identity number, first name, second name, salary, qualification, address, telephone number and email.

The Combination is identified by its Id and Name. The student identified by its Id, First Name, Surname, Address, School fees, Contact Number, Combination Id, Class and Course is identified by the Course Id, Title, Combination,

From the above,

1. Discuss the data types for the attributes identifying entity Teacher, Combination, Student and Subject.
2. Describe the relationships between entities: Teacher, Combination, Student and Subject?

5.1.1 The conceptual level

The conceptual levelis concerned with concept(abstract), an idea of what something is or how it works; something formed in the mind; a mental image.

There exist different models used for the conceptual level to represent all the data elements likely to belong to the database but the most used is the Entity- Relationship Model (ERM) It uses the main concepts like entities, attributes and relationships.

An entity represents a real-world object such as an **employee** or a **project** and has attributes that represent properties such as EmpId, FirstName, Surname, Address and Birthdate. A relationship represents an association or a link among entities.

For example, when an employee works on many projects, a relationship exists between the employee and each project.

If an employee is identified by EmpId, FirstName, Surname, Address and Birthdate while a project is identified by its ProjectId and ProjectName, the diagram representing the Entity Relationship is the following.

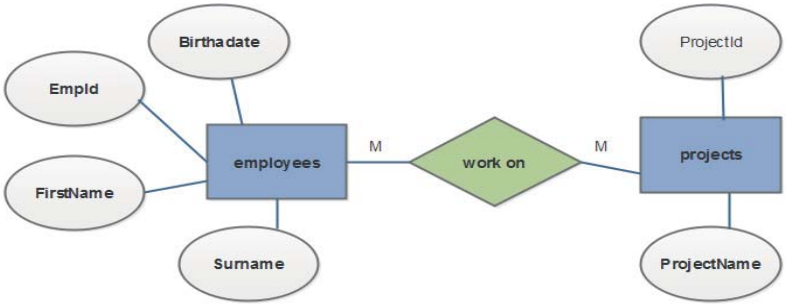


Figure 5.1 Entity Relationship model having employees and projects as entities

5.1.2 The logical level

The logical level makes it possible to create relational structures enabling us to put into practice the conceptualization by imagining a relational Database Management System (DBMS). It is characterized by clear and sound reasoning. At this level there exist different models like network data models and hierarchical data models and relational model. Currently the mostly used is the Relational model. In the case of database having Employee and Project entities the relational model is represented by the following tables.

employees relation

EmpId	FirstName	Surname	Address	Birthdate

projects relation

ProjectID	ProjectName

Figure 5.2 Logical level for the database having employee and project relations

5.1.3 Physical level

The **physical level** is concerned with how data will be encoded and stored. It consists of the practical application (Database Management Systems - DBMS) of all the preceding theories by using computers together with its software to create a database. It deals with storage and processing performance, volumetric (time & space), partitioning and distribution.

The most known development tools to use are MS Access, MySQL (My Structured Query Language), SQL (Structured Query Language) and NoSQL.

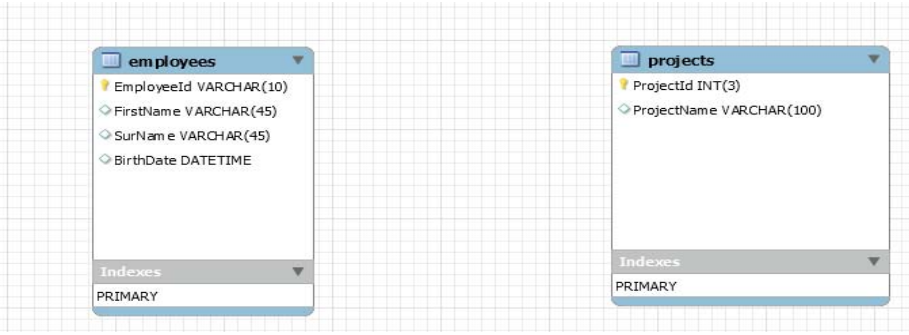


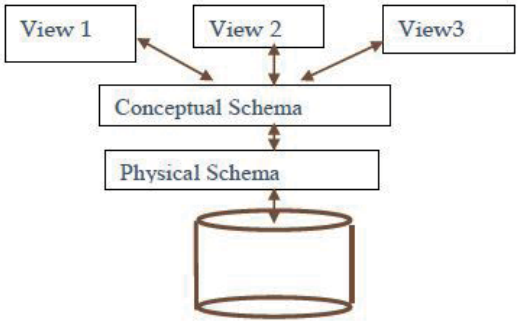
Figure 5.3 Physical level showing how data is encoded and stored

Application Activity 5.1

Galleries keep information about artists, their identification, their names, birth address, age, and art style. For each piece of artwork/product, the artist, the production year, its title, its type of art (e.g., painting, filmmaking, photograph, crafts, sculpture, drawings, etc), and its price in FRW must be stored. Pieces of artwork are also classified into groups of various kinds, for example, Rwandan art, 19th-century Art(old testament, the French art,Haloween,etc), works by Pablo Picasso; a given piece may belong to more than one group.Each group is described by an id and name.

Finally, galleries keep information about customers. For each customer, galleries keep that person's unique identification, full name, and address, total amount of FRW spent in the gallery, the artists and groups of art that the customers tend to appreciate.

Draw conceptual, logical and physical levels for the above described database.



Fugure 5.4. Data base access level

Example: SCHOOLS Database

- A View for registrar office

Course info (coursid:string,enrollment:integer)

- The conceptual schema:

Students(studid: string, name: string, login:string, age: integer, gpa:real)

Courses(coursid: string, ccoursname:string, courcredits:integer)

Enrolled(sstudid:string, coursid:string, grade:string)

- The physical schema:

Relations stored as unordered files.

Index on first column of Students.

5.2. DATABASE DESIGN STEPS

Learning Activity 5.2.

Read the following scenario and give answers to the asked questions

At Groupe Scolaire Kinazi, the management of students' information is not computerized. The Headteacher is looking for a solution so that the database of students can be managed digitally and hence speed up the production of transcripts and the search of information when they are requested for. If you are given the task to design that database, how are you going to proceed?

1. Discuss the different steps to use so that you can succeed the assignment.
2. Discuss the financial savings made by the school when using computerized application instead of manual system.

The most important thing to do to start designing a database is to think ahead.

When a case that needs a database creation is presented, before to switch on the computer, it is better to think about the type of information to work with and the types of questions that the database should answer, what information needs to be stored and what specifically are the links between them. The next phases are to verify all requirements specifications, to represent the data with diagram, and to plan the database, here are three rules of database design: **Rule 1-Plan; Rule 2-Plan; Rule3-Plan.**

When data is more complex, there are more needs to plan. Even the simplest database should be thought through on paper before being created in any tool such as Microsoft Access, SQL, MySQL, etc.

A well-designed database performs well and adapts to future needs by giving users access to essential information. Poor planning often results in a database that fails

to meet overlooked needs.

In planning the database, regardless of its size and complexity the following basic steps are used:

1. Investigate the information.
2. Identify the objects.
3. Model the objects.
4. Identify the types of information for each object.
5. Identify the relationships between objects.
6. Database optimization through normalization.
7. Data entry and manipulation

5.2.1. Investigate information

Before creating a database, there is a need of good understanding of the problem that the database is expected to solve. If the database is to replace the traditional method, file based approach method, then the existing system will give most of the information needed.

During investing information, there is a need to work with everyone involved in the existing system to see what is needed from the new database. Gathering techniques include collect copies of customer information, management reports, and any other documents that are part of the existing system, because these will be useful in designing the database and the interfaces.

5.2.2. Identifying the important entities and their attributes

During the process of gathering/investigating information, the key objects or entities that will be managed by the database must be identified. The object can be a tangible thing, such as a person (for example student, employee, and patient) or a product, or it can be a more intangible item, such as a *department* in an institution, a *Combination* in a school. Each distinct item in the database should have a corresponding table for which column titles are attributes of the entity

5.2.3. Identifying the Relationship Between entities

One of the strengths of an E-R database is the ability to relate or associate information about various items in the database.

Isolated types of information can be stored separately, but the database can combine data when it is required. Identifying the relationship between entities in the design process requires looking at the entities, determining how they are logically related, and adding relational columns that establish a link from one table to another.

5.2.4. Modeling the objects

As the objects in the system and their attributes are identified, they are recorded in a way that represents the system visually. They are recorded using *Relational model* and *Entity R*elational *model*

5.2.5. Identifying the types of information for each object

After identifying the primary objects/entities in the database as candidates for tables, the next step is to identify the types of information that must be stored for each object.

These are the columns in the table of the object.

Fields/columns should be kept simple, the more atomic your fields the more flexible will be your database.

For example, in a database of **names and addresses**, you would keep each part of the person's name as a separate field.

The columns in a database table contain a few common types of information:

Raw data columns

These columns store tangible pieces of information, such as names.

Categorical columns

These columns group the data and store a limited selection of data such as true/false, married/single/divorced/widowed, Male/Female, etc.

Identifier columns

These columns provide a mechanism to identify each item stored in the database table. These columns frequently have an id or number in their name, for example, EmployeeId, StudentId, PersonIdNo and InvoiceNumber. The identifier column is the primary component.

Relational or referential columns

These columns establish a link between information in one entity and related information in another entity.

For example, an entity that tracks sales transactions will generally have a link to the customer's entity so that the complete customer information can be associated with the sales transaction.

5.2.6. Database optimization through normalization

One of the most important step to consider when designing a database is database definition. If tables are not set up properly, it can cause a lot of headaches down the road the time of extracting/retrieving required data. Understanding the rules of normalization enforces redundancy elimination and inconsistent dependency in database designs.

5.2.7. Data entry and manipulation

The goal of data entry is to create data that are valid and well organized to assure their quality during extraction. Well stored data leads to data consistency.

Application activity 5.2.

Discuss the benefits of following database design steps?

5.3. RELATIONAL MODEL

Learning Activity 5.3.

Read the following scenario and give answers to the asked questions

Suppose that School wants to develop an information system in which student studies in one of the combination and learns different subjects. The system will record details concerning student, combination and subject.

1. Which properties would you expect to find in each student object?
2. Arrange those properties in tabular form of mxn dimension

5.3.1. Introduction

The relational data model was introduced by C. F. Codd in 1970. Currently, it is the most widely used data model. It describes the world as "a collection of inter-related relations (or tables)."

a. Relation

A relation, also known as a table or file, is a subset of the Cartesian product of a list of domains characterized by a name. You can also think of it this way: an attribute is used to define the record and a record contains a set of attributes.

The steps below outline the logic between a relation and its domains.

1. Given *n* domains are denoted by *D1, D2 Dn*
2. And **r** is a relation defined on these domains
3. hen r C *D1*×*D2*×…×*Dn*

The following are the key component to know when we are talking about relation:

Relation, Tuple, Attribute, Cardinality, Degree, Primary key, Domain

b. Equivalent Database Concepts

- Relation <=> Table
- Tuple <=> Row or record
- Attribute <=> Column or field
- Cardinality <=> Number of rows
- Degree <=> Number of columns
- Primary key <=> Unique identifier
- Domain <=> Pool of legal values

c. Table

Data is stored in rows and columns. Each Row is known as record and the data items are known as fields. Tables contain data about one type of item, person or event, for example:

- a table of patients, a table of a student, a table of teacher, a table of books and a table of doctor's appointment

For a book the fields could include:

- Title, Author, ISBN, Publication house

Column

A database stores pieces of information or facts in an organized way. The principal storage units are called columns or fields or attributes. These house the basic components of data into which your content can be broken down.

Records

Each row represents a group of related data values, such as a customer or an employee. A row, or record, is also known as a tuple

Cell

The link/intersection between column and record is known as cell. It also needs to be available so that they can be reconstituted into their whole form, the basis of all databases.

A simple table below gives us the clearest picture of how records and fields work together in a database storage project.

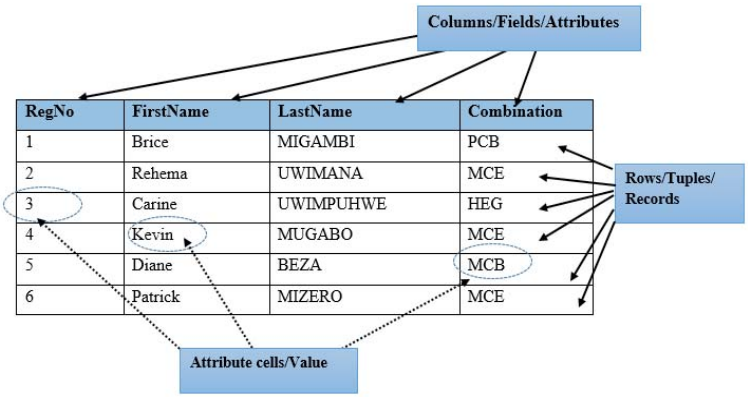


Figure 5.5. Record, Value and Fields.

d. Null value

In many situations every row and column will contain data, but there cases where it makes sense for some columns to not contain a value.

In our example StudentId cannot be null because it is unique but Sex can be null because it is optional field

e. Degree

The degree is the number of attributes in a table. In our example above in Figure 5.4 the degree is **6**.

f. Domain

A domain is the original sets of atomic values used to model data. A domain is a set of acceptable values that a column is allowed to contain.

For example:

- The domain of Marital Status has a set of possibilities: Married, Single, Divorced, Widowed.
- The domain of Shift has the set of all possible days: {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday}.
- The domain of Salary is the set of all floating-point numbers greater than 0 and less than 200,000 FRW.
- The domain of First Name and Surname is the set of character strings that represents names of people.

g. Datatype

Each field in a record has its own data type. The data types in the fields can be text, alphanumeric, numeric, and Boolean or date/time.

Fieldname	Data Type	Size	Reason for Choice
StudentId	Numeric or alphanumeric	4	Numeric- can be sorted on numerical order Alphanumeric-can be given a fixed length-Not used for calculation can include other characters besides digits for example +,()
FullName	**Text/char**	**40**	**Can be sorted in alphabetical order**
Admission_ Date	Date		Automated checking provided by most database management systems, for example the month must have values from 1 to 12
Combination	Text/char	4	Can be sorted in alphabetical order
Sex	Boolean Or text	6	Boolean-only two choices M or F Text-can have validation rules set for a single character restricted to the values M or F

Size: Maximum number of characters you can enter and you expect in the field.

Table 5.1: Datatype example

Application Activity 5.3.

1. Suppose that a hospital wants to develop system which manages doctors and patients.

Patients are treated in a ward by the doctors assigned to them. Each patient will be assigned a single doctor. What fields would you expect to find in each record for a hospital patient and doctor

2. The following table describes information about employees, study it and answer the following questions :

EmpId	EmpSname	EmpFname	EmpLocation	EmpJobCode
1011	RUKUNDO	Jean Claude	Kigali	21
1013	NYIRAMANA	Sylvane	Huye	19
1014	UWIMANA	Yvette	Muhanga	29
1021	KAGABO	Peter	Musanze	13

Table 5.2. : employees table

a. Using correct terminology, identify and describe all the components of employee table.
b. What is the possible domain for field EmpId?
c. How many records are shown?
d. How many attributes are shown?
e. Explain the datatype for each field.

5.3.2 Queries in design view

Learning activity 5.4.

Microsoft Access is a database management system from Microsoft that deals with relational model and it is consisted of the following different views:

a. Layout view
b. Backstage view
c. Design view
d. Datasheet view

Which of the view above contains command buttons that execute operations on entire databases? Explain its advantages.

Data in a Microsoft Access database is stored in various interlinked tables. A database also has forms for data entry though you could enter data directly in tables and queries for manipulating your data and permitting information to be retrieved from a table. It also allows filtering so only the required records and fields are seen.

Steps for creating database in Microsoft Access

- Open Microsoft Access
- Click on Blank database in the task pane
- Name your database file (say Company) and click the Create button to create the database.

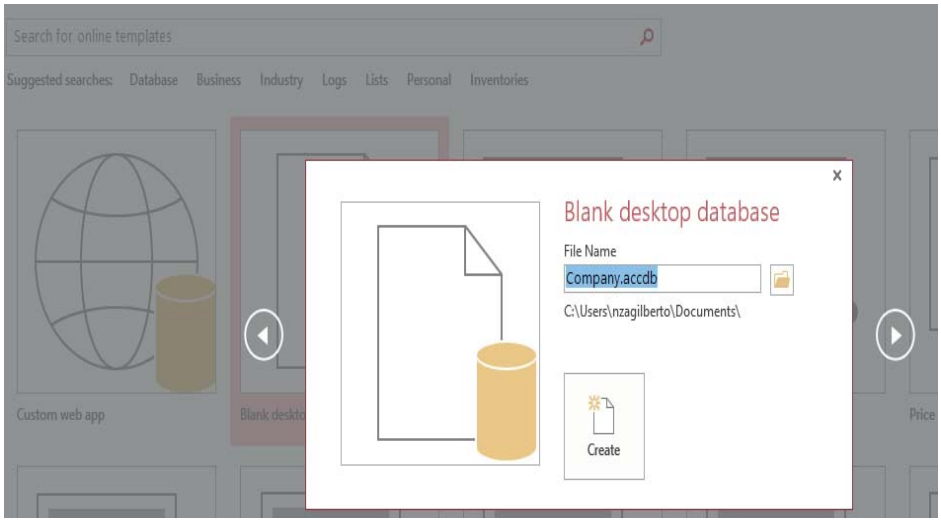


Figure 5. 6. Naming a database in MS Access

Steps for creating a table in Microsoft Access-Design View

1. Open an created database in access DBMS
2. Click on create tab from menu bar
3. In table group click on table design icon
4. Fill field names, data type and their descriptions

Data in an Access table is stored in various fields. Those fields have different properties such as fieldname, size and data types as shown in the figure below:

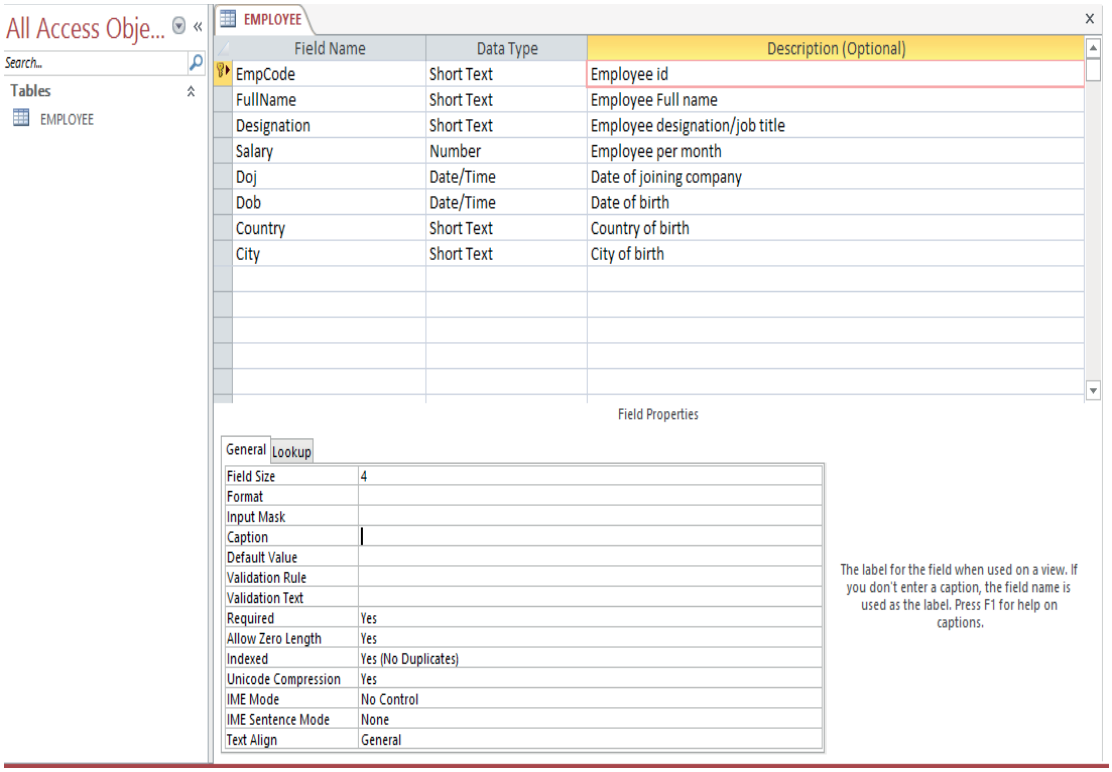


Figure 5.7. Field's properties

Steps for creating relationship between created tables

1. Open a created database in access DBMS
2. Click on database tools tab from menu bar
3. In relationships group click on relationships icon then the relationships panel will be displayed
4. Select a table name and click on add button
5. Repeat step 4 to all table you want to use
6. Click on design tab from menu bar
7. In tools panel click on edit relationship icon

8. On opened window click on create new button
9. Select the table names and their columns that you want to join
10. Click on ok button
11. Repeat 8, 9, and 10 to all pair table to be joined

After creating a table, enter the data. Note that there are various methods to create tables in Access, such as by using Table Wizard, Datasheet view or by importing tables, but this Unit restricts only to Design view.

Employee							
EmpId	FullName	Designation	Salary	Doj	Dob	Country	City
101	KAREKEZI Ange	Accountant	240000	3/23/2003	1/13/1980	RWANDA	HUYE
102	GANZA Kevin	Head-IT	320000	12/2/2010	7/22/1987	UGANDA	KAMPALA
103	MUTONI Rehma	Customer Care	200000	6/24/2009	2/24/1983	BURUNDI	BUJUMBURA
105	NKUSI Anicet	Marketting Officer	310000	11/8/2006	3/3/1984	BURUNDI	BUJUMBURA
108	KARENZI Abdul	CEO	560000	12/29/2004	1/19/1982	RWANDA	KIGALI

Table 5.2. Employee relation

Steps for creating a query using design view-Query by example

Creation of a query in design view has three rows, one to sort the fields, the other to specify whether or not to display a field, and the last to specify some criteria to select the records.

1. Click on Create tab of menu from bar
2. In queries grout click on query Design icon
3. Select tables to use then click on Add button
4. In Query Type group select a query commend to be used, Example Select
5. In query panel select the table names and field names you want to select in different tables
6. Check in show row ender the select item if you want to display them on output at run time
7. Rename the created query
8. In Result group of menu bar click on Run icon

After the above steps you will get a table containing the selected items as shown in the examples below:

Query 1: Company leader wants to display onlyEmpId, Fullname (sorted is ascending order) and salary for all employees.

1. Select EmpId, EmpId, FullName, Salary and check them

2. Select Sort by Ascending on FullName

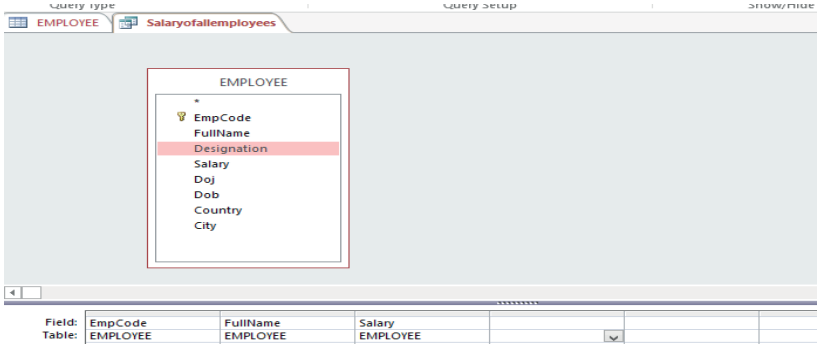


Figure 5.8. Query by example Grid displaying salary of all employees

Output:

EmpIdEmpId	FullName	Salary
102	GANZA Kevin	320000
101	KAREKEZI Ange	240000
108	KARENZI Abdul	560000
103	MUTONI Rehma	200000
105	NKUSI Anicet	310000

Table 5.3: Output for Salary of all employees

Query2: The CEO wants to see only EmpId and FullNames where Salary is less or equal to 240000.

Select EmpId, EmpId , FullName and check them

Set criteria and uncheck Salary field

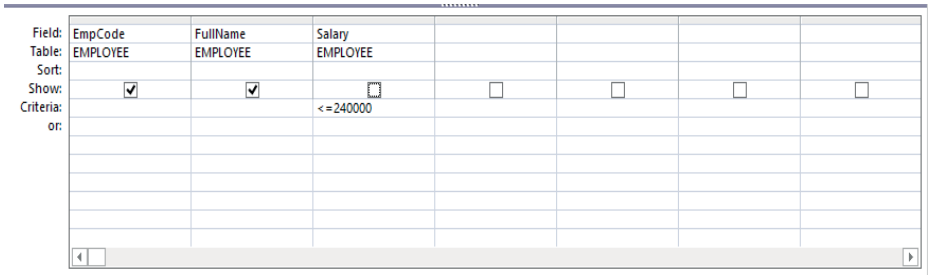


Figure 5.9 Query by example grid displaying EmpId and FullNames where Salary<=240000

Output:

EmpIdEmpId	FullName
101	KAREKEZI Ange
103	MUTONI Rehma

Table 5.4. Output for QBE displaying EmpId and FullNames where Salary<=240000

Query 3: The following search condition was entered using a query design view (Query by example) grid:

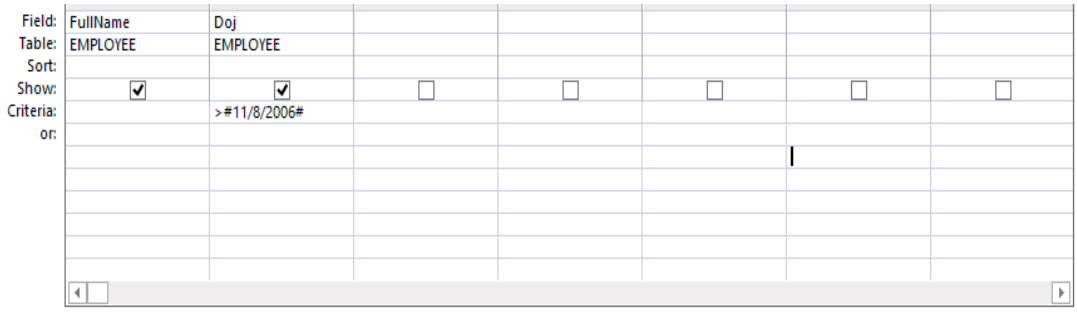


Figure 5.10 QBE grid displaying FullName and Doj where Doj>=11/8/2006

What will be displayed?

Output:

FullName	Doj
GANZA Kevin	12/2/2010
MUTONI Rehma	6/24/2009

Table 5.5 Output for QBE grid displaying FullName and Doj where Doj>=11/8/2006

5.3.3 Dynamic queries using parameter

A parameter is a piece of information you supply to a query right as you run it.

Creating a parameter query

Creating a parameter is similar to adding a normal criterion to a query:

1. Create a select query, and then open the query Design view.
2. In the Criteria row of the field you want to apply a parameter to, enter the text that you want to display in the parameter box, enclosed in square brackets.

   For example, [Enter the Doj:]

3. Repeat step 2 for each field you want to add parameters to.

When you run the query, the prompt appears without the square brackets.

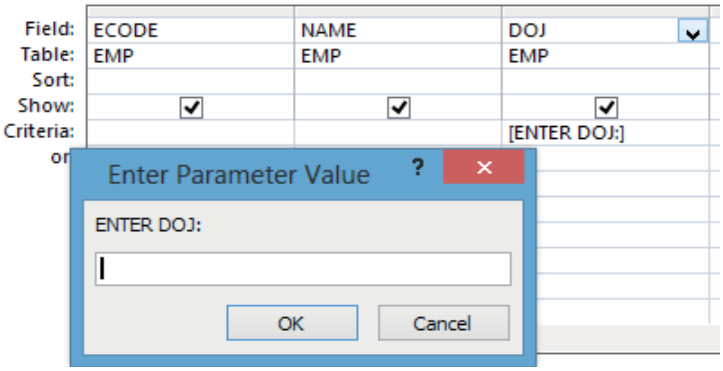


Figure 5.11. Prompt window for adding value

Fill in the value you're looking for, and then click OK.

Specify parameter data types

To specify the data type for parameters in a query:

1. With the query open in Design view, on the Design tab, in the Show/Hide group, click Parameters.
2. In the Query Parameters box, in the Parameter column, enter the prompt for each parameter you want to specify a data type for.

Make sure that each parameter matches the prompt that you used in the Criteria row of the query design grid.

3. In the Data Type column, select the data type for each parameter.

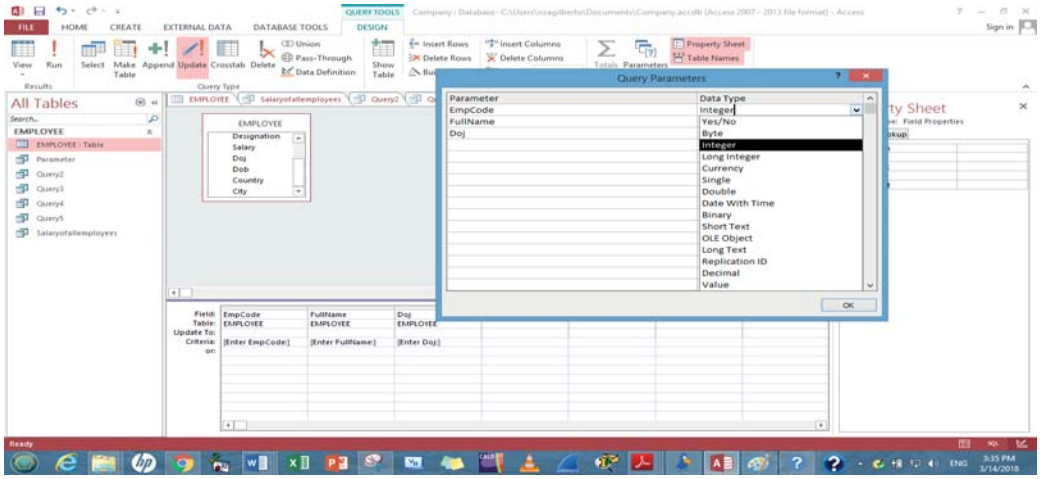


Figure 5.12. Parameter Data type

Application Activity 5.4.

The database WORLD is composed of one table called WORLDCITIES Figure 5.14. which stores information about World Cities. Use it to answer asked questions

1. The query-design view grid below selects all Cities from America where Population is greater than 8 million.

Field:	Name_of_city	Continent	Population
Table:	WorldCities	WorldCities	WorldCities
Sort:			
Show:	☑	☐	☐
Criteria:			
or:		="America"	>8

Figure 5.13 QBE grid to display Name of city located in America having population >8M

Show what would be the output.

2. Copy and Complete the query design view grid below to select and show Name_of_city where Population is greater than to 13 and Area is equal to 1590.

Field:				
Table:				
Sort:				
Show:				
Criteria:				
or:				

Figure 5.14 QBE to display Name_of_city where Population is 13 and Area is equal to 1590

WORLDCITIES						
CityId	Name_of_city	Country	Continent	Population	Density	Area
1	Shangai	China	Asia	13.9	7170	1930
2	Mumbai	India	Asia	13.8	22940	600
3	Karachi	Pakistan	Asia	13	3680	3530
4	Delhi	India	Asia	12.6	29150	430
5	Istanbul	Turkey	Eurasia	12.5	6210	1830
6	Sao Paulo	Brazil	America	11.2	7380	1520
7	Mosco	Russia	Europe	10.6	9770	1080
8	Tokyo	Japan	Asia	8.9	14400	620
9	New York	USA	America	8.4	10450	790
10	London	UK	Europe	7.8	4860	1580
11	Bogota	Colombia	America	7.3	4570	1590
12	Rio de Janeiro	Brazil	America	6.3	5350	1180

Figure 5.15. WORLDCITIES

5.4. ENTITY-RELATIONSHIP MODEL

5.4.1. Introduction

Learning Activity 5.5.

A School needs to store information about Teacher (identified by TeacherId, FirstName, Surname, Salary, Qualification, Address, Contact); Combination (identified by CombinationId, CombinationName),Student (identified by StudentId, FirstName, Surname, Address, Schoolfees, ContactNumber) and Subject (identified by subjectId, SubjectTitle).

Discribe the logical relationships between objects Teacher-Student, Teacher-Combination, Student-Combination and Teacher-Subject??

The entity relationship (ER) data model has existed for over 35 years. It is well suited to data modelling for use with databases because it is fairly abstract and is easy to discuss and explain. ER models are readily translated to relations. ER models, also called an ER schema, are represented by ER diagrams.

ER modelling is based on two concepts:

- Entities, defined as tables that hold specific information (data)
- Relationships, defined as the associations or interactions between entities

5.4.2. Entity Relationship Diagram

The ER diagram is used to represent the conceptual database schema. In ER diagram:

Entity, Attributes and Relationships form the components of ER Diagram and there are defined symbols and shapes to represent each one of them.

1. Entity

A rectangle represents an entity set.

Figure 5.16: Entity sets

2. Relationships between Entities

A diamond represents a relationship.

Figure 5.17.: Relationship between Entities

3. Entity Types

- Entity types ->boxes
- Weak entity type -> double box

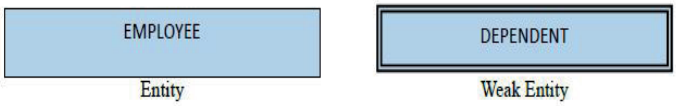


Figure 5.18 Entity Types

Entity types are similar to classes; they describe potential objects (entities) that will appear in the database. Weak entity types describe dependent entities, entities that depend on other entities for identity.

4. Attribute

An ellipse represents an attribute (Property).

Figure 5.19. Attributes

5. Link between attribute and entity set

Lines represent linking of attributes (properties) to entity sets.

Figure 5.20.: Link between Entities set and attributes

5.4.4. Entities and entity sets

An entity set is a set of entities of the same type that share the same properties. A noun is used to represent an entity set. An entity is an instance of an entity set.

For example, an entity can be:

- Concrete (TEACHER or STUDENT)
- Insubstantial (GRADE)
- An occurrence (EXAM)

5.4.5. Attributes

A characteristic of an entity, for example First name, Last name and Age. An attribute is a data item that describes a property of an entity set. Attributes determine, explain, or categorize an entity set. Attributes have values, which can be of different data types such as numbers, character strings, dates, images, sounds, and so on. In a physical model, an attribute is a named column in a table. Each table has a list of attributes (or columns).

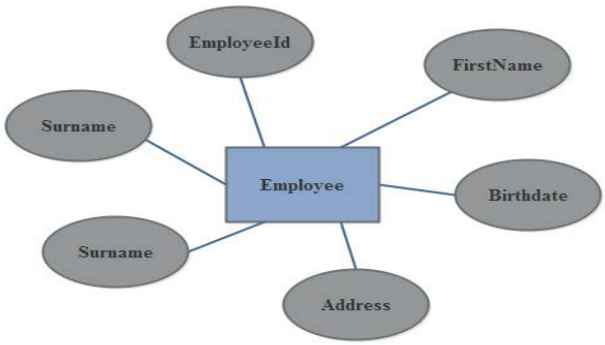


Figure 5.21: How attributes are represented in an ERD.

The types of attributes are:

Simple (atomic) attribute – This type of attribute has a single component. It is called single-valued attribute. For example, the Gender attribute has a single component with two values.

In the COMPANY database, an example of this would be: Name = {John} ; Age = {23}
Composite attribute – A composite attribute consists of many components.

For example, the Name attribute has the components Last name and First name. So this would be written as → Name = {KAGABO+Peter}

Address may consist of Province and District, Number, Avenue. So this would be written as → Address = {KG +'14' + 'Av'}

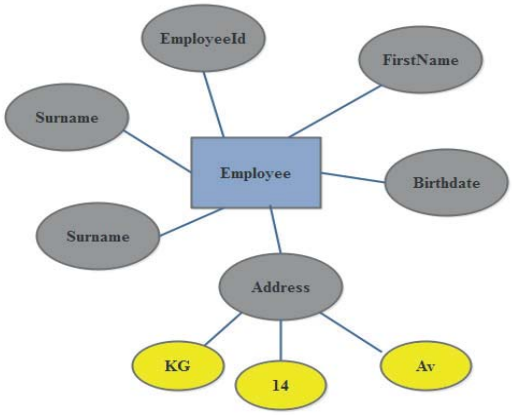


Figure 5. 22. Composite Attribute

Single valued attribute – This type of attribute has one value for one entity. For example, the Title attribute has a single value for each teacher.

Multi-valued attribute – A multi-valued attribute has many values for one entity.

An example of a multivalued attribute from the COMPANY database, as seen in Figure below, are the degrees of an employee: BSc, MSc., PhD.

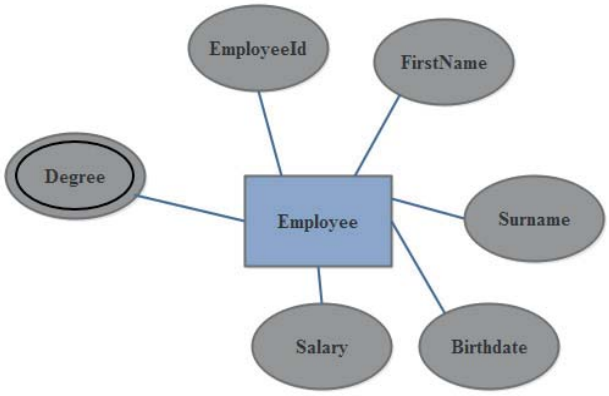


Figure 5. 23. Multi valued attribute

Derived attribute – A derived attribute has its value computed from another attribute or attributes.

A derived attribute is not a part of a table from a database, but is shown for clarity or included for design purposes even though it adds no semantic information; it also provides clues for application programmers.

An example of this can be seen in Figure below. Age can be derived from the attribute Birthdate

In this situation, Birthdate is called a stored attribute, which is physically saved to the database.

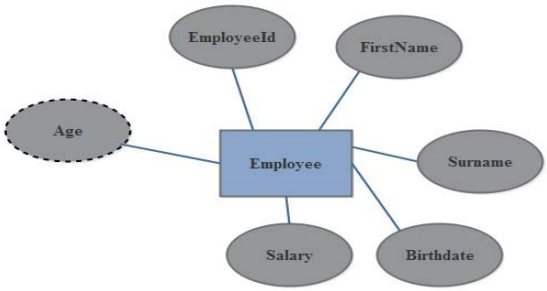


Figure 5.24. Derived attribute

Unstable attributes - This type of attribute have values that always change. For example, the salary of employee.

Mandatory attributes - Mandatory attributes must have a value. For example, in most businesses that track personal information, Name is required.

Optional attributes - Optional attributes may have a value or be left null; For example, address of employee

Unique identifier - This type of attribute distinguishes one entity from another.

For example, in a company, you can distinguish between one employee and another using a EmployeeID.

5.4.6. Keys

An important constraint on an entity is the key. The key is an attribute or a group of attributes whose values can be used to uniquely identify an individual entity in an entity set. In the case of **Logical Model** you use a special approach for unique identifier.The equivalent concept for unique identifier within Logical Model is a key.

A key is a field or a set of fields that has/have a unique value for each record in the relation. You need a key to ensure that you do not meet redundancies within a relation There are several types of keys each having slightly different characteristics:

Super key-is defined as a set of attributes within a table that can uniquely identify each record within a table. Super Key is a superset of Candidate key.

Candidate key – A candidate key is an attribute or set of attributes that uniquely identifies a record in a relation. A candidate key is unique and minimal.

It is unique because no two rows in a table may have the same value at any time. It is minimal because every column is necessary in order to attain uniqueness.

From our Company database example, if the entity is Employee(SerialNo,EmpId, First Name, Surname, Address, Contact, Birthdate, Salary, Depid), possible candidate keys are:

- ***EmpId,***
- ***SerialNo***
- ***FirstName and Surname*** – assuming there is no one else in the company with the same full name
- ***Surname and Depid*** – assuming two people with the same Surname don't work in the same

Department

Composite keys – these keys have multiple attributes.

Using the example from the candidate key section, possible composite keys are:

- First Name and Surname
- Surname and Depid

Primary key – A primary key is one of the candidate keys from a relation. Every relation must have a primary key. A primary key shall be at least: - ***Stable.*** The value of a primary key must not change or become null throughout the life of the entity.

The primary key is indicated in the ER model by underlining the attribute.

For example, consider an employee record; using the Age field as the primary key would not be appropriate because the value of the primary key must not change over time.

It should be Minimal. The primary key should be composed of the minimum number of fields to ensure the occurrences are unique.

In this case Employee (EmpId, First Name, Surname, Address, Contact, Birthdate, Salary, Depid), EmpId is primary key

Secondary key

A secondary key is an attribute used strictly for retrieval purposes (can be composite), for example: Phone and Surname.

Alternate key – An alternate key is any candidate key that is not chosen to be the

primary key. It may become the primary key if the selected primary key is not appropriate.

Simple keys – these keys have a single attribute.

Foreignkeys – these keys exist usually when there are two or more relations. An attribute from one relation has to exist in the other(s) relation. A foreign key is a field (or fields) that points to the primary key of another table. The purpose of the foreign key is to ensure referential integrity of the data. In other words, only values that are supposed to appear in the database are permitted
In the Company database example below, Depid is the foreign key
(EmpId, First Name, Surname, Address, Contact, Birthdate, Salary, Depid)

Attributes and Keys

- Key attributes must be unique for each entity
- Keys are used to identify particular entities
- Partial keys are only partially unique used for weak entity types

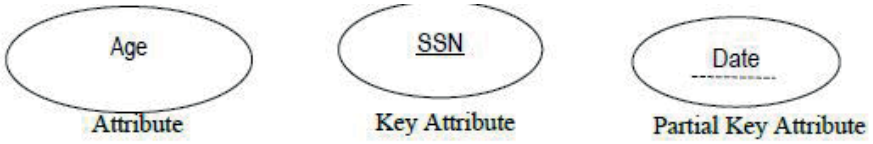


Figure 5. 25. Attributes and Keys

Entity Types and Attributes

All regular entity types must have a key attribute or set of key attributes, Weak entity types must have partial keys, Weak entities get part of their key (and part of their identity) from some related entity.

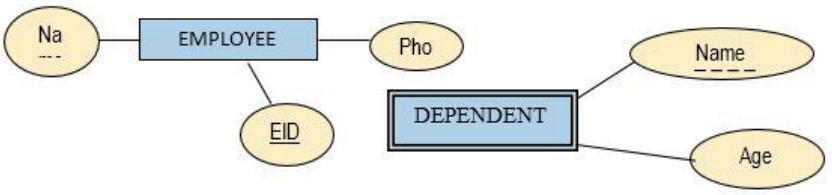


Figure 5. 26. example of Entity Types and Attributes

Nulls

A null is a special symbol, independent of data type, which means either unknown or inapplicable. It does not mean zero or blank.

Features of null include:

- No data entry
- Not permitted in the primary key
- Should be avoided in other attributes
- It can represent

an unknown attribute value

1. a known, but missing, attribute value
2. a "not applicable" condition

Example of null

Employee Salary table

EmpId	JobTitle	Department	Salary	Commission
1012	Sales Manager	Sales	400,000	43000
1013	Logistic Officer	Sales	350,000	null
1014	Director General	Administration	1,000,000	null

Table 5.6. Employee Salary

In this case commission can be left null

Relationship sets

A relationship set is a set of relationships between two or more sets of entities, and are regularly represented using a verb.

A relationship is an instance of a relationship set and establishes an association between entities that are related. These relationships represent something important in the model. Example: an employee works on a project.

A relationship set always exists between two entity sets (or one entity set relating to itself). You need to read a relationship set in double sense, from one entity set to the other.

Relationships representation

Relationships => diamonds

Identifying relationship => double diamond

Figure 5.27. attribute type

Relationships indicate a meaningful connection between two entity types, Relationships may have attributes, but they cannot have key attributes. Identifying relationships connect a weak entity type to some other entity type indicates where the weak entity gets a key to complete its own partial key

5.4.7. Degree of a relationship

The degree of a relationship refers to the number of associated entities. The degree of a relationship can broadly be classified into ***unary, binary, and ternary relationship***.

5.4.7.1. Unary Relationship

The unary relationship is known as recursive relationship.

In the unary relationship the number of associated entities is one. An entity is related to itself is known as recursive relation.

An example is shown by the following figure:

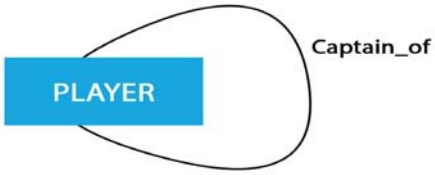


Figure 5.28. Recursive relation

Roles and Recursive Relations

When an entity set appears in more than one relationship, it is useful to add labels to the connecting lines. These labels are called roles.

In this example, labels "Husband" and "wife" are referred to as roles.

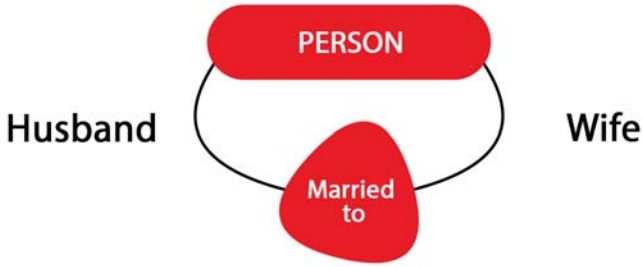


Figure 5.29. Wife-Husband relationships

5.4.7.2 Binary Relationship

In a binary relationship, two entities are involved. Consider the example where each staff will be assigned to a particular department. Here the two entities are STAFF and DEPARTMENT.

Figure 5.30 Staff-Department binary relationships

5.4.7.3. Ternary Relationship

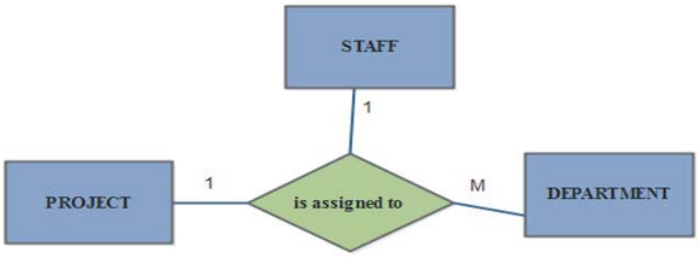


Figure 5. 31. Ternary Relationship

In a ternary relationship, three entities are simultaneously involved. Ternary relationships are required when binary relationships are not sufficient to accurately describe the semantics of an association among three entities.

Example

Consider the example of employee assigned a project.

Here we are considering three entities EMPLOYEE, PROJECT, and LOCATION. The relationship is "assigned-to." Many employees will be assigned to one project hence it is an example of one-to-many relationship.

5.4.7.4. Quaternary Relationships

Quaternary relationships involve four entities. The example of quaternary relationship is "A professor teaches a course to students using slides." Here the four entities are PROFESSOR, SLIDES, COURSE, and STUDENT.

The relationships between the entities are "Teaches."

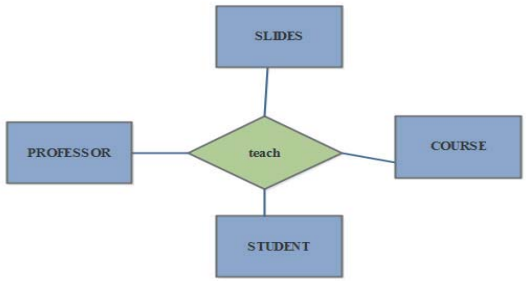


Figure 5.32. SLIDES-PROFESSOR-STUDENT-COURSE relationship

The degree of relationship, which determines how many instances of an entity relate to a single instance of another entity is called Cardinality.

5.4.8. Classifications or types of Relationships

Based on cardinality, the different types of relationships that can exist between entities are:

5.4.8.1. One-to one (1:1)

A single instance of an entity can relate to only one instance of the other entity.It is the relationship of one entity to only one other entity, and vice versa. It should be rare in any relational database design. In fact, it could indicate that two entities actually belong in the same table. An example from our Company database is one employee is associated with one spouse, and one spouse is associated with one employee.

Other Examples:

- A person can have only one passport.
- The relationship between the President and the country is an example of one-to-one relationship. For a particular country at a given time, there will be only one President. In general, a country will not have more than one President at a given time hence the relationship between the country and the President is an example of one-to-one relationship.

5.4.8.2. One-to-many (1: M)

An instance of one entity can relate to multiple instance of another instance.

The relationship that associates one record of entity A to more than one record of entity B is called one-to-many relationship.

Example of one-to-many relationship is a country having states.

For one country there can be more than one state hence it is an example of one-to-many relationship. Another example of one-to-many relationship is parent–child relationship. For one parent there can be more than one child. Hence it is an example of one-to-many relationship.

For Example,

- A class has many students.
- The relationship between EMPLOYEE and DEPARTMENT is an example of many-to-one relationship. There may be many EMPLOYEES working in one DEPARTMENT. Hence relationship between EMPLOYEE and DEPARTMENT is many-to-one relationship.

5.4.8.3. Many-to-One Relationship Type (M: 1)

The relationship between EMPLOYEE and DEPARTMENT is an example of many-to-one relationship. There may be many EMPLOYEES working in one DEPARTMENT. Hence relationship between EMPLOYEE and DEPARTMENT is many-to-one relationship

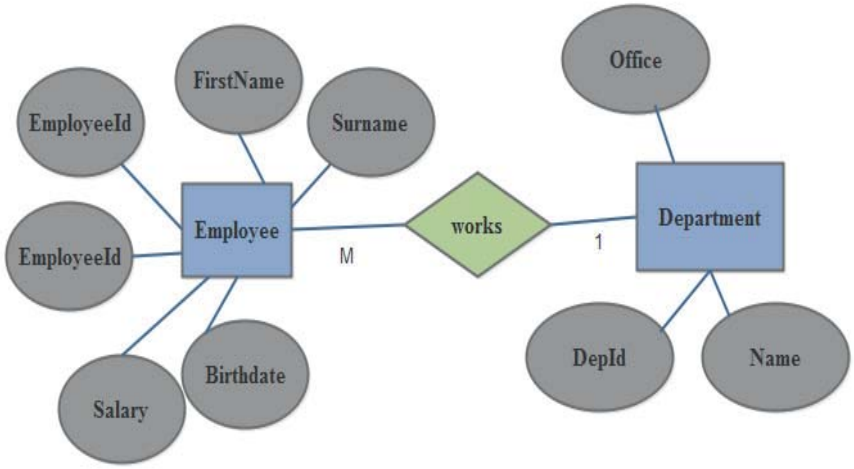


Figure 5.33. Many to one relationship

5.4.8.4. Many-to-many (M: M):

Multiple instances of an entity can relate to multiple instances of another entity.

For our company database. The relationship between employee and department is many-to-many relationship.

Other examples are:

- A customer can purchase more than one book.
- The relationship between TEACHER entity and STUDENT entity is an example of many-to-many relationship. Many one teacher teaches many students and one student is taught by many teachers.

Employee and project, many employees can work on many projects

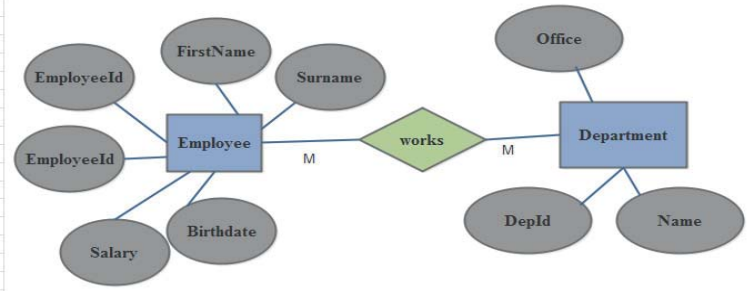


Figure 5.34 Many to Many relationships

Participation and Cardinality

Participation and cardinality define constraints on relationships; Participation indicates whether an entity is required to take part in a relationship, Cardinality ratios and structural constraints place limits on the number of entities that may participate in a relationship.

Participation Constraints

- Total participation → double or thick line indicates required participation
- Partial participation → thin line indicates optional participation

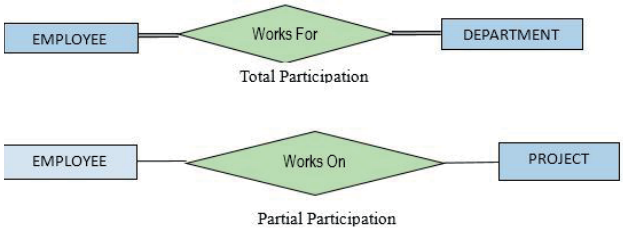


Figure 5.35. Total And Partial Participation

Participation Constraint

Arrowheads can be used to indicate an upper bound of 1 for participation

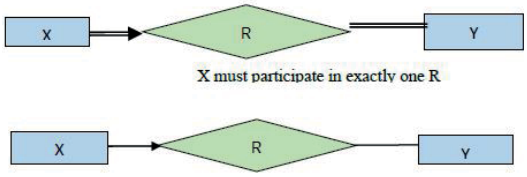


Figure 5. 36. Participation Constraint

Cardinality Ratios: a cardinality ratios specify the maximum number of relationship instances that an entity may participate in.

Cardinality Ratios: a cardinality ratios specify the maximum number of relationship instances that an entity may participate in.

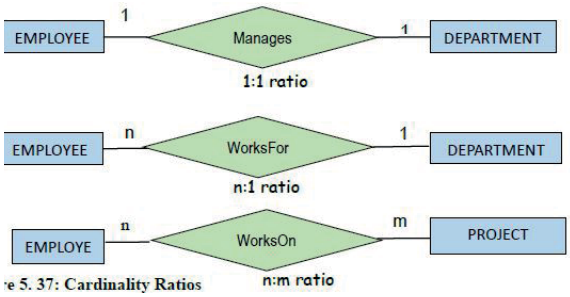
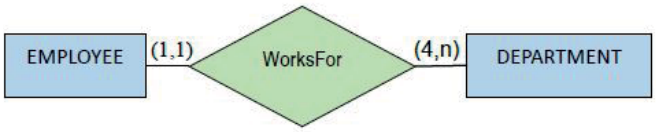


Figure 5. 37: Cardinality Ratios

Structural Constraints

Structural constraints specify the ***minimum*** and ***maximum*** number of relationship instances that an entity may participate in.



An employee must work for exactly 1 department. A department must have at least 4 employees.

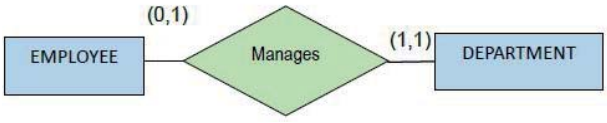


Figure 5. 38.: Structural Constraints

Participation and Cardinality

There's generally numerous ways to express a relationship constraint.

———	(0, n)	Optional participation in any number of relationships
——→	(0, 1)	Optional participation in at most one relationship
═══	(1, n)	Required participation in at least one relationship
═══→	(1, 1)	Required participation in exactly one relationship

Equivalent Notations

An employee can manage at most one department.

A department must have exactly one manager.

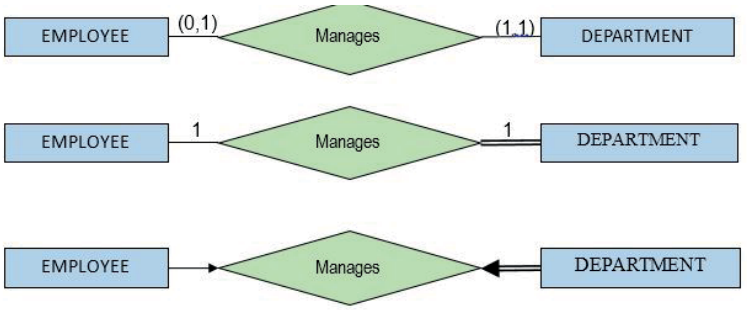


Figure 5. 39.:Equivalent Notations

Equivalent Notations

- An employee must work for exactly one department.
- A department must have at least one employee.

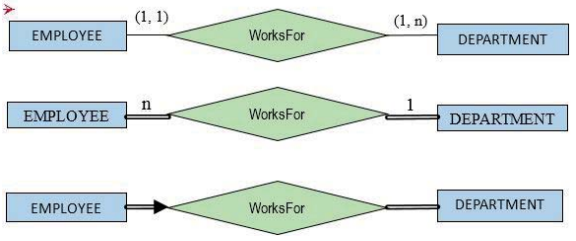


Figure 5. 40: Notation Summary

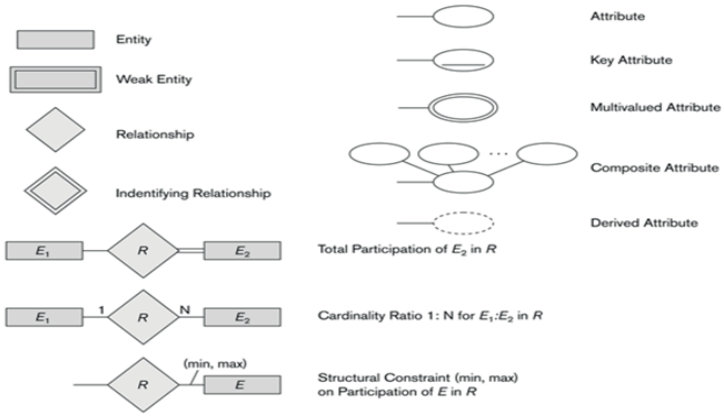


Figure 5.41 :ERD Example 1

Application Activity 5.5

Given an ER diagrams below

Write the degree, cardinality, participation and Constraints of each relationship in ER diagram

Write the names of all: weak entities, composite attributes, multivalued attributes, composite keys and Partial Key Attribute, derivative attribute, identity relationship.

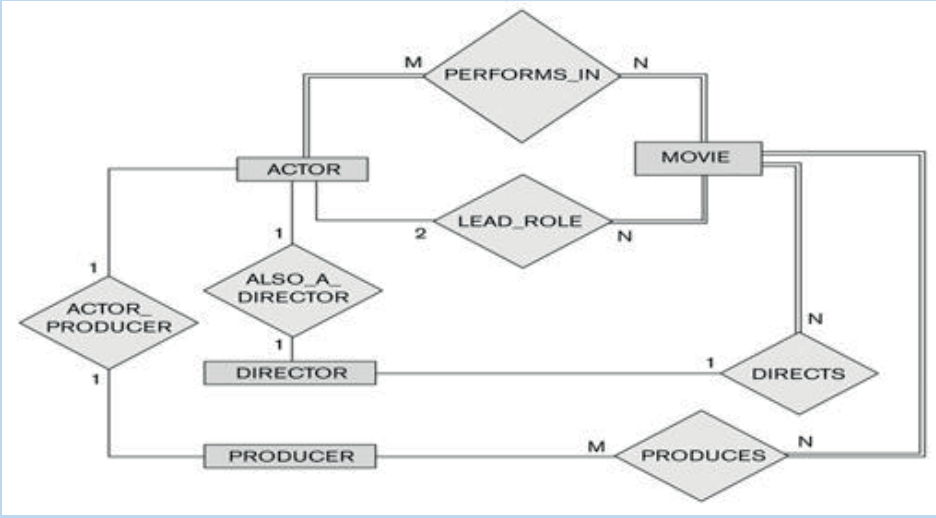


Figure 5.42 :ERD Example 2

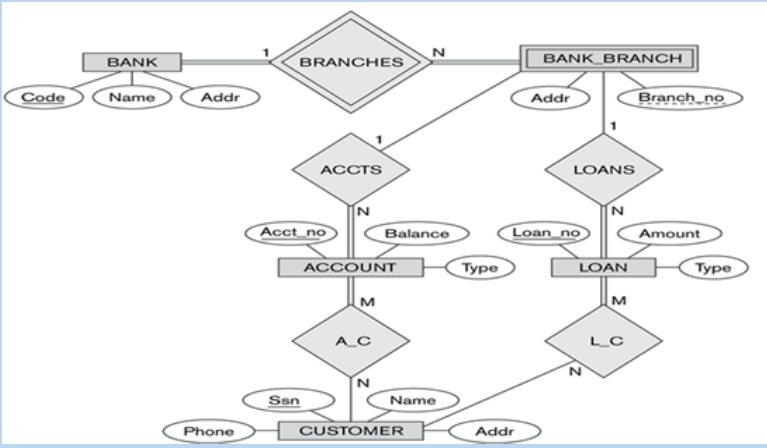


Figure 5.42: ERD example 3

Relationships representation

1. Consider a database used to record information about the marks that students get in the different exams of different subject studying in Math-Computer Science-Economics combination.

STUDENT entity is described by StudentId, StudentName and School; SUBJECT_STUDYING is identified by SSId, SSName, SSClass and SSRoom and EXAM entity is described by EId, EName, ETime and ERoom.

Construct an E-R diagram that models STUDENT, SUBJECT_STUDYING and EXAM as entities that uses ternary relationship.

5.5. Database optimization through normalization

Learning Activity 5.6.

Let consider the database with table CUSTOMER identified by CustomerName, Address, TelephoneNumber, CreditLimit, ItemOrdered,Quantity and Price.

This can easily be implemented in a relational database as follow:

Customer Name	Address	Telephone Number	CreditLimiy	ItemOrdered	Quantity	Price

Table 5.7. Customer relation

However, a customer may order several items and each customer in the database may order a different number of items. This situation makes it difficult to implement the data in a relational database, since we do not know how many order entries to allow.

1. How many order entries our database design would allow every customer?
2. Is this structure simple?
3. If this database structure is complex, identify the anomalies it has and suggest the solution to the database designer to make it simple.

5.5.1. Introduction

Another method for designing a relational database is to use a process commonly known as normalisation. The goal is to generate a set of relation schemas that allows us to store information without unnecessary redundancy, yet also allows us to retrieve information easily. The approach is to design schemas that are in an appropriate normal form. Therefore, normalisation is a part of the database design process. Normalisation is a series of steps designed to remove repeating groups and unwanted functional dependencies. Normalisation is a design technique for constructing a set of table designs from a list of data items. . It is also used to avoid insertion, deletion and updating anomalies. Therefore it used to improve on existing table designs.

5.5.2. Normal Forms

Normalization theory defines six normal forms (NF). Each normal form involves a set of dependency properties that a schema must satisfy and each normal form gives guarantees about the presence and/or absence of update anomalies. To correct update anomalies in a database, you must convert tables to various types of normal forms.

The most common normal forms are First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form(3NF) and Fourth Normal Form(4NF).

a. First Normal Form

A relation (table) that contains a repeating group (multiple entries for a single record) is called unnormalised relation. Removing repeating groups is the starting point in the quest to create tables that are as free of problems as possible. The conversion to first normal form (INF) requires splitting the data into two groups. Tables without repeating group are said to 1NF.

A relation is in first normal form if and only if the domain of each attribute contains only atomic (indivisible) values, and the value of each attribute contains only a single value from that domain

Example:

A database has been designed to store data about sellers and the products they have sold.

The following facts help to define the structure of the database:

- each seller works in a particular shop
- each seller has a unique Surname
- each shop has one or more sellers
- each product which is sold is manufactured by one company only
- each seller can sell any of the products
- the number of products that each seller has sold is recorded

Table: SHOPSALES

SurName	ShopName	ProductName	Number of Products	Manufacturer
KAGABO	Computer Tech Ltd	Digital Camera Printer Laptop	3 2 6	Nickon Canon HP
KALISA	DigiTech	Hair dryer Electric shaver	1 8	Panasonic Phillips
KAMALI	Computer Tech Ltd	Digital Camera Printer Moderm Mobile Phone	2 8 4 3	Nikon Canon Huawei Samsung

Table 5.8. SHOPSALES

This table is not in 1NF because:

1. SHOPSALES table has repeated group of attributes

(ProductName, Manufacturer)

(ProductName, NumberofProducts),

(NumberofProducts, Manufacturer)

2. each seller has a number of products
3. SurName and ShopName would need to be repeated for each record

The following table is in 1NF because it does not contain repeating groups

Table: SHOPSALES				
SurName	**ShopName**	**ProductName**	**Number of Products**	**Manufacturer**
KAGABO	ComputerTech Ltd	Digital Camera	3	Nikon
KAGABO	ComputerTech Ltd	Printer	2	Canon
KAGABO	ComputerTech Ltd	LAptop	6	HP
KALISA	DIgiTech	Hair dryer	1	Panasonic
KALISA	DIgiTech	Electric Shaver	8	Phillips
KAMALI	ComputerTech Ltd	Digital Camera	2	Nikon
KAMALI	ComputerTech Ltd	Printer	8	Canon
KAMALI	ComputerTech Ltd	Modern	4	Huawei
KAMALI	ComputerTech Ltd	Mobile Phone	3	Samsung

Table 5.9. SHOPSALES in 1NF

b. Second Normal Form (2NF)

For the second normal form, the relation must first be in 1NF. The relation is automatically in 2NF if, and only if, the Primary Key comprises a single attribute.

To move to 2NF, a table must first be in 1NF.

The database is changed to the following design:

SELLER (SurName, ShopName)

PRODUCTSSOLD (SurName, ProductName, NumberofProducts, Manufacturer)

Table: SELLERS	
SurName	**ShopName**
KAGABO	ComputerTech Ltd
KALISA	DigiTech
KAMALI	ComputerTech Ltd

Table 5.10 SELLERS

Table: PRODUCTS SOLD			
SurName	**Product Name**	**Number of products**	**Manufacturer**
KAGABO	Digital Camera	3	Nikon
KAGABO	Printer	2	Canon
KAGABO	Laptop	6	HP
KALISA	Hair dryer	1	Panasonic
KALISA	Electric shaver	8	Phillips
KAMALI	Digital Camera	2	Nikon
KAMALI	Printer	8	Canon
KAMALI	Moderm	4	Huawei
KAMALI	Mobile Phone	5	Samsung

Table 5.11: PRODUCTSSOLD

Table SELLER is in Second Normal Form because attribute ShopName depends on primary key SurName; the same on PRODUCTSSOLD where NumberofProducts and Manufacture attributes depends on composite primary key SurName, ProductName. The link between these two tables is: primary key of SELLER table is FirstName, links to FirstName in PRODUCTSSOLD table, FirstName in SalesProducts table is foreign key

c. Third Normal Form (3NF)

To be in third normal form, the relation must be in second normal form. Also all transitive dependencies must be removed; a non-key attribute may not be functionally dependent on another non-key attribute.

In our case on table PRODUCTSSOLD, Manufacturer attribute is dependent on ProductName, which is not the primary key of the PRODUCTSSOLD table therefore there is no key dependency.

Process for 3NF:

1. Eliminate all dependent attributes in transitive relationship(s) from each of the tables that have transitive relationship.
2. Create new table(s) with removed dependency.
3. Check new table(s) and modified table(s) to make sure that each table does not contain contains inappropriate dependencies.

See the three new tables below.

SELLERS (SurName, Shop)

PRODUCTSSOLD (SurName, ProductName, NumberofProducts)

PRODUCTS (ProductName, Manufacturer)

Table: SELLERS	
SurName	**ShopName**
KAGABO	ComputerTech Ltd
KALISA	DigiTech
KAMALI	ComputerTech Ltd

Table 5.12. SELLERS

Table: PRODUCTSSOLDM1

SurName	Product Name	Manufacturer
KAGABO	Digital Camera	Nikon
KAGABO	Printer	Canon
KAGABO	Laptop	HP
KALISA	Hair dryer	Panasonic
KALISA	Electric shaver	Phillips
KAMALI	Digital Camera	Nikon
KAMALI	Printer	Canon
KAMALI	Moderm	Huawei
KAMALI	Mobile Phone	Samsung

Table 5.13.PRODUCTS

Product Name	Manufacturer
Digital Camera	Nikon
Printer	Canon
Laptop	HP
Hair dryer	Panasonic

Electric shaver	Phillips
Digital Camera	Nikon
Printer	Canon
Moderm	Huawei
Mobile Phone	Samsung

Table 5.14. PRODUCTS

d. Boyce-Codd Normal Form (BCNF)

When a table has more than one candidate key, anomalies may result even though the relation is in 3NF. Boyce-Codd normal form is a special case of 3NF. A relation is in BCNF if, and only if, every determinant is a candidate key.

Example

Consider the following table STUDENT

Table: STUDENT		
Student ID	**Combination**	**Class Teacher**
100	MCE	KABALISA Jean Caude
101	PCB	MUGABO Boniface
102	MCB	MUKAMANA Scovia
103	HEG	KAGABO Pascal

Table 5.15. STUDENT

The following facts help to define the structure of the database:

1. Each Student may major in several Combinations.
2. For each Combination, a given Student has only one ClassTeacher.
3. Each Combination has several Advisors.
4. Each ClassTeacher advises only one Combination.
5. Each ClassTeacher advises several Students in one Combination.

The functional dependencies for this table are:

1. StudentId, Combination→ClassTeacher
2. ClassTeacher →Combination

Anomalies for this table include:

- Delete→Student deletes ClassTeacher inforamtion
- Insert→a new ClassTeacher needs a Student
- Update →inconsistencies or contradiction

Because no single attribute which is a candidate key, primary key can be

StudentId, Combination, Student_id,ClassTeacher.

To reduce STUDENT table to BCNF, there is creation of two tables:

- STUDENTTEACHER (Studentid, ClassTeacher)
- TEACHERCOMBINATION(ClassTeacher, Combinationr)

STUDENTTEACHER

Student ID	Class Teacher
100	KABALISA Jean Caude
101	MUGABO Boniface
102	MUKAMANA Scovia
103	KAGABO Pascal

Table 5.16 STUDENTTEACHER

TEACHER COMBINATION	
Class Teacher	**Combination**
KABALISA Jean Caude	MCE
MUGABO Boniface	PCB
MUKAMANA Scovia	MCB
KAGABO Pascal	HEG

Table 5.17 TEACHER COMBINATION

e. Fourth Normal Form (4NF)

Fourth normal form eliminates independent many to one relationships between columns.

A relation is in Fourth Normal Form:

- A relation must first be in Boyce-Codd Normal Form.

- Given relation may not contain more than one multivalued attribute.

The multi valued dependency X→Y holds in a relation R if whenever there is two tuples of R that same in all the attributes of X, then we can swap their Y components and get two new tuples that are also in R.

Table: STUDENTS		
StudentId	**Subject**	**Clubs**
110	Mathematics	Anti SIDA
110	Computer Science	Anti SIDA
110	Mathematics	Anti drugs
110	Computer Science	Anti drugs
151	Music	Environmental

Table 5.18 STUDENTS

Primary key→{StudentId , Subject , Clubs}

- Many StudentId have same Subject.
- Many StudentId have same Clubs.

Thus violates 4NF.

To convert to 4NF, the table STUDENTCLUB is changed to the following design:

STUDENTSUBJECT {StudentId, Subject}

StudentId	Subject
110	Mathematics
110	Computer Science
151	Music

Table 5.19 STUDENTSUBJECT

STUDENTCLUBS {StudentId , Clubs}

StudentId	Clubs
110	Anti SIDA
110	Anti Drugs
151	Environmental

Table 5.20 STUDENTCLUBS

Application Activity 5.6

1. Study the following table and answer corresponding questions

CUSTOMER					
CustomerNumber	**CustomerName**	**District**	**Province**	**ProductId**	**Descrip tion**
005	U W I M A N A Yvonne	Gisagara	South	1 2 3	HP Laser Jet Printer Hair Trimmer HP ProBook4530s
008	KAGABO Peter	Rusizi	West	2 7	Hair Trimmer POSITIVO BGH Laptop
014	RUKUNDO Paul	Nyarugenge	Kigali	5	Lenovo L430 Laptop
002	MUGABO Tom	Gisagara	South	7 1 2	POSITIVO BGH Laptop HP Laser Jet Printer Hair Trimmer

Table 5.21. CUSTOMER

Is this normalised table? If not, convert it to 1NF, 2NF and 3NF.

2. Consider the following table SUBJECT that contains multivalued dependency thus it is in 3NF, use it to answer asked questions

SUBJECT		
SubJect Title		**Books**
Mathematics	Eric	Mathematics fundementals
Mathematics	Bosco	Mathematics for computer Sciences
Computer Science	Callixte	Database System Concepts
Phyisics	Peter	Phisics for Rwandan

Table 5. 22: Subject Table

i. Identify functional dependencies exist in this tabale above
ii. Eliminate those dependencies.

END UNIT ASSESSMENT

Question 1.

The database below is composed of two tables. Use it to answer asked questions

BORROWERS		
BorrowerId	**BorrowerName**	**DOB**
112018	KAYIRANGA	11/12/1987
122018	KAMANA	12/12/1989
132018	KAMIHANDA	12/01/1996

Table 5.23 BORROWERS

BOOKS		
BookId	**BookTitle**	**BorrowerId**
10	Computer Science for Rwandan Schools Senior 4	112018
12	Data communications and Networking	112018
13	Fundamentals of Mathematics	122018
14	Production Economics: The Basic Theory of Production Optimisation	132018
15	Encyclopedia of Mathematics	122018

Table 5.24 BOOKS

i. What is primary key for each table and give reasons.
ii. Identify the foreign key in the BOOKS table.
iii. Identify the candidate keys in both tables.
iv. Identify the relationship between BORROWERS and BOOKS tables and draw the E-R model.
v. Does the BOOKS table exhibit referential integrity? Why or why not?

Question 2.

Write an ER diagram for Banking System. Assume your own entities (minimum 5), attributes and relation. Mention the cardinality ratio.

Question 3.

A picture gallery owner has decided to set up a database to keep information about the pictures he has for sale. The database table, PICTURE, will contain the following fields:

PictureTitle; PictureArtist; PictureDescription; CatalogueNumber; PictureSize (in MB); PicturePrice; ArrivedDate (date picture arrived at gallery); Status(whether picture is already sold)

PICTURE							
Catalogue Number	Picture Title	Picture Artist	Picture Description	Picture Size	Picture Price	Arrived Date	Status
P1000	Volcanoes In Rwanda	GATOTO	This pictures shows all volcanoes existed in Rwanda	10MB	150,000	21/12/2016	Unsold
P1001	Women and guitar	KAGABO	This pictures shows how women can also play guitar	20MB	200,000	01/06/2017	Sold
P1010	Flag Rwanda	GATOTO	Rwanda national flag	5MB	80,000	13/01/2016	Sold

Table 25. PICTURE

PictureTitle; PictureArtist; PictureDescription; CatalogueNumber; PictureSize (in MB); PicturePrice; ArrivedDate (date picture arrived at gallery); Status(whether picture is already sold)

i. State what data type you would choose for each field.
ii. State which field you would choose for the primary key.
iii. Complete the query-by-example grid below to select and show the CatalogueNumber, PictureTitle and PicturePrice of all unsold pictures by the artist 'GATOTO'.

Field:					
Table:					
Sort:					
Show:					
Criteria:					
or:					

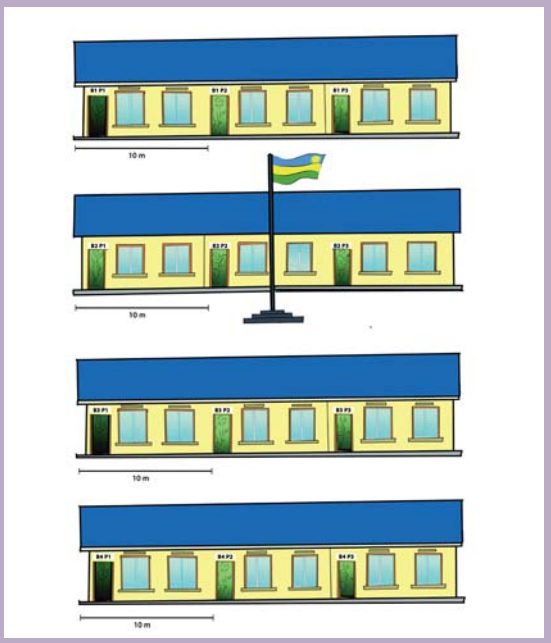
Figure 5.43. Relational Picture

UNIT 6
POINTERS AND STRUCTURE IN C++

UNIT 6: POINTERS AND STRUCTURE IN C++

INTRODUCTORY ACTIVITY

The Groupe Scolaire Nderonziza which is located in District Gakenke, Sector Amahoro, Cell Bwiza and Village Rumuri is a twelve-year basic education school. It has 4 blocks of 3 equal classrooms each. Each classroom has 10 meters of length and can contain a maximum of 30 students. Each classroom is labelled with a name made of the name of block and the name of the level as showed in the next figure.



After analyzing this figure, respond to the following questions.

1. Describe the role of each label of a classroom?
2. Discuss the way of knowing the content of each classroom?
3. If you are asked to describe the entity student, what are its attributes?
4. In C++ Programming language, how are you going to declare the variable student?
5. Suppose that at the beginning of the next year there are 60 new students who are enrolled at that school in S4. If the schools need to maintain the quality of teaching and learning by keeping a maximum of 30 students per classroom, what will happen for the 30 students who don't have space in S4?
6. In C++ programming language, how to create that space?

6.1 Pointers in C++

ACTIVITY 6.1

1. The student called Kayiranga of Computer Science Senior 5 has a class work and wants to borrow a C++ programming language book and a computer from his classmate Iradukunda. Kayiranga needs to use those materials which are on the desk of Iradukunda to submit the work on time while Iradukunda was absent.

   Discuss

   a. What could Kayiranga do if he takes those materials without informing Iradukunda?
   b. What could happen to Iradukunda when he comes back?

3. On The list, the students of Senior 5 are arranged in ascending order according to their names. From one name of a student, how to reach the next student?
   a. How to get to the student number 5 while you are on student number 2?
   b. How to get to the student number 8 while you are on student number **15?**

6.1.1 Definition of Pointers

A **pointer** is a variable which contains the address in memory of another variable. There can be a **pointer** to any variable type. The unary or monadic operator & gives the ``address of a variable''. The indirection or dereference operator * gives the ``contents of an object pointed to by a **pointer**''.

C++ gives the power to manipulate the data in the computer's memory directly. The space in the memory can be assigned and de-assigned. This is done using pointer variables. Pointer variables are variables that points to a specific address in the memory pointed to by another variable.

6.1.2 Dereference operators

The asterisk sign (*) in the cout statements is called the dereference operator. If the dereference operator is used, the "value pointed by" a pointer is got.

The statement:

cout << *ptr; means to print the content of the memory space pointed by ptr.

Example: use a pointer in calculating area of circle

```
#include<iostream>
using namespace std;
int main ()
{
float PI = 3.14;
float *PIptr;
float r=5, A;
float x;
PIptr=&PI;   //PIptr contains the address of PI
x=*PIptr;     // value stored in PI(3.14)is assigned to x.
A=r*r*x;
cout<<"The area of circle is"<<A;
getchar ();
return 0;
}
```

6.1.3 Reference operator

The ampersand sign (&) is called the reference operator. If the reference operator is used, it will get the "address of" a variable. The ptr = &x; means that it stores the address of the variable x in the pointer ptr.

Example:

```
#include<iostream>
using namespace std;
int main()
        {
                int x;
                int *ptr;
                x = 5;
                ptr = &x;//
            cout << *ptr;
                getchar();
                return 0;
        }
```

Note: If you forget to place * (in front of the pointer) in the **cout** statement, you will print the address of integer x.

Example of pointer demonstrating the use of & and *:

```
#include <iostream>
using namespace std;
int main()
{
  /* Pointer of integer type, this can hold the address of an integer type variable.   */
  int *p;
  int var = 10;
  /* Assigning the address of variable var to the pointer * p. The p can hold the address of
var because var is * an integer type variable. */
  p = &var;
  cout<< "Value of variable var is:"<< var <<endl;
  cout<<"\n Value of variable var is: "<< *p <<endl;
  cout<<"\n Address of variable var is: "<< &var <<endl;
  cout<<"\n Address of variable var is: "<< p <<endl;
  cout<<"\n Address of pointer p is: "<< &p <<endl;
  getchar();
  return 0;
}
```

Output:

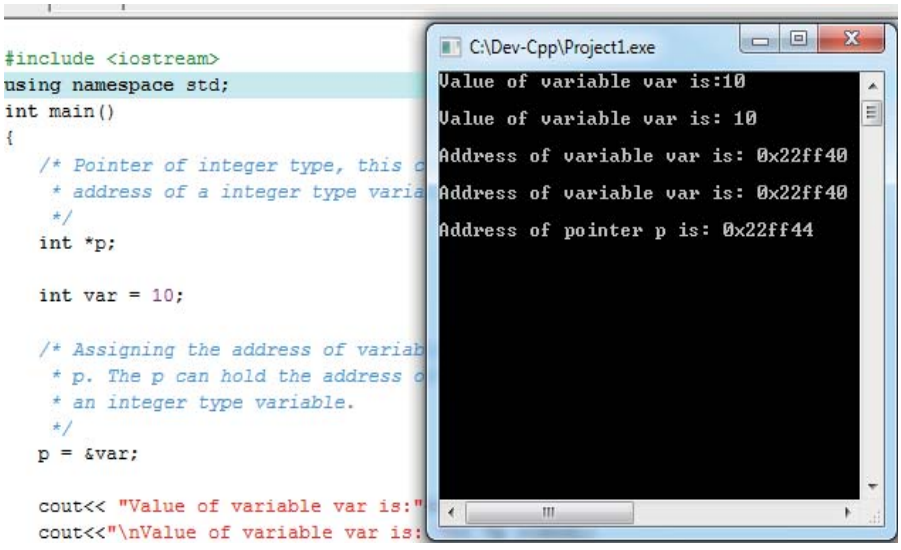


Figure 6.1: output of pointer demonstrating the use of & and * program:

Interpretation of this output:

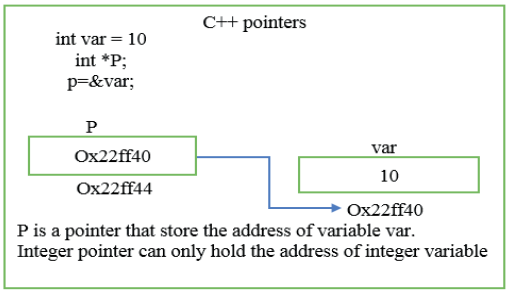


Figure 6.2: interpretation of pointer example

6.1.4 Declaring pointer variables

A pointer is declared by using the * (asterisk) operator before an identifier. It is necessary to initialize the pointer before you can use a pointer in for instance a cout statement. Due to the ability of a pointer to directly refer to the value that it points to, it becomes necessary to specify in its declaration which data type a pointer is going to point to. It is not the same thing to point to a char as to point to an int or a float.

Pointer declaration Syntax:

data_type *pt_name;

Where ***data_type*** is the data type of the value that the pointer is intended to point to.

The following are three declarations of pointers and all of them are pointers and will occupy the same amount of space in memory:

```
int * number;
char * character;
float * greatnumber;
some Pointer declaration examples
 int  *iptr;       // iptr is pointer to int
char  *cptr;      // cptr is a pointer to char
float  *fptr;      // fptr is a pointer to float
List  *lptr;       // lptr is a pointer to List object
```

6.1.5 Initialization of Pointer Variable

Pointers can be initialized to point to specific locations at the very moment they are defined,You need to initialize a pointer by assigning it a valid address.

pointer_variable = &variable;

Example of initialization:

```
int *ptr; // pointer to int declared, value undefined
int x = 5; // int declared and initialized to 5
cout << x; //prints 5
cout << *ptr; //Error! Prints undefined value, since ptr not initialized
ptr =&x; //ptr now contain value of x
cout<<*ptr; // Prints 5
Example program of Declaring and Initializing pointer:
#include<iostream>
using namespace std;
int main ()
{
        // Declaring Variables and Pointer
        int Num=10;
        int *Ptr;
        Ptr=&Num;

        // Printing Values of Pointer
        cout <<"\n Value Of Num :" << Num << endl ;
        cout <<"\n Address Of Num :" << &Num << endl;
        cout <<"\n Value Of Ptr :" << Ptr << endl;
        cout <<"\n Address Of Ptr :" << &Ptr << endl;
        cout <<"\n Ptr's Pointer Value:" << *Ptr << endl;
        cout <<"\n Ptr Equal to &Num :" << *(&Num) << endl;

        // Change Values of Using Pointer
        cout <<"\n\n We Can Change Value Of Num,Without Using Num" << endl;
        *Ptr=100;
        cout <<"\n Address Of Num :"<<&Num << endl;
        cout <<"\n Ptr's Pointer Value:"<<*Ptr << endl;
        getchar();
}
```

6.1.6 Graphical representation of pointers for referencing memory allocation.

Consider the following,

*int jim,tom,*blaise;*
jim = 10;
tom= jim;
blaise = &jim;

The following is graphical representation of the value contained in each variable after the execution of this, *are* shown in *the* following diagram:

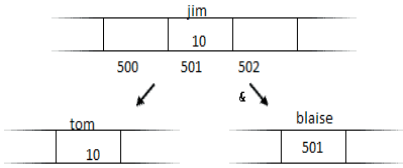


Figure 6.3: Graphical representation of the value contained variable

First, we have assigned the value 10 to jim (a variable whose address in memory we have assumed to be 501). The second statement copied to tom the content of variable jim which is 10. the third statement copies to blaise not the value contained in jim but a reference to it (its address, which we have assumed to be 501). The reason is that in this third assignment operation we have preceded the identifier jim with the reference operator (&), so we were no longer referring to the value of jim but to its reference (its address in memory).

1.1.7 Incrementing a Pointer

Incrementing a pointer is generally used in array because we have contiguous memory in array and we know the contents of next memory location. Incrementing pointer variable depends upon data type of the Pointer variable

Three Rules should be used to increment pointer

Address + 1 = Address

Address++ = Address

++Address = Address

The following program increments the variable pointer to access each succeeding element of the array –

#include <iostream>

```
using namespace std;
const int MAX = 3;
int main () {
  int  var[MAX] = {10, 100, 200};
  int  *ptr;
  // let us have array address in pointer.
  ptr = var;
  for (int i =0 ; i < MAX; i++) {
    cout << "Address of var[" << i << "] = ";
    cout << ptr << endl;
    cout << "Value of var[" << i << "] = ";
    cout << *ptr << endl;
    // point to the next location
    ptr++;
  }
  getchar();
  return 0;}
```

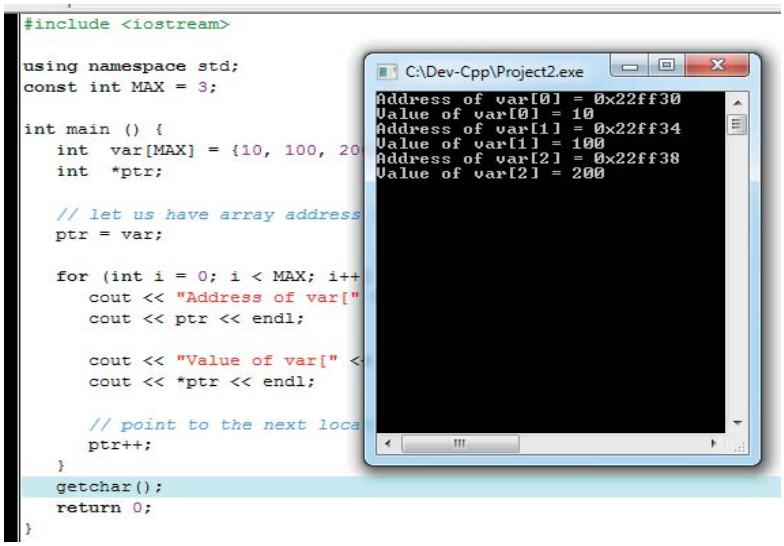


Figure 6.4: output of incrementation example program

6.1.8 Decrementing a Pointer

As shown in example below, the program decrementing a pointer, which decreases its value by the number of bytes of its data type

```
#include <iostream>
using namespace std;
const int MAX = 3;
```

```
int main () {
 int  var[MAX] = {10, 100, 200};
 int  *ptr;
  // let us have address of the last element in pointer.
  ptr = &var[MAX-1];
  for (int i = MAX; i > 0; i--) {
  cout << "Address of var[" << i << "] = ";
   cout << ptr << endl;
   cout << "Value of var[" << i << "] = ";
   cout << *ptr << endl;
   // point to the previous location
   ptr--;
  }
  getchar();
  return 0;
}
```

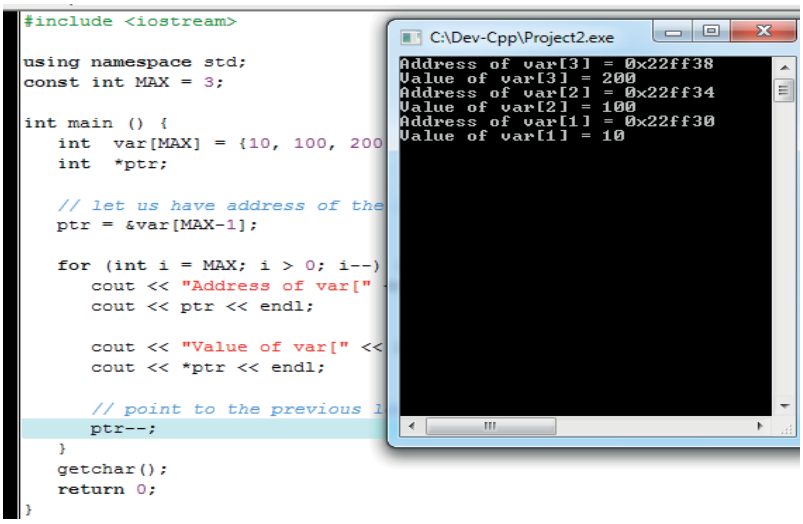


Figure 6.5: Output of Decrementing example program

6.1.9 Arrays and pointers

The name of an array is a pointer to the first element in the array.

```
int a[10];
int *aptr = a;
a          // is equivalent to &a[0]
aptr = a    // is equivalent to  aptr = &a[0];
aptr+5     // is equivalent to &a[5]
```

(aptr+5)
// is equivalent to a[5]

Exemple :

```
#include <iostream>
using namespace std;
int main()
{
  float arr[10];
  float *ptr;
  cout << "Displaying address using arrays: " << endl;
  for (int i = 0; i < 10; ++i)
  {
    cout << "&arr[" << i << "] = " << &arr[i] << endl;
  }
  // ptr = &arr[0]
  ptr = arr;
  cout<<"\n Displaying address using pointers: "<< endl;
  for (int i = 0; i < 10; ++i)
  { cout << "ptr + " << i << " = "<< ptr + i << endl;
  }
 getchar();
  return 0;}
```

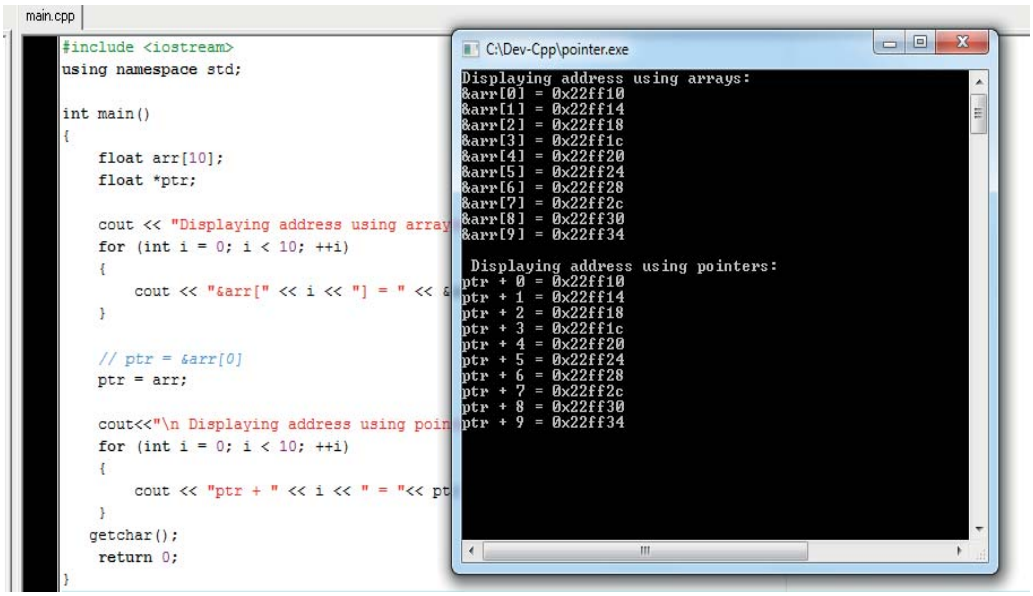


Figure 6.6: Output of array example program

Different pointer **ptr** is used for displaying the address of array elements **arr**. Pointers are the variables that hold address. Not only can pointers store address of a single variable, it can also store address of cells of an array.

6.1.10 Comparison of pointers

Pointers may be compared by using relational operators, such as ==, <, and >.

The following program incrementing the variable pointer so long as the address to which it points is either less than or equal to the address of the last element of the array, which is &var[MAX - 1].

```
#include <iostream>
using namespace std;
const int MAX = 3;
int main () {
  int  var[MAX] = {10, 100, 200};
  int  *ptr;
  // let us have address of the first element in pointer.
  ptr = var;
  int i = 0;
  while ( ptr <= &var[MAX - 1] ) {
    cout << "Address of var[" << i << "] = ";
    cout << ptr << endl
    cout << "Value of var[" << i << "] = ";
    cout << *ptr << endl;
    // point to the previous location
    ptr++;
    i++;
  }
  getchar();
  return 0;}
```

6.1.11 Subtracting pointers

Differencing means subtracting two Pointers. Subtraction gives the Total number of objects between them. Subtraction indicates "How apart the two Pointers are?"

Example:

```
#include<iostream>
using namespace std;
int main()
```

```
{
int num , *ptr1 ,*ptr2 ;
ptr1 = &num ;
ptr2 = ptr1 + 2 ;
cout << " the difference of two pointers is: " << ptr2 - ptr1 << endl;
getchar ();
return 0;
}
```

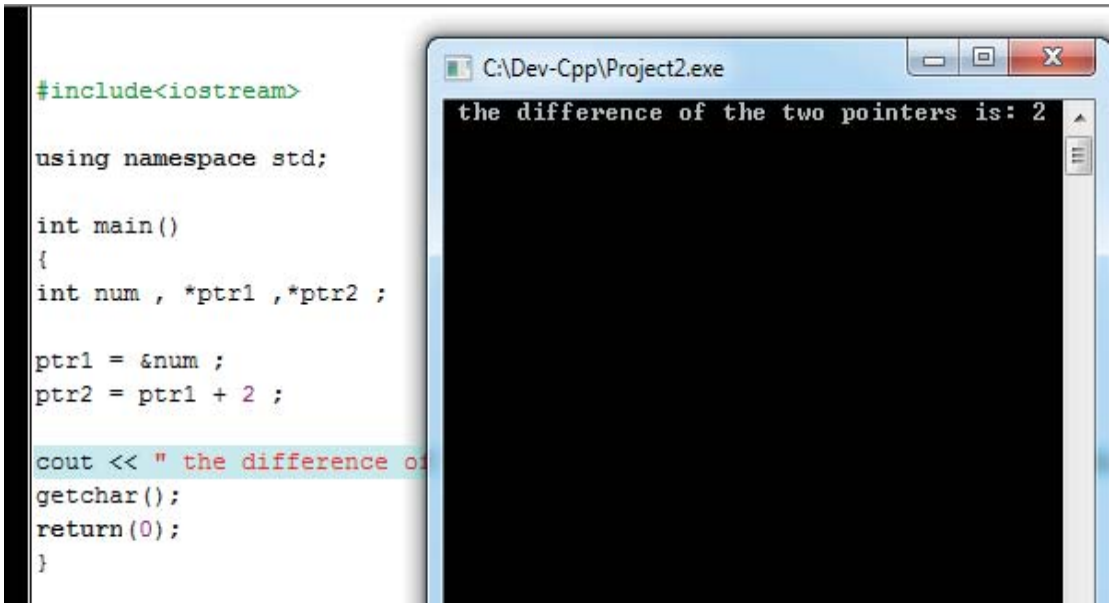


Figure 6.7: Output of subtracting example program

6.1.12 Assignment of pointers

In pointer assignment, the pointer is associated with a target. If the target is undefined or disassociated, the pointer acquires the same status as the target

pointer-object => target ,

6.1.13 Pointers and function in C++

A function pointer is a variable that stores the address of a function that can later be called through that function pointer.

Example:

```
#include<iostream>
using namespace std;
void swapping (int *ptr_a, int *ptr_b)
```

```
        {
                int tmp; tmp = *ptr_a;
                *ptr_a = *ptr_b;
                *ptr_b= tmp;
                cout << "In function:\n" << *ptr_a << '\n' << *ptr_b<< '\n';
        }
int main()
        {
                int c,d;
                c=5;
                d=10;
                cout << "Before:\n" << c << '\n' << d << '\n';
                swapping(&c,&d);
                cout << "After:\n" << c << '\n' << d << '\n';
        getchar();
                return 0;
}
```

The address of variable is passed during function call rather than the variable itself.

```
#include <iostream>

using namespace std;
const int MAX = 3;

int main () {
   int  var[MAX] = {10, 100, 200};
   int  *ptr;

   // let us have address of the first
   ptr = var;
   int i = 0;

   while ( ptr <= &var[MAX - 1] ) {
      cout << "Address of var[" << i <
      cout << ptr << endl;

      cout << "Value of var[" << i <<
      cout << *ptr << endl;

      // point to the previous location
      ptr++;
      i++;
   }
   getchar();
   return 0;
}
```

```
C:\Dev-Cpp\Project2.exe
Address of var[0] = 0x22ff30
Value of var[0] = 10
Address of var[1] = 0x22ff34
Value of var[1] = 100
Address of var[2] = 0x22ff38
Value of var[2] = 200
```

Figure 6.8: Output of pointer and function example program

Activity1:

a. What is the pointer in C++

b. Explain the use of Address Operator (&) and operator (*).

Activity 2:

Given the following declarations, which of the following statements are not valid?

int i;
*int *pi;*
double d;
*double *pd;*
*a. i=π b.*pi=&i; c. pd=π d. pd=i; e. pi=&i;*

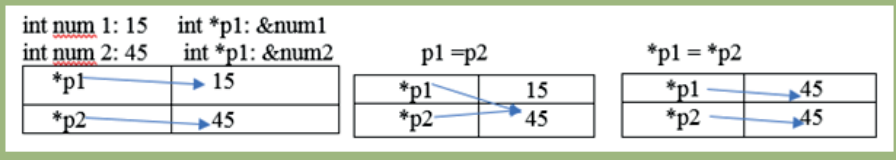
Activity3:

What is the error in each of the following expressions?

*int *p=10;*
int a=10;
*int **p=&a;*
float x=5;
*int **p=x;*

Activity 4:

Write a program containing pointers to implement what is taking place in the tables below



//pointers -store the address of a variable

// (*) dereference operator

//*p1 ==> return value of the variable the pointer is pointing to

//p1 ==> return address of where variable is stored

//p1 ==> return address of pointer

//&p1==> returns address of pointer

6.2. Memory allocation

ACTIVITY 6.2

Public bonded warehouse known as MAGERWA (Magazin Generaux Rwandais) is a Rwandan company where all goods from abroad are stored before being distributed in the country. When an importer brings his/her goods to MAGERWA, he/she is allocated a space where to put those goods for checking and is charged the space per square meter (m2) per day. When he/she withdraws his/her goods from MAGERWA, the occupied space is given to other customers for use.

Discuss: What would happen to the MAGERWA store if all customers failed to withdraw their goods on time?

6.2.1 Definition

Memory allocation means reserving memory for specific purposes. It is a process by which computer programs and services are assigned with physical or virtual memory space, the process of reserving a partial or complete portion of computer memory for the execution of programs and processes. Once the program has finished its operation, the memory is released and allocated to another program or merged within the primary memory. Programs and services are assigned with a specific memory as per their requirements when they are executed

6.2.2 Dynamic memory allocation in C++

Up to this level, all memories needed in programs were determined before the program execution by defining the variables where it is reserved for. But there may be cases where the memory needed in a program can only be determined during runtime.

When a memory is dynamically allocated, the operating system is asked to reserve some of that memory for that program's use. If it can fulfill this request, it will return the address of that memory to the current application. From that point forward, the application can use this memory as it wishes. When the application is done with the memory, it can return the memory back to the operating system to be given to another program. This allows to obtain more memory when required and release it when not necessary.

C++ integrates the operators **new** and **delete** for allocating dynamic memory. With the functions malloc, calloc, realloc and free are available in C++ and can also be used to allocate and deallocate dynamic memory.

A. Operators:

i. Operator new and new[]

Dynamic memory is allocated using operator ***new***. **New** is followed by a data type specifier and, if a sequence of more than one element is required, the number of these are within brackets []. It returns a pointer to the beginning of the new block of memory allocated.

Its syntax is:
pointer = new type
pointer = new type [number_of_elements]

The first expression is used to allocate memory to contain one single element of type type. The second one is used to allocate a block (an array) of elements of type type, where number_of_elements is an integer value representing the amount of these. For example:

```
int * mem;
mem= new int[6];
```

There is a substantial difference between declaring a normal array and allocating dynamic memory for a block of memory using new. The most important difference is that the size of a regular array needs to be a constant expression, and thus its size has to be determined at the moment of designing the program, before it is run, whereas the dynamic memory allocation performed by new allows to assign memory during runtime using any variable value as size.

The dynamic memory requested by a program is allocated by the system from the memory heap (unused memory of the program). However, computer memory is a limited resource, and it can be exhausted. Therefore, there are no guarantees that all requests to allocate memory using operator new are going to be granted by the system.

ii. Operators delete and delete[]

In most cases, memory allocated dynamically is only needed during specific periods of time within a program; once it is no longer needed, it can be freed so that the memory becomes available again for other requests of dynamic memory. This is the purpose of operator delete, whose syntax is:

```
delete pointer;
delete[] pointer;
```

The first statement releases the memory of a single element allocated using new, and the second one releases the memory allocated for arrays of elements using

new and a size in brackets ([]).

The value passed as argument to delete shall be either a pointer to a memory block previously allocated with new, or a null pointer (in the case of a null pointer, delete produces no effect).

b. Functions

Malloc (): void* malloc (size_t size);

Allocate memory block. Allocates requested size of bytes of memory and returns a pointer to the beginning of the block (first byte of allocated space). Simply pass in how big you want your memory to be (in bytes), and you get a pointer to that memory back. The content of the newly allocated block of memory is not initialized, remaining with indeterminate values. If it fails it returns NULL. It is routine used to allocate a single block of memory.

Calloc (): void* calloc (size_t num, size_t size);

Allocate and zero-initialize array. Allocates space for an array element, initializes to zero and then returns a pointer to memory. The calloc() takes two integer arguments and multiplied together to specify how much memory to allocate. This is routine used to allocate arrays of memory.

Free (): void free (void* ptr);

Deallocate memory block (the previously allocated space). A block of memory previously allocated by a call to malloc, calloc or realloc is deallocated, making it available again for further allocations.

Realloc (): void* realloc (void* ptr, size_t size);

Reallocate memory block, Change the size of previously allocated space pointed to by ptr. The function may move the memory block to a new location (whose address is returned by the function).

Example:

```
#include <iostream>
#include <stdlib.h>
#include <string.h>
using namespace std;
int main () {
  char *str;
  /* Initial memory allocation */
  str = (char *) malloc(10);
  strcpy(str, "HelloWorld");
  cout<<"String is:"<<str<<"  "<<"Address is:"<<&str<<endl;
```

```
 /* Reallocating memory */
 str = (char *) realloc(str, 15);
 strcat(str, ".com");
cout<<"String is:"<<str<<"  "<<"Address is:"<<&str<<endl;
 free(str);
str = NULL;
 getchar();
 return(0);
}
```

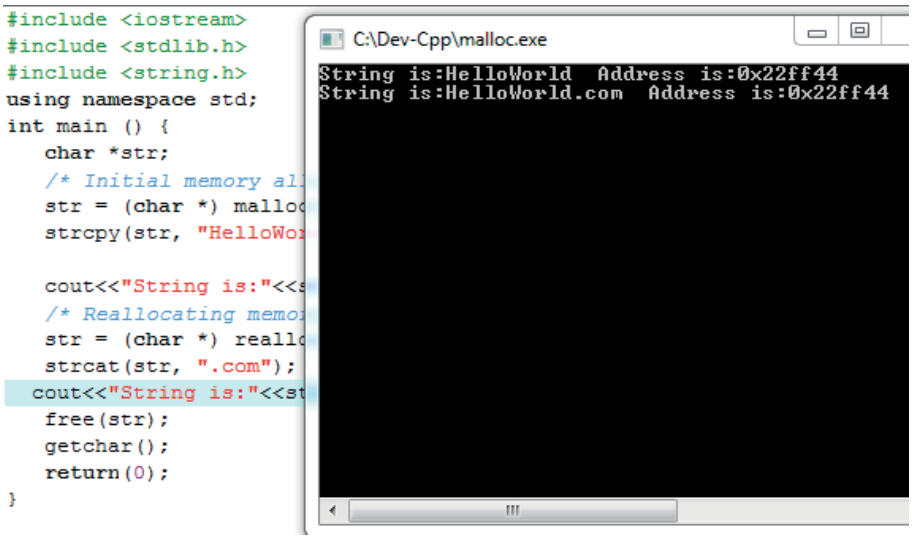


Figure 6.9: Output of Dynamic memory allocation example program

Notice: In the above program:

Malloc () function dynamically allocates initial value of the size maximum 10 characters (e.g the size of HelloWord has 9 characters) otherwise it returns null and is copied to str variable which has address in memory.

Realloc() function :reallocate the content of str concatenating with **.com** and change the size from 10 to 15 but it keeps the existing address which is 0x22ff44.

Free(): the memory size of (10) characters that has been reallocated with realloc () is released after using free() because it is no longer needed.

6.2.3 Static memory (or Compile Time) allocation

The memory for your variables is allocated when the program starts, the size is fixed when the program is created

Statically allocated variables have their storage allocated and initialized before main starts running and are not deallocated until main has terminated. Statically allocated

local variables are not re-initialized on every call to the function in which they are declared. A statically allocated variable thus has the occasionally useful property of maintaining its value even when none of the functions that access the variable are active.

6.2.4 Advantages of dynamic memory allocation over static memory allocation

An advantage is that the program determines how much memory it needs at run time and allocate exactly the right amount of storage and released when it's no longer needed.

- Dynamic Memory Allocation
  * Memory allocated "on the fly" during run time
  * Dynamically allocated space usually placed in a program segment known as the heap or the free store
  * Exact amount of space or number of items does not have to be known by the compiler in advance.
  * For dynamic memory allocation, pointers are crucial
- Static (or Compile Time) Allocation
  * Memory for named variables is allocated by the compiler
  * Exact size and type of storage must be known at compile time
  * For standard array declarations, this is why the size has to be constant

Application activity 6.2

Activity 1:

Explain the memory allocation and compare dynamic memory over static memory allocation.

6.3 C++ Structures

ACTIVITY 6.3

The company Urukundo Ltd specialized in production of Milk Powder has a group of 20 employees and its management is not computerized. Each employee is identified by the attributes first name, second name, age, degree and salary. The programmer of that company wants to write a program in C++ programming language that will allow to constitute a file containing the information of each employee.

1. What should be the data type of each attribute?
2. Is it easy to enter the value of each attribute one by one? What should be the solution?
3. If employee is considered as a single entity, how will it be declared in the program?

6.3.1 Introduction to C++ structure

Structure is a collection of variables of different data types under a single name. These data elements, known as members, can have different types. To store some information about a student: Name, Age and student ID. Creation of different variables name, Age, ID needed to store this information separately.

6.3.2 How to declare a structure in C++ programming?

The struct keyword defines a structure type followed by an identifier (name of the structure).

Data structures can be declared in C++ using the following syntax:

```
struct type_name {
member_type1 member_name1;
member_type2 member_name2;
member_type3 member_name3;
. } object_names;
```

Where:

type_name is a name for the structure type, object_name can be a set of valid identifiers for objects that have the type of this structure. Within braces {}, there is a list with the data members, each one is specified with a type and a valid identifier as its name.

6.3.3 Initializing structure in C++ programming

Just like any variable, we can specify the initial value of a structure at the time of its definition.

Example of Initialization of structure:

```
#include<iostream>
using namespace std;
struct Book
    {
char name [100];
float price;
int  pages;
   };
int main ()
{
struct Book b1= {"Computer Programing with C++",30000,250};
```

```
struct Book b2= {"Computer Networking ",15000,200};
cout<<"Book 1 Details\n"<<endl;
cout<<"Title:\t"<<b1.name<<endl;
cout<<"Price:\t"<<b1.price<<endl;
cout<<"Pages:\t"<<b1.pages<<endl<<"\n";
cout<<"Book 2 Details\n"<<endl;
cout<<"Title:\t"<<b2.name<<endl;
cout<<"Price:\t"<<b2.price<<endl;
cout<<"Pages:\t"<<b2.pages<<endl;
getchar ();
return 0;
}
```

Members of object b1 and b2 are initialized. Values which will be passed to members of the structure are listed in the braces.

6.3.4 Calculate Memory consumption by structure

To generate the program below, padding must be added to the structs to satisfy this alignment criteria**.** It creates two structs; both contain 3 values of the same types and of the same name. It then prints out the sizes of these structs (in bytes).

```
#include <iostream>
using namespace std;
struct A
{
  char a;
  long b;
  char c;
};
struct B
{
  char a;
  char c;
  long b;
};
int main()
{
A a;
  B b;
cout << "sizeof(a): " << sizeof(a) << endl;
cout << "sizeof(b): " << sizeof(b) << endl;
getchar ();
  return 0;
```

```
}
Output
sizeof(a): 12
sizeof(b): 8
```

Note: Output may vary depending on the systems

The C++ standard defines that struct members must be arranged in memory in the same order that they are defined within their access level. Different types must have certain memory alignment requirements; long (on a 64-bit platform) uses 8 bytes of memory and must start at a memory location divisible by 8 char, on the other hand, only uses 1 byte of memory and can start at any memory address.

6.3.5 Accessing members of a structure

We need to specify both the structure name (name of the variable) and the member name when accessing information stored in a structure. There are two types of operators used for accessing members of a structure:

1. **Member operator (.) or Dot Operator**

structure_variable_name.member_name

For Example:
We can access the member age of employee structure variable employee_one as:

```
struct employee
{
  char name[50];
  int age;
  float salary;
  char department[30];
} employee_one = {"paul", 25, 150000.5, "ICT"};
int age = employee_one.age;
```

2. **Structure pointer operator (->) or Arrow Operator (->)**

Structure pointer operator or Arrow operator is used to access members of structure using pointer variable. When we have pointer to a structure variable, then we can access member variable by using pointer variable followed by an arrow operator and then the name of the member variable.

structure_pointer->member_name;

For example:

```
struct employee
{
 char name [50];
 int age;
 float salary;
 char department [30];
} employee_one = {"Paul", 25, 150000.5, "ICT"};
struct employee *ptr = &employee_one;
int age = ptr->age;
```

//C++ Program to print the members of a structure using dot and arrow operators:

```
#include <iostream>
#include <conio.h>
using namespace std;
struct employee
{
char name[100];
int age;
float salary;
char department[50];
};
int main()
{
struct employee employee_one, *ptr;
 cout<<"Enter Name, Age, Salary and Department of Employee\n";
cin>>employee_one.name>>employee_one.age>>employee_one.
salary>>employee_one.department;
  /* Printing structure members using dot operator */
       cout<<"Employee Details\n" <<employee_one.name<<"\t" <<employee_one.
age<<"\t" <<employee_one.salary <<"\t" <<employee_one.department ;
  /* Printing structure members using arrow operator */
 ptr=&employee_one;
cout<<"\n\nEmployee Details\n"<<ptr->name<<"\t"<<ptr->age<<"\t"<<ptr-
>salary<<"\t"<<ptr->department;
 getch();
 return 0;
}
```

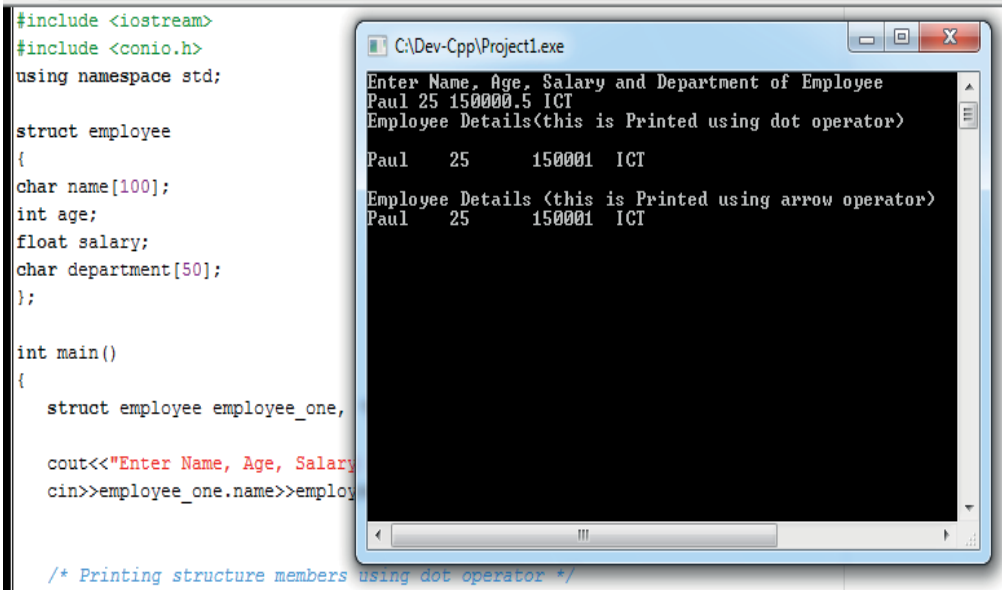


Figure 6.10: Output of structure members example program

6.3.6 Global use of structure

As defined, a structure is a compound data type that contains different members of different types, **structs** are simpler to be managed by the programmer and the compiler,a better example of using a global variable, and a situation where global variables are completely necessary, is when passing a structure to a function. In that case, you must declare the structure as global so that all functions can access variables of that structure type.

6.3.7 Pointer to structure

Just like other variables we can use pointers to access information stored in a structure. Structures can also be pointed by pointers and store pointers. The rules are the same as for any fundamental data type. The pointer must be declared as a pointer to the structure.

6.3.8 Nesting structure

Structures can be nested within other structures in C++ programming. Structures can also be nested so that a valid element of a structure can also be another structure.

When a structure contains another structure, it is called nested structure. For example, we have two structures named Address and Student. To make Address nested to Student, we have to define Address structure before and outside Student structure and create an object of Address structure inside Student structure.

Syntax for structure within structure or nested structure

```
struct structure1
        {
            ----------
            ----------
        };
        struct structure2
        {
            ----------
            ----------
            struct structure1 obj;
        };
```

Example:

```
#include <iostream>
#include<conio.h>
using namespace std;
   struct Address
    {
        char SchoolName[25];
        char Prov[25];
        char District[25];
        char City[25];
        char Code[25];
    };
    struct Student
    {
      int Id;
      char FName[25];
      char SName[25];
      char StudentClasse[25];
      char StudentCourse[25];
      float Marks;
      struct Address Add;
    };
    int main()
    {
        int i;
        Student S;
        cout << "\n\t Enter Employee Name : ";
        cin >> S.FName;
        cout << "\n\t Enter Student Second Name : ";
        cin >> S.SName;
```

```
        cout << "\n\tEnter Student Id : ";
        cin >> S.Id;
        cout << "\n\tEnter Student Shool Name : ";
        cin >> S.Add.SchoolName;
        cout << "\n\tEnter Student Shool province : ";
        cin >> S.Add.Prov;
        cout << "\n\tEnter Student Shool District : ";
        cin >> S.Add.District;
        cout << "\n\tEnter Student Shool City : ";
        cin >> S.Add.City;
        cout << "\n\tEnter Student Shool code : ";
        cin >> S.Add.Code;
        cout << "\n\tEnter Student marks : ";
        cin >> S.Marks;
         cout << "\nDetails of Employees";
         cout << "\n\tStudent First Name: " << S.FName;
        cout << "\n\tStudent Second Name : " << S.SName;
        cout << "\n\tStudent ID : " << S.Id;
        cout << "\n\tSchool Name : " << S.Add.SchoolName;
        cout << "\n\t School Province : " << S.Add.Prov;
        cout << "\n\t School District : " << S.Add.District;
        cout << "\n\t School City : " << S.Add.City;
        cout << "\n\t school code : " <<S.Add.Code;
        cout << "\n\t Marks No : " << S.Marks;
getch();
return 0;
    }
```

6.3.9 Array of structure

Arrays of structure are good for storing information of a single entity. By declaring an array of structures, you specify the number of reserved structures inside array brackets when you declare the structure variable.

Example:

Consider learning activity 6.3 example:

```
#include <iostream>
#include <conio.h>
#include<stdio.h>
#include<string.h>
using namespace std;
struct Employee
```

```
{
 int empcode;
 char empname[20];
  char empdesig[20];
  float empsalary;
};

int main ()
{
  int i;
//   clrscr();
  // Define 1 local structure variable with 20 employee information
  Employee emp[3];
  for (i=1; i<= 3; i++)
  {
     cout<<"\nEnter info about Employee No "<<i <<":\n\n" ;
    //get into employee information
     cout<<"\tEnter employee code:";
     cin >>emp[i].empcode;
     cout<<"\tEnter employee name:";
     cin>>emp[i].empname;
     cout<<"\tEnter employee designation:";
     cin>>emp[i].empdesig;
     cout<<"\tEnter employee salary:";
     cin>>emp[i].empsalary;
}
//print the structure emp1 information to the screen
      cout<<"\n\nHere is the employee information:\n\n";
      for(i=1;i<=3;i++)
      {
     cout<<"\nInformation about Employee No "<<i <<" is \n" ;
     cout<<"\n\t Employee code:"<<emp[i].empcode;
     cout<<"\n\t Employee name:"<<emp[i].empname;
     cout<<"\n\t Employee designation:"<<emp[i].empdesig;
      cout<<"\n\t Employee salary:"<<emp[i].empsalary<<"\n";
     }
  getch();
return 0;
}
```

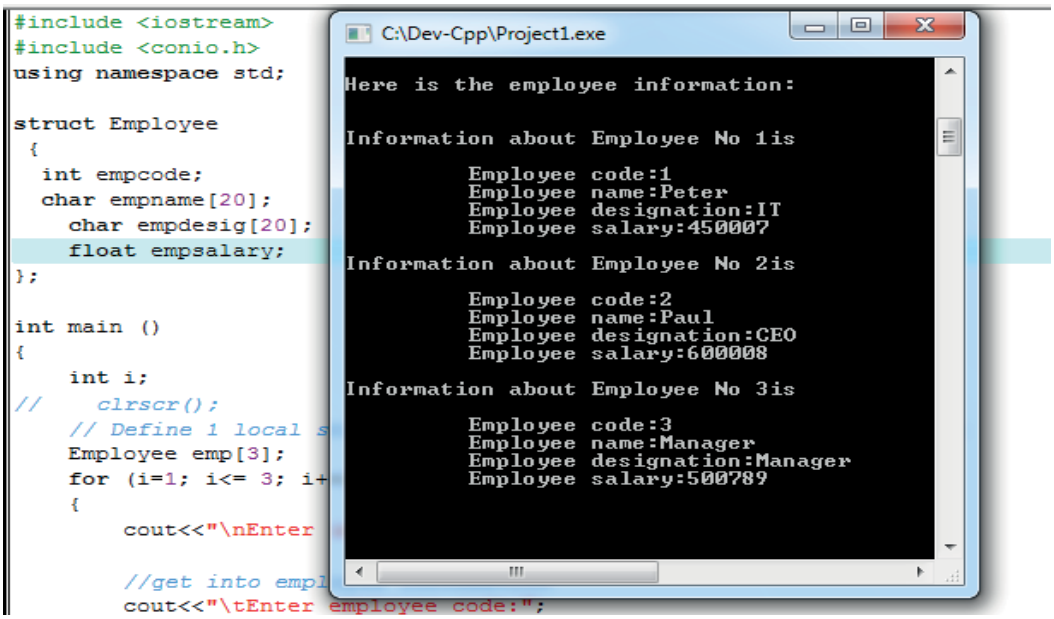


Figure 6.11: Output of array structure example program

Application activity 6.3

1.

a. Define a structure in C++

b. Given a structure definition,

struct Employee {
string emName;
string emSex;
string emAddress;
}; Employee Emp;

A member of the structure can be accessed by which one of the following statements?:

a. Employee.emName;
b. Emp.emName;
c. Emp->emName;

2. Store Information of students and Display It Using Structure by creating an array having 4 elements of structure student. Each of the element stores the information of a student.

   For example, st[0] stores the information of the first student, st[1] for the second and so on.

3. What is the output of the following program?

```
#include<iostream>
#include<conio.h>
using namespace std;
void swap (int m, int n) {
iint x = m;
      m = n;
     n = x;
}
main() {
  int x = 5, y = 3;
swap(x,y);
          cout<<x<<" "<<y;
          getch();
}
```

4. What are the operators used to access member function of a structure using its object?
5. What is the output of the following program?

```
#include<iostream>
using namespace std;
main() {
  class student
{
    int rno = 10;
  } v;
 cout<<v.rno;
}
```

END UNIT ASSESSMENT

1. What would be printed from the following C++ program?

```
#include<iostream>
#include<conio.h>
using namespace std;
int main(){
 int x[5]={1,2,3,4,5};
 int *p=x;
 int i;
 for(i=0;i<2;i++)
  {
   int temp=*(p+i);
   *(p+i)=*(p+4-i);
   *(p+4-i)=temp;
  }
 for(i=0;i<5;i++)
cout<<x[i]<<"\t";
 getch();
 return 0;
}
```

2. By accessing member of structure using C++ write a simple running program that will display the output of your birthday and today date.
3. Write a program that calculates the roots of a quadratic Equation 2x2 +6x +4 in C++ using pointers.

UNIT 7

OBJECT ORIENTED PROGRAMMING IN C++

UNIT 7: OBJECT ORIENTED PROGRAMMING IN C++

Key Unit Competence:

To be able to explain common concepts of Object Oriented Programming (OOP) and implement them in C++.

INTRODUCTORY ACTIVITY

Human beings can be grouped into males and females. Each of them has two legs, two hands, two eyes, a nose, a heart, and so on. There are body parts that are common to males and females, but there are specific body parts for males and not for Females, and vice versa.

Both Males and Females perform some common functions like walking, eating, seeing, talking, hearing etc. But there are some specific functions for each group of Human Beings.

After reading the above, do the following.

1. Categorize human beings into two main groups
2. What are the body parts that all human beings have in common?
3. List the parts that are specific for females and males.
4. What all human beings can do in common? What are specific functions only for each gender?
5. Based on database concepts, write down the properties of the entities human being, male and female. Design on papers a Diagram of entities found in the above scenario.
6. Add to each entity the functions raised in question 3, 4 and 5. Which entity depends on the other one?
7. Do you need to repeat the properties and functions of human being entity to male entity and to female entity? Explain your answer.
8. After considering the question above, design a diagram of the new entities.
9. Discuss your answer

7.1. Introduction to Object Oriented Programming (OOP)

Activity 7.1

Compare the programming paradigms that you have learnt so far.

1. Which one can help you to create a program for the "Introductory activity"?
2. Search and write an essay on the concepts and principles of that new programming paradigm.

7.1.1 Definition of Object Oriented Programming(oop)

***O**bject-**o**riented **p**rogramming* (**OOP**) refers to a type of computer programming (software design) in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure.
In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another.

7.1.2 Basic Concepts of Object Oriented Programming

The main concepts and principles used within Object Oriented Programming are:

- Objects
- Classes
- Abstraction
- Encapsulation
- Inheritance
- Polymorphism
- Exception Handling

Definitions of OOP concepts

a. Object

An objects is the basic unit of OOP. It is an instance of class, which has data members and use various member functions to perform tasks.

b. Class

A class is basically a blueprint of an object. It can also be defined as user defined data type but it also contains functions in it. It declares and defines what data variables the object will have and what operations can be performed on the class's object.

c. Encapsulation

Encapsulation means that the internal representation of an object is generally hidden from view outside of the object's definition. Typically, only the object's own methods can directly inspect or manipulate its fields. It can also be said data binding. Encapsulation is all about binding the data variables and functions together in class.

d. Abstraction

Data abstraction and encapsulation are closely tied together, because a simple definition of data abstraction is the development of classes, objects, types in terms of their interfaces and functionality, instead of their implementation details. Abstraction denotes a model, a view, or some other focused representation for an actual item.

In C++, classes can provide methods to the outside world to access & use the data variables, keeping the variables hidden from direct access, or classes can even declare everything accessible to everyone, or maybe just to the classes inheriting it. This can be done using access specifiers.

e. Inheritance

Inheritance as the key feature of Object-Oriented Programming is a way to reuse code of existing objects, or to establish a subtype from an existing object, or both, depending upon programming language support. In classical inheritance where objects are defined by classes, classes can inherit attributes and behavior from pre-existing classes called base classes, super classes, parent classes or ancestor classes. The resulting classes are known as derived classes, subclasses or child classes. The relationships of classes through inheritance gives rise to a hierarchy.

The derived class inherits all the features from the base class and can have additional features of its own.

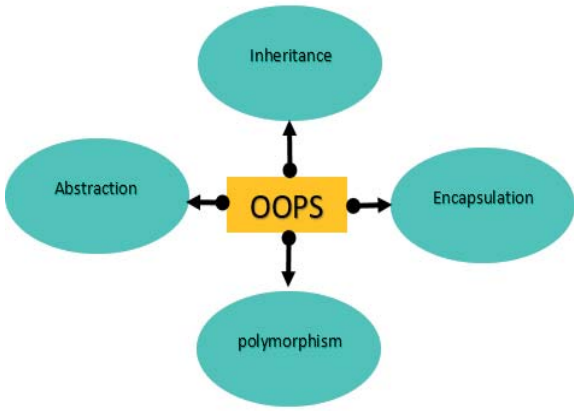
f. Polymorphism

It is a feature, which lets programmers create functions with same name but different arguments, which will perform different actions. That means, functions with same name, but functioning in different ways. Or, it also allows users to redefine a function and provide it with a completely new definition.

g. Exception Handling

Exception handling is a feature of OOP, to handle unresolved exceptions or errors produced at runtime (during the running of the program).

Graphically the principles of Object Oriented Paradigm can be represented like this.



Advantages of Object Oriented Programming

One of the principal advantages of Object-Oriented Programming techniques over procedural programming techniques is that they enable programmers to create modules that do not need to be changed when a new type of object is added. A programmer can simply create a new object that inherits many of its features from existing objects. This makes object-oriented programs easier to modify.

So, Object Oriented Programming has great advantages over other programming styles.

1. **Code Reuse and Recycling**: Objects created for Object Oriented Programming can easily be reused in other programs.
2. **Data Redundancy:** Inheritance is the good feature for data redundancy. If you need a same functionality in multiple classes, you can write a common class for the same functionality and inherit that class to sub class.
3. **Data hiding:** Implementation details are hidden from other modules and other modules has a clearly defined interface.
4. **Design Benefits**: Large programs are very difficult to write. Object Oriented Programming force designers to go through an extensive planning phase, which makes for better designs with less flaws. In addition, once a program reaches a certain size, Object Oriented Programs are actually *easier* to program than non-Object Oriented ones.
5. **Software Maintenance:** An Object Oriented Program is much easier to modify and maintain than a non-Object Oriented Program. So although a lot of work is spent before the program is written, less work is needed to maintain it over time.

Application Activity 7.1

1. The following are the principles of OOP: Object, Class, Encapsulation, inheritance and Polymorphism. Explain them one by one.
2. What are the advantages of OOP?
3. Name the features that are added to standard C++.

7.2 Class definition in C++

Activity 7.2

Observe the C++ program below and answer the following questions:

```
#include<iostream>
Using namespace std;
class Test
{
     int a;
    float b;
  public:
    void number1()
    {  a = 2;
    return a;}
    float number2()
    {
      b = 3.5;
      return b;
    }
  };
int main()
{
test m1;
cout <<"First number is: "<< m1.number1()<<'\n';
cout <<"Second number is: "<< m1.number2()<<'\n';
return0;
}
```

From the program above, answer the following questions:

1. What are new key words in the above program?
2. Describe the structure of the part starting with the word class.
3. List the data members contained in the class "Test"
4. What are the functions found in that class?
5. What is the role of the variable m1?

7.2.1 Definition of a class

A class is defined in C++ using keyword class followed by the name of class.

The body of class is defined inside the curly brackets and terminated by a semicolon at the end.

7.2.2 Syntax

```
class class_name {
 access_specifier_1:
  member1;
 access_specifier_2:
  member2;
 ...
};
```

Example1:

```
classTest
{
private:
int data_1;
double data_2;
public:
void function_1()
{
 data_1 =2;
}
double function_2()
{
 data_2 =3.5;
return data_2;
}
};
```

Here, the class is named Test. This class has two data members: data_1 and data_2 and two member functions: function_1() and function_2().

A Class is an expanded concept of data structures, like data structures, it can contain data members, but it can also contain functions as members. Where class_name is a valid identifier for the class, object_names is an optional list of names for objects of this class. The body of the declaration can contain members, which can either be data or function declarations, and optionally access specifiers.

Classes have the same format as plain data structures, except that they can also include functions and have these new things called access specifiers (data hiding). An access specifier is one of the following three keywords: private, public or protected.

Access specifiers are used to identify access rights for the data and member functions of the class. There are three main types of access specifiers in C++ programming language:

- **A private** member within a class denotes that only members of the same class have accessibility. The *private* member is inaccessible from outside the class.
- **Public** members are accessible from outside the class.
- **A protected access specifier** is a stage between *private* and *public* access. If member functions defined in a class are *protected*, they cannot be accessed from outside the class but can be accessed from the derived class.

When defining access specifiers, the programmer must use the keywords: *private*, *public* or *protected* when needed, followed by a colon and then define the data and member functions under it.

```
class AddNumbers
{
  private:
  int x,y;
  public:
  void sum()
  {
    .....
    .....
  }
};
```

In the code above, the member *x* and *y* are defined as private access specifiers. The member function sum is defined as a public access specifier.

General structure for defining a class is:

```
class class_name
{
  access_specifier:
  data member;
  member_functions;
   access_specifier:
  data_member;
  member_functions;
};
```

Generally, in class, all members (data) would be declared as private and the member functions would be declared as public. If no access specifiers are identified for members of a class, the members are defaulted to private access (Private is the default access level for specifiers).

```
class AddNumbers
{
  int a,b;
  public:
  void sum()
  {
    ……
    ……
  }
};
```

In this example, for members **a** and **b** of the class **AddNumbers** there are no access specifiers identified. This means that by default **a** and **b** are private members of class **AddNumbers**.

You can have functions inside a class. These functions can be either under public or private, and their syntaxes are the same. For example, if you have a function to display information of an account called *display*, and other classes should be able to access that function, the following line would be added under the public area:

```
void display (accountno);
```

And use the following syntax to implement the function:

```
void BankAccount::display (int accountno)
{
……;
……;
}
```

The computer accesses an object through the use of one of the object's methods. The method performs some action to the data in the object and returns this value to the

computer. Classes of objects can also be further grouped into hierarchies, in which objects of one class can inherit methods from another class. The structure provided in object-oriented languages makes them very useful for complicated programming tasks.

7.2.3 Defining member functions inside or outside the class definition

Member functions of a class can be defined either inside or outside the class definition. In both cases, the function body remains the same, however, the function header is different.

7.2.3 .I Inside the Class

A member function of a class can be defined inside the class. However, when a member function is defined inside the class, the class name and the scope resolution operator are not specified in the function header. Moreover, the member functions defined inside a class definition are by default inline functions.

To understand the concept of defining a member function inside a class, consider this example.

Example: Definition of a member function inside a class

```
Class student
{
string fname;
float marks;
public:
void getdata(); //declaration
void putdata()//definition inside the class
{
cout<<"\nstudent name is: "<<fname;
cout<<"\nhis marks are: "<<marks;
} ;
```

In this example, the member function putdata() is defined inside the class book. Hence, putdata()is by default an inline function.

Note that the functions defined outside the class can be explicitly made inline by prefixing the keyword inline before the return type of the function in the function header. For example, consider the definition of the function getdata().

```
inline void student ::getdata ()
{
body of the function
}
```

7.2.3.2 Outside the Class

Defining a member function outside a class requires the function declaration (function prototype) to be provided inside the class definition. The member function is declared inside the class like a normal function. This declaration informs the compiler that the function is a member of the class and that it has been defined outside the class. After a member function is declared inside the class, it must be defined (outside the class) in the program.

The definition of member function outside the class differs from normal function definition, as the function name in the function header is preceded by the class name and the scope resolution operator (: :). The scope resolution operator informs the compiler what class the member belongs to.

Here is the syntax for defining a member function outside the class:

```
Return_type class_name :: function_name (parameter_list)
{
// This is the body of the function
}
```

To understand the concept of defining a member function outside a class, consider this example.

Example

Let us re-write the program which converts the degree from Fahrenheit to Celsius and displays the corresponding degrees on the screen. By creating the method outside the class, we have:

```
#include<iostream>
#include<conio.h>
using namespace std;
class temperature
{
private:
float degreeCelcius;
float tempFahr;
public:
void conversion (float); // declaration of a method or fnct
}; // end of the class
void temperature::conversion(float a)  //definition of a method
{
```

```
tempFahr=a;
degreeCelcius=((tempFahr-32)*(0.55));
cout<<"After conversion the degree in Celcius is:"<<degreeCelcius<<endl;
} // end of the function
int main()
{
temperature celsius;  // creation of an object statically
float b;
cout<<"Enter degree in Fahrenheit:"<<endl;
cin>>b;
celsius.conversion (b); // calling a method
getch();
}
```

Note that the member functions of the class can access all the data members and other member functions of the same class (private, public or protected) directly by using their names. In addition, different classes can use the same function name.

Application Activity 7.2

The below piece of code is for class declaration, analyze it and answer the following questions:

```
#include <iostream>
using namespace std;
        class Rectangle
{
   int width,height;
public;
        set_values (int,int);
        int area() {return width*height;}
}
```

a. Show errors found in the program above
b. Correct those errors

7.3 Object in C++

Activity 7.3

Your District wants to create a program that will help to know its primary and secondary schools. Create the class "school" with the following attributes and functions:

Decide on the types of attributes

a. Attributes:

1. Name of the school
2. Code of the school
3. Number of student in your school
4. Number of teachers
5. reading the attributes
6. printing the attributes

B (i) Write a program in C++ that reads and prints the list of schools available in your District.

(ii) complete the table below with the following list:

1. school
2. School name
3. Code of the school
4. Combinations
5. Number of students
6. Number of teachers
7. Read_school ()
8. Print_school()

Data member(s)	Member function(s)	Class(s)	Object(s)

The first characteristic of an object language is to place at your disposal **the objects**.

An object can be regarded as an additional structure of information, a type of super-variable.

Indeed, we know that a variable is a place in primary memory, characterized by **an address – a name - and a type** (integer, double, character, Boolean, etc).

In a variable, we can store one and only one information.

Even if an indexed variable is used - a table - the various memory spaces thus defined will store all obligatorily information of the same type.

7.3.1 Definition:

An *object* is an instantiation of a class. In terms of variables, a class would be the type, and an object would be the variable. So, an object is a group of variables of various types. It thus usually gathers tens of very different information from/to each other within the same structure, thus making this information easier to handle. By analogy with equation of Wirth, we could say that the equation of the Object Oriented Programming (OOP) is: **methods+-data=objects**

With the difference of what occurs with a table, the various variables of the same object are not indicated by an index, but by a name which is proper for them. In fact, these names which characterize the various variables within an object are called properties of the object. As consequence, any property of object obeys strictly the rules which apply to the variables in any language (type, size, rules of assignment…).

It will be also said that several objects which have the same properties are of the same type, or for better explanation, of the same class.

Example

let us say that you want to make a program which will record or process the data on the books in a library or bookshop, take an object of everyday usage: **"A book"**.

The properties of a book are:

its size (number of the pages), the name of the author, date of publication, printing company, its price, its name (title), if a person can borrow it or not, etc.

You can easily find the type of each one of these properties:

- The title and the name of author are the properties of the character type.
- The size (a number of pages) and the price are the properties of the numerical type.

The situation "if one can borrow it from the library or not" is a Boolean property.

7.3.2 Creating single object

When a class is defined, only the specifications for the object is defined, the memory or storage is not allocated. To use the data and access functions defined in the class, you need to create objects.

7.3.2.1 Syntax to define Object in C++

className objectVariableName;

You can create objects of *Test* class (defined in above example) as follows:

```
classTest
{
private:
int data1;
float data2;

public:
void function1()
{  data1 =2;}
float function2()
{
data2 =3.5;
return data2;
}
};
int main()
{
Test o1, o2;
}
```

Here, two objects ***o1*** and ***o2*** of ***Test*** class are created.

In the above class *Test*, *data1* and *data2* are data members and *function1()* and *function2()* are member functions

Example: Class and Object

```
// Program to illustrate the working of objects and class in C++ Programming
#include<iostream>
usingnamespace std;
classTest
{
private:
int data1;
float data2;
public:
void insertIntegerData(int d)
{
data1 = d;
```

```
 cout <<"Number: "<< data1;
}
float insertFloatData(){
cout <<"\nEnter data: ";
cin >> data2;
return data2;
}};
int main()
{Test o1, o2;
float secondDataOfObject2;
o1.insertIntegerData(12);
secondDataOfObject2 = o2.insertFloatData();
cout <<"You entered "<< secondDataOfObject2;
return0;
}
```

Output

Number: 12

Enter data: 23.3

You entered 23.3

In this program, two data members *data1* and *data2* and two memberfunctions insertIntegerData() and insertFloatData() are defined under *Test* class.

Two objects *o1* and *o2* of the same class are declared.

The insertIntegerData() function is called for the *o1* object using:

o1.insertIntegerData(12);

This sets the value of data1 for object *o1* to 12.

Then, the insertFloatData() function for object *o2* is called and the return value from the function is stored in variable *secondDataOfObject2* using:

secondDataOfObject2 = o2.insertFloatData();

In this program, *data2* of *o1* and *data1* of *o2* are not used and contains garbage value.

7.3.2.2 Creating array of objects

```
#include <iostream>
using namespace std;

class MyClass {
 int x;
public:
 void setX(int i) { x = i; }
```

```
 int getX() { return x; }
};

int main()
{
 MyClass obs[4];
 int i;

 for(i=0; i < 4; i++)
  obs[i].setX(i);

 for(i=0; i < 4; i++)
  cout << "obs[" << i << "].getX(): " << obs[i].getX() << "\n";

 return 0;
}
```

OUTPUT

obs[0].getX(): 0

obs[1].getX(): 1

obs[2].getX(): 2

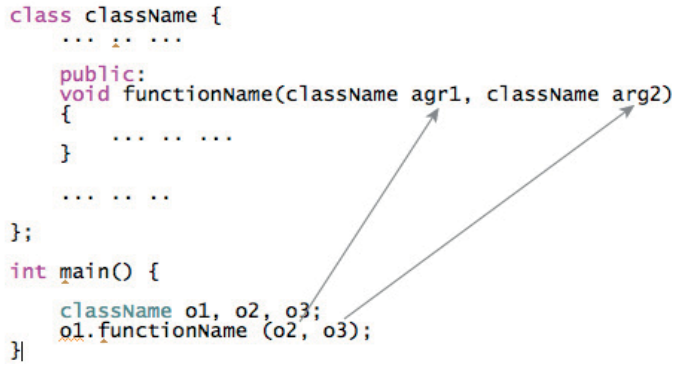
obs[3].getX(): 3

7.3.3. Passing object to function

As it is known, we can pass (give) any type of arguments within the member function which can have any numbers of arguments. In C++ programming language, it is also possible to pass an object as an argument within the member function of class.

This is useful, when we want to initialize all data members of an object with another object, we can pass objects and assign the values of supplied object to the current object. For complex or large projects, we need to use objects as an argument or parameter.

Syntax:



Example: C++ program to add two complex numbers by passing objects to a function.

```
#include <iostream>
using namespace std;
class Demo
{
        private:
                int a;
        public:
                void set(int x)
                {
                        a = x;
                }

                void sum(Demo ob1, Demo ob2)
                {
                        a  = ob1.a + ob2.a;
                }

                void print()
                {
                        cout<<"Value of A :  "<<a<<endl;
                }
};
int main()
{
        //object declarations
        Demo d1;
        Demo d2;
        Demo d3;
        //assigning values to the data member of objects
        d1.set(10);
        d2.set(20);
        //passing object d1 and d2
        d3.sum(d1,d2);
        //printing the values
        d1.print();
        d2.print();
        d3.print();
        return 0;
}
```

Output:

Above example demonstrate the use of object as a parameter. We are passing d1 and d2 ob-

jects as arguments to the sum member function and adding the value of data members a of both objects and assigning to the current object's (that will call the function, which is d3) data member a

Application activity 7.3

1. Write the syntax of class definition.
2. Create a class called person which has 2 functions: getdata() and putdata(). The getdata() prompts the user to enter his or her first name name, last name and age, the putdata() displays on the screen the user first name last, name and age.
3. Write a C++ program which converts in Celsius the Fahrenheit degree entered on the screen and displays it on the screen. Use the concepts of class and object to resolve the problem. The class is called temperature and the object is called Celsius.

   Formula of conversion:

$$Celsius = (Fahrenheit - 32) * \frac{5}{9}$$

$$Fahrenheit = \left(\frac{Celsius * 9}{5}\right) + 32$$

4. Write a C++ program which calculate the sum, the mean and the percentage of the marks of a student in 5 courses namely Visual Basic (vb), C++ programming (cpp), web design (wd), mathematics (math) and physics (phy). The User shall be asked to enter the name and surname of the student then the marks obtained by that students in 5 courses, the program will then calculate the sum, mean and the percentage then display on the screen the name, surname, mean and percentage of that student.

Instructions: Use the concepts preserved by the Object Oriented Programming, by creating a class which is called **result** and an object called **test**. Create 4 functions namely **readident, readmarks, calculate** and finally **display**. Define the methods inside the class.

The function **readident** will be used to read the name and surname of the student.

- The function **readmarks** will receive the marks of 5 courses in parameters and assigns the values of these 5 marks to 5 variables called VB, CPP, KINYARWANDA, MATH and PHYSICS respectively.
- The function **calculate** will be used to calculate the sum, mean and the percentage. Use the following variables to store the values: sum, mean and percent.
- The function **display** will be used to display the name, surname, courses studied by the students and the marks obtained by the students in those 5 courses out of 20 marks in each course then the sum, mean and the percentage.

5. What is the output of the following program

```
#include<iostream>

using namespace std;

class sum
{
int x,y,total;
public:
sum( int a,int b)
{
x=a;
y=b*2;
}
void display()
{
total = x + y;
cout<<total;
}
};
int main()
{
sum s(9,5);
s.display();
return 0;
}
```

7.4 Data Encapsulation and Data Abstraction in C++

Activity 7.4

Mr KAGABO drives a car from MUHANGA to KIGALI but he will take a break when he reaches Musambira to buy milk for his family when returned in the car he took a remote of the radio and start listening to music:

1. List all mechanisms that KAGABO will use to stop and start the car
2. How the ignition happens
3. How exactly the gear box allows you to control speed
4. State all parameters that make the radio working in KAGABO's car?

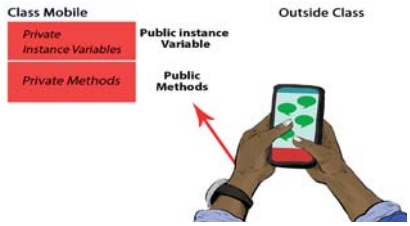
dlAll C++ programs are composed of the following two fundamental elements

- **Program statements (code)**: This is the part of a program that performs actions and they are called functions.
- **Program data**: The data is the information of the program which gets affected by the program functions.

Encapsulation is an Object Oriented Programming concept that binds together the data and functions that manipulate the data, and that keeps both safe from outside interference and exploitation. Data encapsulation led to the important OOP concept of **data hiding**.

Data encapsulation is a mechanism of bundling the data, and the functions that use them and **data abstraction** is a mechanism of exposing only the interfaces and hiding the implementation details from the user.

Consider an example of Bluetooth in a mobile phone. When we switch on the Bluetooth we are able to connect another mobile but not able to access the other mobile features like dialing a number, accessing inbox etc. This is because, Bluetooth feature is given some level of abstraction. Another point is when mobile A is connected with mobile B via Bluetooth whereas mobile B is already connected to mobile C then A is not allowed to connect C via B. This is because of accessibility restriction.



C++ supports the properties of encapsulation and data hiding through the creation of user-defined types, called **classes**. We already have studied that a class can contain **private, protected** and **public** members. By default, all items defined in a class are private. For example

```
class Box {
  public:
    double getVolume(void) {
      return length * breadth * height;
    }
  private:
    double length;     // Length of a box
    double breadth;    // Breadth of a box
    double height;     // Height of a box
};
```

The variables length, breadth, and height are private. This means that they can be accessed only by other members of the Box class, and not by any other part of your program. This is one way encapsulation is achieved.

To make parts of a class public (i.e., accessible to other parts of your program), you must declare them after the public keyword. All variables or functions defined after the public specifier are accessible by all other functions in your program.

Making one class a friend of another exposes the implementation details and reduces encapsulation. The ideal is to keep as many of the details of each class hidden from all other classes as possible.

Data Encapsulation Example

Any C++ program where you implement a class with public and private members is an example of data encapsulation and data abstraction. Consider the following example

```
#include <iostream>
using namespace std;

class Adder {
  public:
    // constructor
    Adder(int i = 0) {
      total = i;
    }
     // interface to outside world
    void addNum(int number) {
```

```
    total += number;
  }

  // interface to outside world
  int getTotal() {
    return total;
  };

 private:
  // hidden data from outside world
  int total;
};
int main() {
 Adder a;

 a.addNum(10);
 a.addNum(20);
 a.addNum(30);

 cout << "Total " << a.getTotal() <<endl;
 return 0;
}
```

When the above code is compiled and executed, it produces the following result

Total 60

Above class adds numbers together, and returns the sum. The public members **addNum** and **getTotal** are the interfaces to the outside world and a user needs to know them to use the class. The private member **total** is something that is hidden from the outside world, but is needed for the class to operate properly.

Application Activity 7.4

1. Define data encapsulation
2. What is the difference between data encapsulation and data abstraction?
3. Write an example of program and show with comment the encapsulated data and abstracted data

7.5 Friend function in C++

Activity 7.5

In NYAMAGABE District there are two schools (school GS UMWANANKUNDI and GS TUMURERE) GS TUMURERE don't have required books in its library but GS UMWANANKUNDI have books for all subject stored in their library and accessed only by members of the school (student and staff).

1. What the GS TUMURERE will do to create a link that will let them also access the library of GS UMWANANKUNDI
2. Relate to this scenario to how friend function may work

One of the important concepts of OOP is data hiding, i.e., a non-member function cannot access an object›s private or protected data. But, sometimes this restriction may force programmer to write long and complex codes. So, there is mechanism built in C++ programming to access private or protected data from non-member functions. This is done using a friend function or/and a friend class. If a function is defined as a friend function then, the private and protected data of a class can be accessed using the function.

The compiler knows a given function is a friend function by the use of the keyword **friend**. For accessing the data, the declaration of a friend function should be made inside the body of the class (can be anywhere inside class either in private or public section) starting with **keyword friend.**

Declaration of friend function in C++

```
class class_name
{
  ... .. ...
  friend return_type function_name(argument/s);
  ... .. ...
}
```

Now, you can define the friend function as a normal function to access the data of the class. No friend keyword is used in the definition.

```
class className
{
  ... .. ...
  friend return_type functionName(argument/s);
  ... .. ...
```

```
}
return_type functionName(argument/s)
{
  ... .. ...
  // Private and protected data of className can be accessed from
  // this function because it is a friend function of className.
  ... .. ...
}
```

Example 1: Working of friend Function

```
/* C++ program to demonstrate the working of friend function.*/
#include <iostream>
using namespace std;

class Distance
{
  private:
    int meter;
  public:
    Distance()
                {
                meter=0; }
    //friend function
    friend int addFive(Distance);
};

// friend function definition
int addFive(Distance d)
{
  //accessing private data from non-member function
  d.meter += 5;
  return d.meter;
}

int main()
{
  Distance D;
  cout<<"Distance: "<< addFive(D);
  return 0;
}
```

Output

Distance: 5

Here, friend function addFive() is declared inside Distance class. So, the private data *meter* can be accessed from this function. Though this example gives you an idea about the concept of a friend function, it doesn't give show you any meaningful use. A more meaningful use would to when you need to operate on objects of two different classes. That's when the friend function can be very helpful.You can definitely operate on two objects of different classes without using the friend function but the program will be long, complex and hard to understand.

Example 2: Addition of members of two different classes using friend Function

```
#include <iostream>
using namespace std;
// forward declaration
class B;
class A {
  private:
    int numA;
  public:
    A(): numA(12) { }
    // friend function declaration
    friend int add(A, B);
};
class B {
  private:
     int numB;
  public:
     B(): numB(1) { }
     // friend function declaration
     friend int add(A , B);
};

// Function add() is the friend function of classes A and B
// that accesses the member variables numA and numB
int add(A objectA, B objectB)
{
  return (objectA.numA + objectB.numB);
}

int main()
{
```

```
    A objectA;
    B objectB;
    cout<<"Sum: "<< add(objectA, objectB);
    return 0;
}
```

Output

Sum: 13

In this program, classes A and B have declared add() as a friend function. Thus, this function can access private data of both class. Here, add() function adds the private data *numA* and *numB* of two objects objectA and objectB, and returns it to the main function.

To make this program work properly, a forward declaration of a class class B should be made as shown in the above example.

This is because class B is referenced within the class A using code: friend int add(A , B);.

Application Activity 7.5

1. How friend function is declared
2. Why we use friend functions in C++
3. write a C++ program that apply friend function

Activity 7.6

Bosco is a student in primary school, most of times he goes to school for his studies, at the end of school he came back at home, the mother of Bosco sends him to the market to buy the salt for meals, Here Bosco have to behave differently depend on the place he is.

1. How Bosco will behave when he is at home?
2. How he will behave at the market?
3. How he will behave at school?

Polymorphism is derived from two Greek words: **poly** and **morphs**. The word "poly" means many and **morphs** means forms. So polymorphism means many forms.

The process of representing one Form in multiple forms is known as **Polymorphism**. Here one form represent original form or original method always resides in base class and multiple forms represents overridden method which resides in derived classes.

7.7 Overloading

C++ allows you to specify more than one definition for a **function** name or an **operator** in the same scope, which is called **function overloading** and **operator overloading** respectively.

An overloaded declaration is a declaration that is declared with the same name as a previously declared declaration in the same scope, except that both declarations have different arguments and obviously different definition (implementation).

When you call an overloaded **function** or **operator**, the compiler determines the most appropriate definition to use, by comparing the argument types you have used to call the function or operator with the parameter types specified in the definitions. The process of selecting the most appropriate overloaded function or operator is called **overload resolution**.

a. Operator overloading

You can redefine or overload most of the built-in operators available in C++. Thus, a programmer can use operators with user-defined types as well.

Overloaded operators are functions with special names: the keyword "operator" followed by the symbol for the operator being defined. Like any other function, an overloaded operator has a return type and a parameter list.

Box operator+(const Box&);

declares the addition operator that can be used to **add** two Box objects and returns final Box object. Most overloaded operators may be defined as ordinary non-member functions or as class member functions. In case we define above function as non-member function of a class then we would have to pass two arguments for each operand as follows

Box operator+(const Box&, const Box&);

Following is the example to show the concept of operator over loading using a member function. Here an object is passed as an argument whose properties will be accessed using this object, the object which will call this operator can be accessed using **this** operator as explained below

```
#include <iostream>
using namespace std;
```

```
class Box {
  public:
    double getVolume(void) {
      return length * breadth * height;
    }
    void setLength( double len ) {
      length = len;
    }
    void setBreadth( double bre ) {
      breadth = bre;
    }
    void setHeight( double hei ) {
      height = hei;
    }
       // Overload + operator to add two Box objects.
    Box operator+(const Box& b) {
      Box box;
      box.length = this->length + b.length;
      box.breadth = this->breadth + b.breadth;
      box.height = this->height + b.height;
      return box;
    }
      private:
    double length;     // Length of a box
    double breadth;    // Breadth of a box
    double height;     // Height of a box
};

// Main function for the program
int main() {
  Box Box1;          // Declare Box1 of type Box
  Box Box2;          // Declare Box2 of type Box
  Box Box3;          // Declare Box3 of type Box
  double volume = 0.0;    // Store the volume of a box here
   // box 1 specification
  Box1.setLength(6.0);
  Box1.setBreadth(7.0);
  Box1.setHeight(5.0);
```

```
 // box 2 specification
 Box2.setLength(12.0);
 Box2.setBreadth(13.0);
 Box2.setHeight(10.0);
 // volume of box 1
 volume = Box1.getVolume();
 cout << "Volume of Box1 : " << volume <<endl;
 // volume of box 2
 volume = Box2.getVolume();
 cout << "Volume of Box2 : " << volume <<endl;
 // Add two object as follows:
 Box3 = Box1 + Box2;
 // volume of box 3
 volume = Box3.getVolume();
 cout << "Volume of Box3 : " << volume <<endl;
 return 0;
}
```

When the above code is compiled and executed, it produces the following result.

Volume of Box1 : 210

Volume of Box2 : 1560

Volume of Box3 : 5400

Overloadable/Non-overloadableOperators

Following is the list of operators which can be overloaded

+	-	*	/	%	^
&	|	~	!	,	=
<	>	<=	>=	++	--
<<	>>	==	!=	&&	||
+=	-=	/=	%=	^=	&=
|=	*=	<<=	>>=	[]	()
->	->*	new	new []	delete	delete []

b. Function overloading

You can have multiple definitions for the same function name in the same scope. The definition of the function must differ from each other by the types and/or the number of argu-

ments in the argument list. You cannot overload function declarations that differ only by return type.

Following is the example where same function **print()** is being used to print different data types

```
#include <iostream>
using namespace std;

class printData {
  public:
    void print(int i) {
      cout << "Printing int: " << i << endl;
    }
    void print(double  f) {
      cout << "Printing float: " << f << endl;
    }
    void print(char* c) {
      cout << "Printing character: " << c << endl;
    }
};
int main(void) {
  printData pd;

  // Call print to print integer
  pd.print(5);

  // Call print to print float
  pd.print(500.263);

  // Call print to print character
  pd.print("Hello C++");

  return 0;
}
```

When the above code is compiled and executed, it produces the following result

Printing int: 5

Printing float: 500.263

Printing character: Hello C++

Example: this is the example that explain how minus (-) operator can be overloaded for prefix as well as postfix usage.

```
#include <iostream>
using namespace std;

class Distance {
  private:
    int feet;          // 0 to infinite
    int inches;        // 0 to 12

  public:
    // required constructors
    Distance() {
      feet = 0;
      inches = 0;
    }
    Distance(int f, int i) {
      feet = f;
      inches = i;
    }

    // method to display distance
    void displayDistance() {
      cout << "F: " << feet << " I:" << inches <<endl;
    }
    // overloaded minus (-) operator
    Distance operator- () {
      feet = -feet;
      inches = -inches;
      return Distance(feet, inches);
    }
};
int main() {
  Distance D1(11, 10), D2(-5, 11);
  -D1;               // apply negation
  D1.displayDistance();   // display D1

  -D2;               // apply negation
  D2.displayDistance();   // display D2

  return 0;
}
```

When the above code is compiled and executed, it produces the following result

F: -11 I:-10

F: 5 I:-11

Application Activity 7.6

1. Which among the following is not true for polymorphism?

It is feature of OOP

Ease in readability of program

Helps in redefining the same functionality

Increases overhead of function definition always

2. Complete the following table with No or Yes

OPERATOR	OVERLOADED
+	
?:	
.*	
-	

3. Why do we use the operator overloading in a C++ program?

4. Observe the following C++ codes then write down the output of that program

```
#include<iostream>
using namespace std;
class student
{
  public : int marks;
        void gakwaya()
        {
              cout<<"GAKWAYA went to school ";
        }};
        class topper:public student
        {
              public :
              void gakwaya()
              {
                    cout<<"He is back from school";
              }
};
        int main() { student s; topper t;
        s.gakwaya();
        t.gakwaya();
}
```

7.8 Constructors and Destructors

Activity 7.7

How can you define the following terms?

a. Constructor
b. Destructor

Introduction:

Constructors and destructors determine how the objects of a class are created, initialized, copied, and destroyed. They are member functions whose names are distinguished from all other member functions because they have the same name as the class they belong to. Constructors and destructors have many of the characteristics of normal member functions.

You declare and define them within the class, or declare them within the class and define them outside--but they have some unique features.

7.8.1. Constructor

A constructor is a member function that is automatically called when an object of that class is declared. Inside a constructor function you will do all the initialization you want on the data members. For example you can assign the account number, put in the amount of money, etc.

A constructor has the same name as the class in which it is created. It does not return any type and can have an unspecified number of arguments.

Notice that you are allowed to have more than one constructor. This is called overloading of functions. Therefore, if you have more than one constructor in your class you can have more than one method to declare an object of that kind.

A constructor is member function, defined as other functions and is called automatically at each creation of an object.

a. Types of constructors

a.1 Default constructor

Default constructor is the constructor which does not take any argument. It has no parameter.

Syntax:

Class_name()
{ Constructor Definition }

Example:

```
class cube
{
int side;
public:
cube()//default constructor definition
{
side=10;
}
};
int main ()
{
cube c; // creating object by using default constructor
……
return 0;
}
```

In this case, as soon as the object is created the default constructor is called which initialize its data members.

A default constructor is so important for initialization of object member, that even if we do not define a constructor explicitly, the compiler will provide a default constructor implicitly.

a.2 Parametarised constructor

These are the constructors with parameters. Using constructor you can provide different values to data members of different objects, by passing the appropriate values as argument.

Syntax:

```
Class_name( parameters)
{ Constructor Definition }
```

Example:

```
#include<iostream>
using namespace std;
class cube
{
int side;
public:
cube(int i)
{
side=i;
}
```

```
};
int main ()
{
cube c(10);//creating object c using parameterised constructor
cube m(45);//creating object m using parameterised constructor
return 0;
}
```

a.3 Copy constructor

These are special type of constructor which takes an object as argument, and is used to copy values of data members of one object into other object.

b. Characteristics of constructor

Name of the constructor function is same as that of the class they are part of.

No return type is required for the constructor function.

Constructor functions are automatically called at the time of object creation/declaration.

Constructor functions are always defined in the public type access specifier.

Constructor functions can be overloaded.

c. Call of the constructors

If an object has a constructor, its declaration must obligatorily comprise the corresponding arguments. For example, if the class point has a prototype constructor:

cube(int);

The following declaration will not be correct:

cube a; //incorrect: the constructor waits one argument
cube b(3,2); // incorrect (same reason)
This one on the other hand will be appropriate:
cube a (7); // correct because the constructor has one argument

If there are several constructors, it is enough that the declaration comprises the arguments required by one of them. Then, if the class pointer has the following constructors: (it is the overloading of the constructors).

cube( ); //constructor 1
cube(int); //constructor 2

The following declaration will be rejected:

cube a(5); //incorrect, no constructor has one argument

But these ones will be appropriate:
cube a; //correct: call of the constructor 1
cube b(9); //correct: call of the constructor 2

With regard to the chronology (sequence of events), we can say that:

- The constructor is called after the creation of the object;
- The destructor is called before the destruction of the object.

Up to now we noted that the call of a constructor is done during the creation of the object. But the object can be created in static or dynamic way.

d. Calling parameterized constructors (Implicitly, Explicitly)

example 1. In static: i.e. cube a (6);

cube: name of the class

a: name of the object, created statically

(6): the parameter which is between the brackets.

Example 2. Dynamically: i.e. cube *pa=new cube (6);

The constructor is called by defining a pointer to an object of desired type, then by affecting it the value returned by "new".

Example of a program implementing the constructor

```
#include<iostream>
#include<conio.h>
using namespace std;
class example
{
double h,w,area;
public:
example(double height, double width)
{
h=height;
w=width;
}
void rectanglearea(void)
{
area=h*w;
}
void display(void)
```

```
{
cout<<"The area of the rectangle is:"<<area<<endl;
}
};  //end of the example class

int main( )
{
double a,b;
cout<<"Enter a: and b:"<<endl;
cin>>a>>b;
example object(a,b);
object.rectanglearea( );
object.display( );
}
```

7.8.2 Overloading of constructors

Constructors are just like other functions, they can also be overloaded, that is same named constructers can be used to give different set of outputs depending upon the kind of input given to them. The concept of constructer overloading can be easily understood by the program given below.

```
# include<iostream>
# include<conio.h>
using namespace std;
 class work
{
 int X,Y;
public:
work() //1
{
 X=10;
Y=30;
  }
work(int c)  //2
  {
 X=c;
Y=2*c;
 }
 work (int x,int y) //3
{
```

```
  X=x;
Y=y;
   }
};
int main() {
 work w;          // 10: Call for constructor 1
 work r( 30,60);  // 20: Call for constructor 3
work rr(30);      // 30: Call for constructor 2
}
```

7.8.3 Destructors

The destructor is the counterpart of a constructor. It is a member function that is called automatically when a class object goes out of scope. It has same name as that of the class , proceeded by a tilde (~). A destructor destroys an object. Destructors are declared in the header files of their class, or are inherited from the base class of that class.

a. Characteristics of Destructors:

A destructor has the same features as that of the constructor that are

- Name of the destructor function is same as that of the class they belong to, preceded by tilde (~)
- No return type is required for the destructor function
- Destructor functions are automatically called when the object goes out of scope
- Destructor functions are always defined in the public type access specifier

b. Use of destructors

When an object is created, sufficient amount of memory is allocated to it. When the scope of an object comes to an end, resource allocated to it must be freed so that it can be used elsewhere. This process can be achieved by the use of a destructor. This function is automatically invoked when the scope of an object comes to an end.

Note: In case the programmer does not provide for a destructor manually, the compiler provides a default destructor just as the default constructor.

The definition of the destructor is similar to that of the constructor, but its name is preceded by a tilde ("~"), and it does not have the arguments.

Briefly:

It has the same name as the name of the class in which it is defined,

It is preceded by a tilde

It does not have a return type (not even void)

Example:

```
point:: ~point ( )
{
//statements
}
```

Example: use of destructor

```
# include<iostream>
 # include<conio.h>
using namespace std;
  class trial {
 private:
 int t1,t2;
 public:
 trial (int a,int b)  // constructor function
   {
 t1=a;
 t2=b;
cout<<" This is a trial program";    }

    void output()
   {
 cout<<endl<<" The sum of the variables entered is "<<t1+t2;
  }
  ~trial()   // destructor function
 {
  cout<<" \n The trial program is over";
  }
};
int main()
{
trial t(1000,2000);
t.output();
 getch();
}
```

Destructors are less used compared to constructors. It will be used only if the objects are linked to other objects by pointers. Thus, we can use the destructor to delete the pointers towards these objects.

c. Call of the destructor

The destructor is called differently, according to the object to which it belongs.

- The destructor is called automatically, at the end of the program for the static case

- In the dynamic case, the destructor is called by the keyword “delete”, which allows freeing the memory occupied by the object.

Application Activity 7.7

Part I

1. What are the characteristics of constructor?
2. What is the default constructor? write an example of a program that contain a default constructor

Part II

1. Using comment line, locate the declaration of constructor and destructor in this program:

```
#include<iostream>
using namespace std;
class A
{
        public:
A()
 {
  cout << “Hello “;
 }
~A()
 {
  cout << “welcome to the world of programming”;
 }
};

int main()
{
 A obj1;
}
```

2. Write a C++ program that calculate the sum of two numbers entered by the user, using constructor to receive those numbers and destructor for calculations and display of result.

7.9. Inheritance in C++

Activity 7.8

1. A Father gives his property to his child, father got that properties from child's grandfather, so child is the taker and father is giver, hence father works as a base class and child as derived class.
2. Mother + father both are parent and give some of their properties to child!
3. After receiving the property from his father who received it from child's grandfather, the child also receives other property from his father in law.

Questions:

a. In scenario 1, between father and child, who can be considered as base class and who the derived class is?
b. Draw a diagram that Represents scenario 1
c. Draw a diagram that represents scenario 2
d. Interpret scenario 3 with a diagram

7.9.1 Introduction

Programmers seem to hate coding something from scratch; why code the same thing over and over again? Fortunately, it is possible to reuse code, thanks to one of the key concepts of object-oriented programming. That is inheritance, which is the subject of this chapter.

7.9.2 Definition of Inheritance

- Inheritance is the process by which new classes called *derived* classes are created from existing classes called *base* classes. The derived classes have all the features of the base class and the programmer can choose to add new features specific to the newly created derived class.
- In Object Oriented Programming, Inheritance is the process by which objects of one class acquire the properties and functionality of objects of another class. It supports the concept of hierarchical classification. For example, the bird robin is a part of the class flying bird which is again a part of the class bird.

Creating or deriving a new class using another class as a base is called inheritance in C++. The new class created is called a **Derived class** and the old class used as a base, is called a **Base class** in C++ inheritance terminology. For example, a programmer can create **a *base* class named fruit** and define ***derived* classes as mango, orange, banana, etc**. Each of

these derived classes, (mango, orange, banana, etc.) has all the features of the *base* class (fruit) with additional attributes or features specific to these newly created derived classes. Mango would have its own defined features, orange would have its own defined features, banana would have its own defined features, etc.

Another example, if we define a class 'computer' then it could serve as the base class for defining other classes such as 'laptops', 'desktops' etc..

A derived class is defined by indicating its relationship with the base class in addition to its own details.¶ The General Form of Inheritance:

class*derived-class: access-specifier base-class*

```
{
  ...
  ... //instructions
  ...
};
```

The two points indicate that derived-class is derived from base-class. The access-specifier is optional and if it is present can be private or public.

The mode of visibility by default is private. It indicates if the aspects of the base class are inherited in a private way or publicly.

Parent has the attributse and the methods(the traits)

↑

children need to use some attributes and methods and the methods coming from the parent class

inheritance is also called derivation.

Example:

Suppose, in your school, you want three characters a **maths teacher**, a **footballer** and a **businessman**.

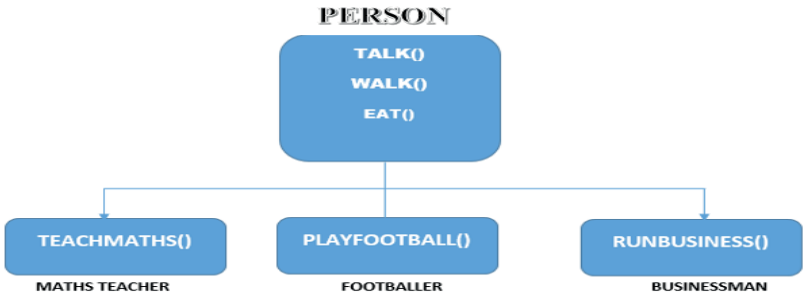
Since, all of the characters are persons, they can walk and talk. However, they also have some special skills. A math's teacher can **teach maths**, a footballer can **play football** and a businessman can **run a business**.

You can individually create three classes who can walk, talk and perform their special skill as shown in the figure below.

In each of the classes, you would be copying the same code for walk and talk for each character. If you want to add a new feature - eat, you need to implement the same code for each character. This can easily become error prone (when copying) and duplicate codes.

It'd be a lot easier if we had a **Person** class with basic features like talk, walk, eat, sleep, and add special skills to those features as per our characters. This is done using inheritance.

Illustrated bellow:



7.9.3. Advantages of Inheritance

1. Reusability:

Inheritance helps the code to be reused in many situations. The base class is defined and once it is compiled, it need not be reworked. Using the concept of inheritance, the programmer can create as many derived classes from the base class as needed while adding specific features to each derived class as needed.

2. Saves Time and Effort:

The above concept of reusability achieved by inheritance saves the programmer time and effort; since the main code written can be reused in various situations as needed.

3. Increases Program Structure which results in greater reliability.

Attention!

Some of the exceptions to be noted in C++ inheritance are as follows:

The constructor and destructor of a base class are not inherited,

The assignment operator is not inherited,

The friend functions and friend classes of the base class are also not inherited.

7.9.4. Inheritance rules

a. class derived_classname: inheritance_ specifier baseclassname

For example, if the *base* class is *student* and the derived class is peter it is specified as:
class student: public peter

The above makes peter have access to both *public* and *protected* variables of base class *student*.

b. Base class

A base class is a class that is created with the intention of deriving other classes from it. It is also called super/parent/old class.

c. Child Class

A child class is a class that was derived from another, that will now be the parent class to it. It is also called sub/derived/inherited class.

The **protected** and **public** variables or members of the base class are all accessible in the derived class, but a private member variable is not accessible by a derived class.

The derived class inherits some or all the features from the parent class. A class can also inherit the properties of more than one class or more than one level.
A derived class with only one parent class is called **simple inheritance**. And that with several base classes is called **multiple inheritance**. In addition, the features of a class can be inherited by more than one class. ¶This process is known as a **hierarchical inheritance**.

The mechanism to derive a class from another derived class is known as **multilevel inheritance**.

Declaration	Commentary
class Circle : public GraphicObject	Simple inheritance: circle derives from GraphicObject in public mode
class GraphicText : public GraphicObject, public Chain	Multiple inheritance: GraphicText derives at the same time from GraphicObject and from Chain and it is, each time in public.
class john : private person	john inherits uniquely from the person class in private.

Table 7. 1. Some examples of inheritance

d. The role of the inheritance modifier and the access modifier: protected

A protected attribute is not accessible outside the class but is accessible from the derived classes. This modifier is thus intermediate between private and public. The following table recapitulates the accesses provided by these three modifiers:

Access Modifier	Visibility in the children classes	Visibility from outside
Private	No	No
Protected	Yes	No
Public	Yes	Yes

Table 7 2 access to the members and modifiers

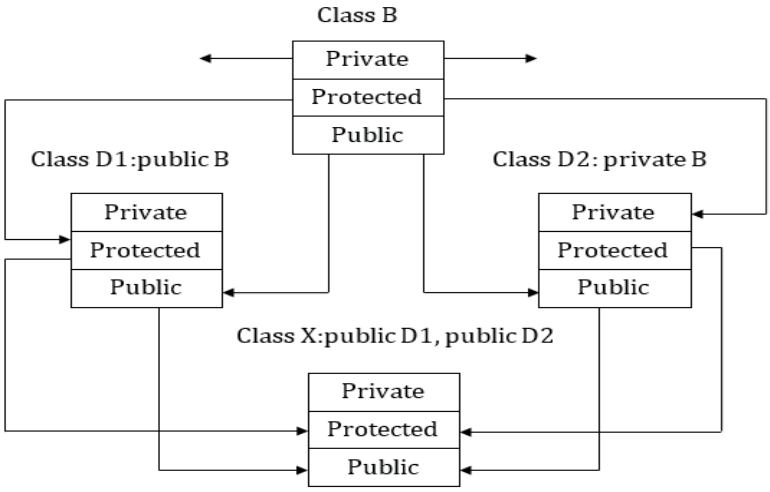
The inheritance modifier can be public or private. It conditions the visibility of the members of the parent class in the derived class. The following table indicates to which access is associated a member with the parent class in the child class according to the inheritance modifier:

Access in the parent class	Access in the child class	
	Public inheritance	Private inheritance
Private	Not accessible	Not accessible
protected	protected	Private
Public	Public	Private

Table 7 3 Effects of the inheritance specifiers on the Table 7. 2 access specifiers

Examples

```
Class ABC: private XYZ //private derivation
{
Members of ABC
}
Class ABC: public XYZ // public derivation
{
Members of ABC
}
Class ABC: XYZ // private derivation by default
{
Members of ABC
```

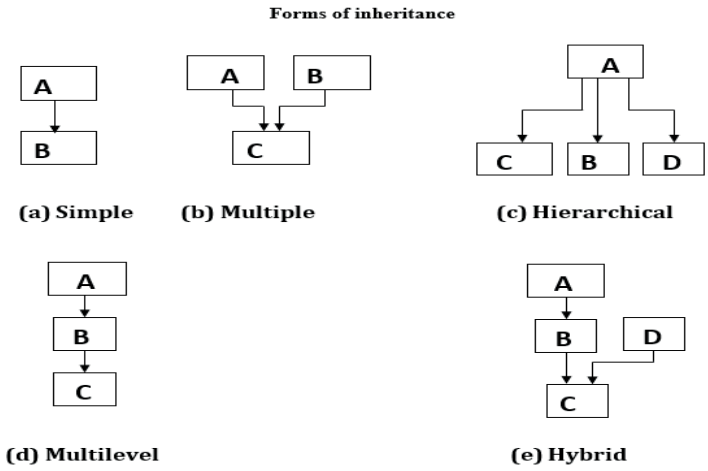


Interpretation of the above sketch diagram

When a base class is inherited using private inheritance specifier by a derived class, the public members of the base class become the private members of the derived class. The result is that no member of the base class is accessible to the objects from the derived class. In addition, when the base class is publicly inherited, ' the public members of the base class become ' the public members of the derived class and thus they are accessible to the objects from the derived class.

In both cases, the private members are not inherited and thus, the private members of a base class will never become the members of its derived class.

7.9.5 1 Types/ Forms of Inheritance



7.9.5.1 Single Inheritance

In this type of inheritance one derived class inherits from only one base class. It is the simplest form of Inheritance.

Example:

```
//a program to add two numbers using inheritance
#include<iostream>
using namespace std;

class AddData       //Base Class
{
    protected:
        int num1, num2;
    public:
        void accept()
        {
            cout<<"\n Enter First Number : ";
            cin>>num1;
            cout<<"\n Enter Second Number : ";
            cin>>num2;
        }
};
class Addition: public AddData  //Class Addition – Derived Class
{
        int sum;
    public:
        void add()
        {
            sum = num1 + num2;
        }
        void display()
        {
            cout<<"\n Addition of Two Numbers : "<<sum;
        }
};
int main()
{
    Addition a;
    a.accept();
    a.add();
    a.display();
    return 0;
}
```

Output:

Enter First Number : 20

Enter Second Number : 30

Addition of Two Numbers: 50

7.9.5.2 Multiple inheritance

Multiple inheritance is the construction in which objects inherit from more than one object type or class. This contrasts with single inheritance, where objects can only inherit from one type or class.

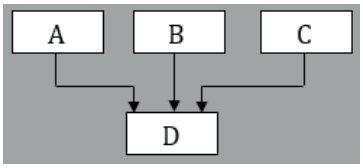
```
#include<iostream.h>
#include<conio.h>
class A
{
public:
void addition(int a,int b)
{
cout<<"Addition:"<<a+b<<endl;
}

};
class B
{
public:
void subtraction(int a,int b)
{
cout<<"Subtraction:"<<a-b<<endl;

}
};
class C
{
public:
void multiplication(int a,int b)
{
cout<<"Multiplication:"<<a*b<<endl;
}

};
class D:public A,public B,public C
{
public:
```

```
void division(int a,int b)
{
cout<<"Division:"<<a/b<<endl;
}
};
int main( )
{
int a,b;
cout<<"a and b?"<<endl;
cin>>a>>b;
D ob;
ob.addition(a,b);
ob.subtraction(a,b);
ob.multiplication(a,b);
ob.division(a,b);
getch( );
}
```



c. Hierarchical Inheritance

In this type of inheritance, multiple derived classes inherits from a single base class.

```
// a program to find the sum and product of two numbers using hierarchical inheritance
#include <iostream>
using namespace std;

class A //single base class
{
  public:
        int x, y;
        void getdata()
        {
          cout << "\nEnter value of x and y:\n"; cin >> x >> y;
        }
};
class B : public A //B is derived from class base A
{
  public:
        void product()
        {
          cout << «\nProduct= « << x * y;
```

```
        }
};
class C : public A //C is also derived from class base A
{
  public:
        void sum()
        {
    cout << "\nSum= " << x + y;
        }
};
int main()
{
  B obj1;      //object of derived class B
  C obj2;      //object of derived class C
  obj1.getdata();
  obj1.product();
  obj2.getdata();
  obj2.sum();
  return 0;
} //end of program
```

Output:

```
Enter value of x and y:
2
3
Product= 6
Enter value of x and y:
2
3
Sum= 5
```

7.9.5.3 Multi-level inheritance

Let us consider an example: let us assume that the result of an exam of the student is stored in the 3 different classes. Class number stores the registration number, class test stores marks obtained in the two subjects and finally the class result contains the total of the points obtained in the exam. The class result can inherit the details, such as the points obtained in the exam and the registration number of the student through the multilevel heritage.

Example:

```
#include<iostream>
using namespace std;
```

```
class number
{
protected:
int reg_number;
public:
void get_number(int);
void display_number(void);
};
void number::get_number(int a)
{
reg_number =a;
}
void number::display_number(void)
{
cout<<"Registration number="<<reg_number<<"\n";
}
class test:public number
{
protected:
float cpp,vb;
public:
void get_marks(float,float);
void display_marks(void);
};
void test::get_marks(float b,float c)
{
cpp=b;
vb=c;
}
void test::display_marks( )
{
cout<<"The marks in cpp="<<cpp <<"\n";
cout<<" The marks in vb="<<vb<<"\n";
}
class result:public test
{
float total ;
public:
void display(void);
};
void result::display( )
{
total=cpp + vb;
```

```
display_number( );
display_marks( );
cout<<"Total marks="<<total<<"\n";
}
int main( )
{
result object ;
int d ;
float e,f ;
cout<< "Enter the registration number"<< "\n";;
cin>>d ;
cout<< "Enter the marks in cpp"<< "\n";
 cin>>e ;
cout<< " Enter the marks in vb"<< "\n";
cin>>f ;
object. get_number(d);
object. get_marks(e,f);
object. display( );
}
```

7.9.5.4 Hybrid Inheritance

There could be situations where we must apply two types or more of inheritance to develop a program.

The following example shows how the hybrid inheritance looks like.¶

Implementation of hybrid inheritance

```
#include <iostream>
#include<conio.h>
using namespace std;
class identification
{
private:
char name[25];
int nbr;
public:
void getident(void)
{
cout<<"What is your name?"<<endl;
cin>>name;
cout<<"and your number?"<<endl;
cin>>nbr;
```

```
}
void displayident(void)
{
cout<<nbr<<"\t"<<name;
}
};

class theory:public identification
{
protected:
int math,fr;
public:
void gettheory(void)
{
cout<<"Enter marks in mathematics and french"<<endl;
cin>>math>>fr;
}

void displaytheory(void)
{
cout<<»\t»<<math<<»\t»<<fr;
}
};

class practice
{
protected:
int maint,prog;
public:
void getpractice(void)
{
cout<<"Enter marks in maintenance and in programming"<<endl;
cin>>maint>>prog;
}
void displaypractice(void)
{
cout<<»\t»<<maint<<»\t»<<prog;
}
};

class result:public theory,public practice
{
private:
```

```
int total;
public:
void displayresult(void)
{
total=math+fr+maint+prog;
cout<<»\t»<<total<<endl;
}
};
int main( )
{
result ob[29];
for(int a=0;a<5;a++)
{
ob[a].getident( );
ob[a].gettheory( );
ob[a].getpractice( );
}

cout<<"_____"<<
endl;
cout<<endl<<"Nbr\tName\tMath\tFr\tMaint\tProg\tTotal"<<endl;
cout<<"----------"<<endl;

for(int a=0;a<5;a++)
{
ob[a].displayident( );
ob[a].displaytheory( );
ob[a].displaypractice( );
ob[a].displayresult( );
}
cout<<endl<<"----------";
getch( );
}
```

Application activity 7.8

1. Consider the following declaration and answer the questions given below:

```
class school
{
        int H;
        protected :
                int S;
        public :
                void INPUT ()
                {}
                void OUT()
                {}
};
class Teacher : private school
{
        int T;
 protected :

                int U;
        public :
                void INDATA()
                {}
                void OUTDATA()
                {}
};
class student : public Teacher
{
        int M;
        public :
                void DISP( void )
                {}
};
```

   i. Name the base class and derived class of the class teacher
   ii. Name the data member(s) that can be accessed from function DISP().
   iii. Name the member function(s), which can be accessed from the objects of class student.
   iv. Is the member function OUT() accessible by the object of the class teacher?

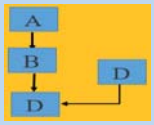
2. Write a C++ program that execute contains three classes A,B and C where A is a base class of B and B is a base class of C, then use an object of class C to Access public members of class A

3. What is the output of the following program?

```
#include <iostream>
using namespace std;
class polygon
{
protected:
int width, height;
public:
void set_values (int a, int b)
{
width = a; height = b;}
};
  class output1
{
public:
void output (int i);
};
void output1::output (int i)
{
     cout << i << endl;
}
class rectangle: public polygon, public output1
{
public:
int area ()
{
return (width * height);
}
};
class triangle: public polygon, public output1
{
public:
int area ()
{
return (width * height / 2);
}
};
int main ()
{
```

```
rectangle rect;
triangle trgl;
rect.set_values (4, 5);
trgl.set_values (4, 5);
rect.output (rect.area());
trgl.output (trgl.area());
return 0;
}
```

1. Write the syntax code of the following diagram



END OF UNIT ASSESSMENT

1. Define the following terms in relation to object oriented programming:
   a. Class
   b. Encapsulation
   c. Object
   d. Inheritance
2. How can you declare a class and write its syntax of declaration?
3. Write a C++ program using class called Rectangle and an object called rect. This class should have four members: two data members of type int with private access and two member functions with public access: set_values() and area().Set_values() to initialize the values of rectangle and area() to return the area of rectangle.
4. Give the output for the following program

A.

```
#include<iostream>
 using namespace std;
 class account
 {
 int balance,accno;
 public:
 account()
 {balance=0;}
 account(account &s)
 {
  balance = ++s.balance;
  accno=0;
  }
  void assign( int u)
  {
  balance = 110; accno = u;
  }
  void disp()
  {
 cout<<"Balance"<<balance<<endl;
  cout<<"Acc no."<<accno<<endl;
  }
  };
 int main()
 {
 account a1;
 a1.assign(9);
 a1.disp();
 account a2(a1);
 a2.disp();
 account a3(a2);
 int y=10;
 a3.assign(-y);
 a3.disp();
 return 0;
 }
```

B.

```
# include <iostream>
 # include <conio.h>
 using namespace std;
  class Trial
 {
  int a, b;
  public:
  Trial()
  {
  a=0;b=0;
  }
   void disp()
          {
          cout<<a<<b<<endl;
          }
          void raise()
          {
          a+=10;
          b+=10;
          }
 void down()
 {
 a-=5;
 b-=5;
 } };

 // Main Function
 int main() {
 Trial T;
 T.disp();
 T.raise();
 T.disp();
 T.down();
 T.disp();
 }
```

5. The country A has 50M inhabitants, and its population grows 3% per year. The country B, 70M and grows 2% per year. Tell in how many years A will surpass B.
6. Define a class student with the following specification
   Private members of class student:
   admno integer
   sname 20 character
   eng. math, science float
   total float
   ctotal() a function to calculate eng + math + science with float return type.
   Public member function of class student:
   Takedata() Function to accept values for admno, sname, eng, science and invoke ctotal() to calculate total.
   Showdata() Function to display all the data members on the screen.
7. Define a class BOOK with the following specifications :
   Private members of the class BOOK are:
   BOOK NO integer type
   BOOKTITLE 20 characters

PRICE float (price per copy)
TOTAL_COST() A function to calculate the total cost for N number of copies where N is passed to the function as argument.
Public members of the class BOOK are
INPUT() function to read BOOK_NO. BOOKTITLE, PRICE
PURCHASE() function to ask the user to input the number of copies to be purchased. It invokes TOTAL_COST() and prints the total cost to be paid by the user.
Note: You are also required to give detailed function definitions.

8. Write the definition for a class called **Distance** that has data member feet as integer and inches as float. The class has the following member functions:
void set(int, float) to give value to object
void disp() to display distance in feet and inches
Distance add(Distance) to sum two distances & return distance
1. Write the definitions for each of the above member functions.
2. Write main function to create three Distance objects. Set the value in two objects and call add() to calculate sum and assign it in third object. Display all distances.

9. Write the definition for a class called **time** that has hours and minutes as integer. The class has the following member functions:
void settime(int, int) to set the specified value in object
void showtime() to display time object
time sum(time) to sum two time object & return time
1. Write the definitions for each of the above member functions.
2. Write main function to create three time objects. Set the value in two objects and call sum() to calculate sum and assign it in third object. Display all time objects.

10. Find the errors in the following programs.

```
class circle
 {
  private
float radius;
public:
void getdata();
{
cout<<" Enter the radius ";
cout<<radius;
}
 float area();
}
    circle:: float area()
{
  return (3.14 *radius*radius);
   }
```

11. Answer the questions (i) to (iv) on the basis of the following code:
    i. Write the names of data members which are accessible from objects belonging to class 'branch'.

ii. Write the names of all the member functions which are accessible from objects belonging to the class employee.

iii. Write the names of all the members which are accessible from member functions of class 'employee'.

```
class company
{
char website[12];
char name[20];

protected:
int noemployees;

public:
company();
void registration();
void status();
};
class branch:public company
{
long nocomputer;
protected:
float expense;
public:
branch();
void enter();
void show();
};
class employee:private branch
{
int eno;
char ename[20];
float salary;

public:
employee();
void joining();
void showdetail();
};
```

UNIT 8

Introduction to Visual basic

UNIT8: Introduction to Visual basic

KeyUnitCompetency.

To be able to describe a Visual Basic Integrated Development (VB-IDE) and write a program in Visual Basic.

INTRODUCTORY ACTIVITY

Read the bellow scenario:

ABC hotel provides different services to its customers such as business center, room services, restaurant and others. For the room services, front desk office receives clients and information is collected as shown below:

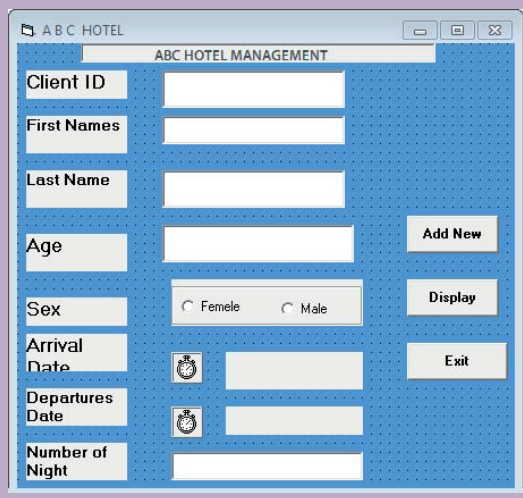


Figure 8.0 management system of ABC Hotel

Observe the above picture and answer the following questions:

1. What do you think of the above figure?
2. Explain why ABC hotel needs to collect data of each client
3. Investigate why the figure hold these words such :Add new ,Display and Exit
4. Which programming language would use to design the above figure?
5. Can you try to design the above figure?

In programming, the above scenario requires specific programming language that enables us to enter data using forms and save the data in database to receive them whether is a need. This unit will enable to build a simple desktop application for a real life situation.

8.1. Understanding Visual Basic.

LEARNING ACTIVITY. 8.1

1. Click on start button, All programs, Microsoft Visual basic 6.0, Microsoft visual basic 6.0. What is the name of the window that has opened?
2. Make double click on standard EXE. What do you observe?
3. Differentiate desktop and web applications.

8.1.0. General introduction

VISUAL BASIC is a high level and **Event-driven Programming Language** which evolved from the earlier Desk Opening System (DOS) version called BASIC. (**BASIC** means: **B**eginners' **A**ll-purpose **S**ymbolic **I**nstruction **C**ode).

Visual basic for DOS and Visual Basic for Windows were introduced in 1991 and evolved through the years to give Visual basic 3.0(in 1993), Visual basic 4.0 (in 1995) Visual basic 5.0 (in 1996) and now we use Visual basic 6.0 version released in 1998.

Visual basic 6.0 has the following advantages:

- It is easier for the user to minimize code writing.
- The user will become more familiar with visual approach for other visual languages.
- It provides Input box and Output box as interactive windows with user.
- It is very easy program language compare with other.

8.1.1. Definition of terms

d. Graphical User Interface.

A graphical user interface (GUI) is an interface through which a user interacts with electronic devices such as computers, hand-held devices and other appliances. GUI representations are manipulated by a pointing device such as a mouse, trackball, stylus, or a finger on a touch screen.

The need for GUI became apparent because the first human/computer text interface was through keyboard text creation by what is called a prompt (or DOS prompt). Commands were typed on a keyboard at the DOS prompt to initiate responses from a computer. The use of these commands and the need for exact spelling made this interface difficult to use and inefficient.

A GUI uses a combination of technologies and devices to provide a platform that user can interact with, for the tasks of gathering and producing information.

Graphical user interface (GUI) is different from command line interface (CLI) or command language interpreter as command line interface (CLI) enables users to type commands in a terminal or console window to interact with an operating system. Users respond to a visual prompt by typing a command on a specified line (MLI), and receive a response back from the system. Users type a command or series of commands for each task they want to perform.

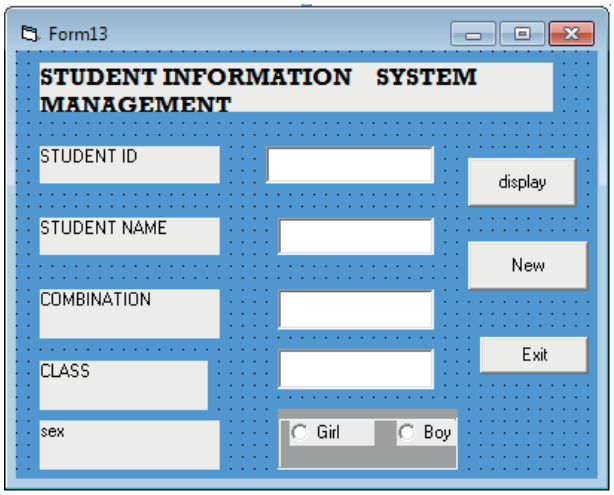


Figure 8.1. Graphical User Interface (GUI) created using VB

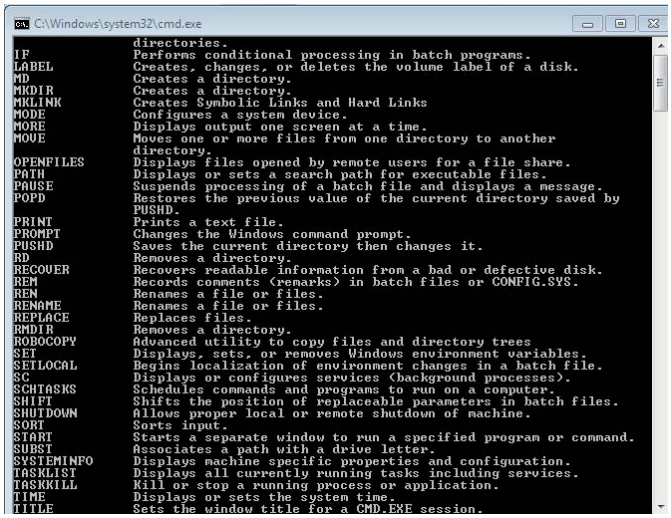


Figure 8.2 Command Line Interface (CLI) example (DOS

e. Desktop application

A desktop application is a computer program that runs locally on a computer device, such as desktop or laptop computer, in contrast to a web application, which is delivered to a local device over the Internet through browser from a remote server. Different user environments can impact whether a desktop or a web application is the best solution for your needs.

Difference between desktop andweb applications.

Desktop applications

- They must be developed for and installed on a particular operating system.
- Have strict hardware requirements that must be met to ensure that they function correctly.
- Updates to the applications must be applied by the user directly to their installation, and may require hardware upgrades or other changes in order to work.

Web applications

- A web application is any computer program that performs a specific function by using a web browser.
- The user accesses the application using the web browser and works with resources available over the internet, including storage and CPU processing power.
- This approach allows for "thin clients" (machines with limited hardware capabilities) to provide access to complex applications delivered from a centralized infrastructure.

f. Event oriented programming.

Event oriented programming is a paradigm in which the flow of program is determined by events, such as user actions (mouse clicks, key presses), sensor outputs or messages from other programs is common used in graphical user interfaces and other applications like Web applications, JavaScript, C#.

Other types of programming paradigm are: C, C++

- Procedural programming used by Basic, Fortran and COBAL
- Declarative programming used by Prolog

Event oriented programming using Visual Basic

Visual Basic is Event oriented programming because of the following

- The programmer needs to write code that performs some tasks in response to certain events.
- Has events that occur by mouse clicking and moving or keyboard stokes (Some of the events are load, click, double click, drag and drop, pressing the keys and more.)
- Focus on the use of Graphical User Interface.

- The events usually comply but not limited to the user's inputs.

8.1.2. The Features of Visual Basic

Visual Basic has the following features:

1. **Data Access Features**: this allows programmers to develop database front end applications and server side components for most popular database formats including MS SQL and other Databases.
2. **Active X Technologies:** that allows programmers to use the functionality provided by other applications, such as MS Office and other windows applications.
3. **Internet capabilities** make it easy to provide access to documents and applications across internet server applications.
4. **Your finished application is a true executable** (.exe) file that uses a visual Basic Virtual Machine that you can freely distribute.

8.1.3 Standard EXE Visual Basic Application

Using Visual Basic one can develop one of these main projects as indicated in the screen below:

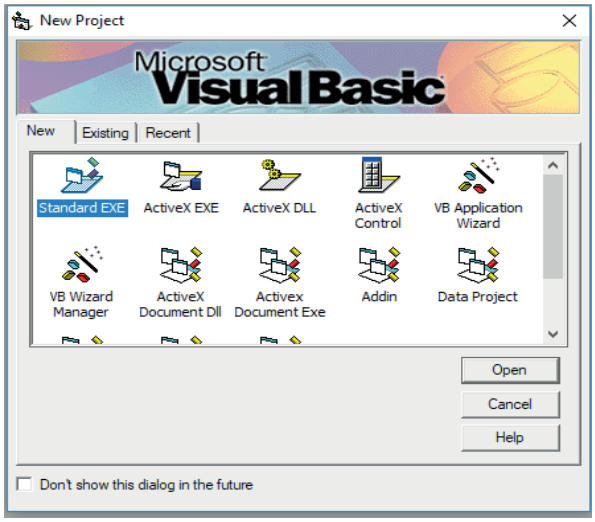


Figure 8.3 The highlighted is standard EXE project which we are emphasising in this unit.

1. **Standard Exe project** is a typical application in which can use the database manipulation.

A standard exe application is one that is created using Standard EXE project. It is the most widely used Project type using VB6. Standard EXE application is normally the most widely used among the available Project types in Visual Basic. Stand-alone

programs have an .EXE file extension. A standard EXE application is normally used when you want to develop a stand-alone application. Examples include calculators, text editors, and other similar applications.

8.1.4. Starting VB 6.0

Opening application of visual basic 6.0

On start up, Visual basic 6.0 will display a dialog box, and one can choose to start a new project, open a new existing project, or select a list of recently opened programs.

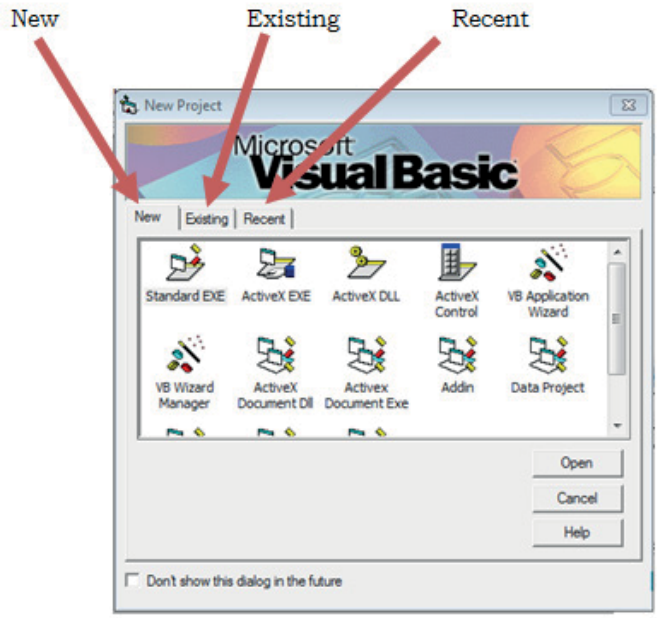


Figure 8.4. Dialog Box of VB 6.0 for project creation.

Application activities (8.1)

1. With examples, discuss the difference between the following:
   a. Graphical User Interface and Command User Interface.
   b. Desktop application and Online application.
2. State the process followed to start up Visual basic 6.0.

8.2 Visual Basic Standard EXE Integrated Development Environment (VB-IDE)

Learning Activity 8.2

1. Observe the figure and name the objects labeled A, B, C, D, E and F

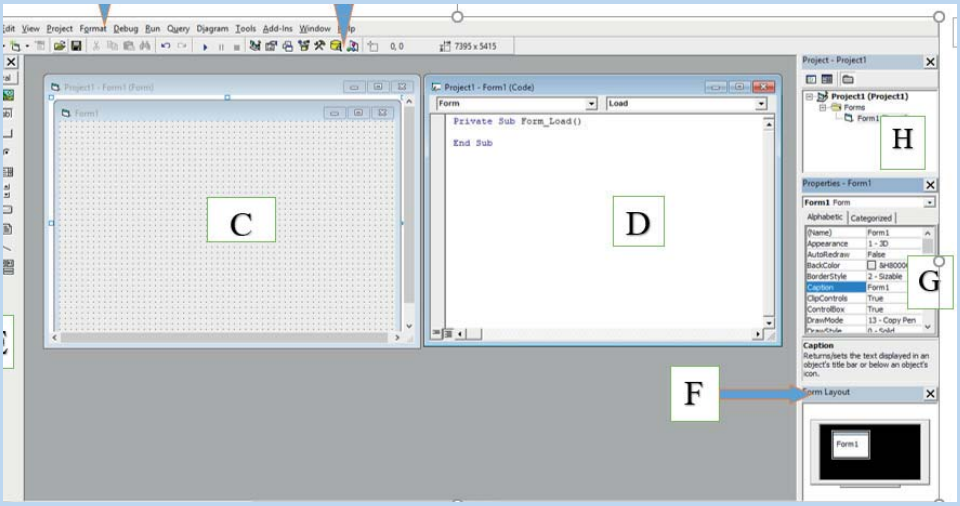


Figure 8.5 visual basic integrated development environment.

a) Visual basic Integrated Development Environment describes the interface and environment that we use to create our applications in VB.

It is called integrated because we can access virtually all of the development tools that we need from one screen called interface. The IDE is also commonly referred to as the design environment, or the program.

Project description in standard exe. Project window.

To design a form window in Visual basic integrated development environment, first install Microsoft visual basic 6.0 into PC, and follow the following in order to use it.

Start>all program> Microsoft visual basic 6.0> Dialog Box >Standard EXE> formwindow appear.From the **New Project window** we can select new, existing and recent project.**The Standard EXE Project** looks like this:

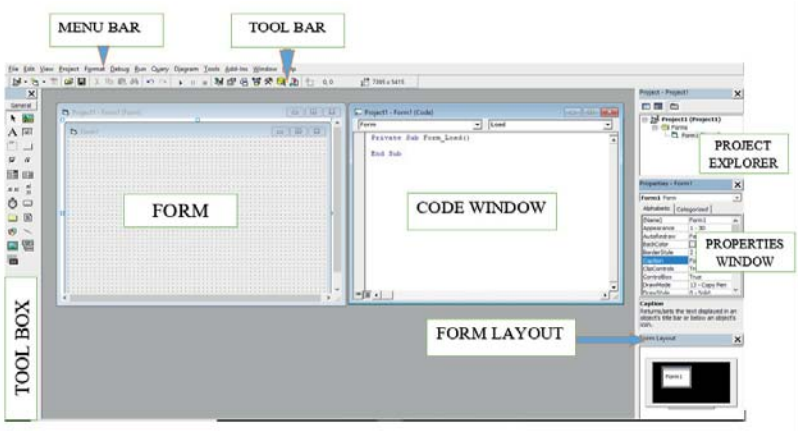


Figure 8.6 components of VB 6.0 Standard EXE IDE.

This is the main IDE window of Visual Basic. The following are components of the standard exe.windows.

8.2.1 Tool Box

Items of tool box are used to design the application interface. These items are called **Controls** and are shown in the image below.

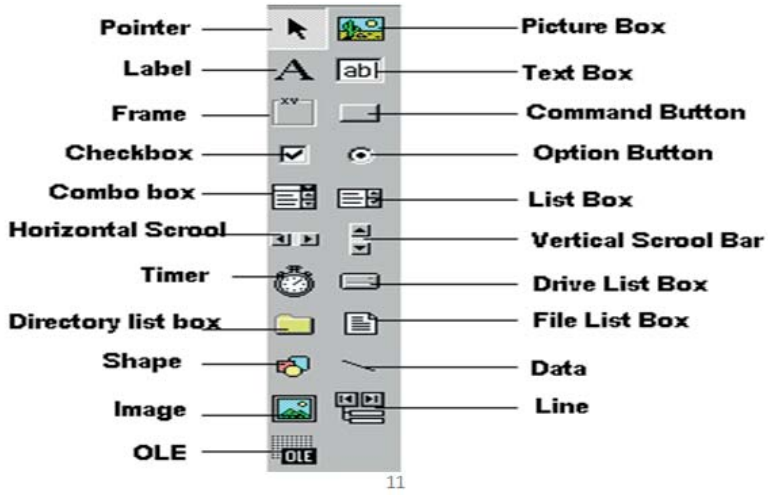


Figure 8.7 VB 6.0 toolbox

8.2.2 Form window

The form window is the window or background, where the user can design his form using various controls from the toolbox

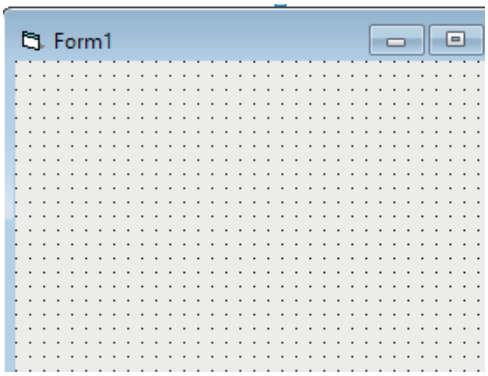


Figure 8.8. Form window

8.2.3 Code window or VB Editor

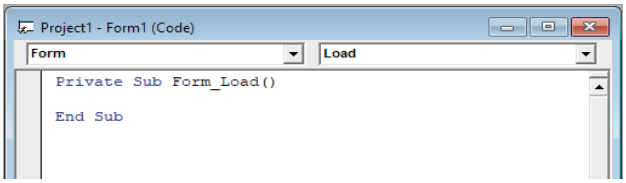


Figure 8.8 Code window

Each standard form has a code window in which the user can write to direct the behavior of a control. You open the code window by double clicking on a form or a control. If you double click a form, you will be taken to a procedure for the form, but once the code window is open, you can go to any procedure for any object on the selected form.

The codes are of two Categories:

Declaration is written before any procedure in the code window.
Statements. The user selects the required event then code statements are written inside these event procedures.

8.2.4 Project explorer

Contains forms related to your application, it also contains code modules and classes.

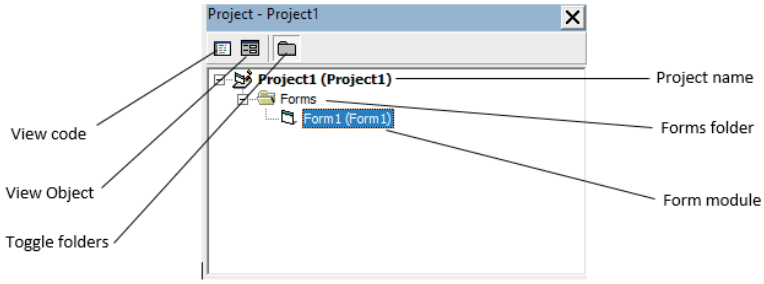


Figure 8.9 Project explorer windows

8.2.5 Properties Window

Properties are the attribute of controls. Every object has properties, for example a Pen, has its Color, Metal Type, Ink Color, Type etc. In the same manner every control in VB has many properties which are applied by using the property fields in *property box* . Some properties are read only, which means the values of such properties can't change using code, while others are Read and Write. You can move Property window any side of VB IDE Window, can appeal using tool bar or by Pressing F4.

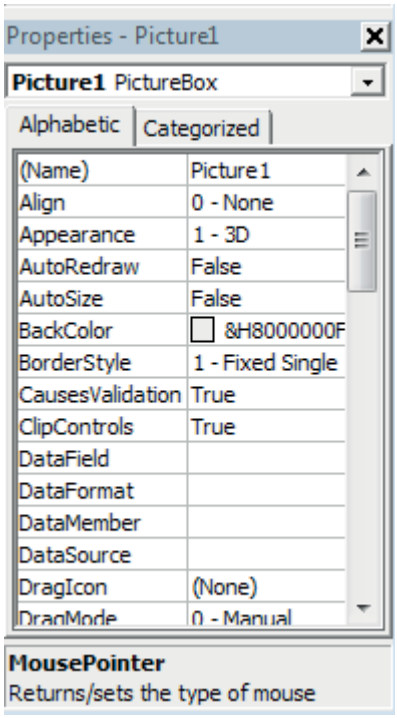


Figure 8.10. Properties Window

It is also possible to change the properties at runtime to give special effects such as change of color, shape, animation effect and so on.

8.2.6 Form Layout Window

The Form Layout window is a visual design tool which is used to control the placement of the forms in the windows environment when they are executed especially when you have more than one form in your program,

To position a form in the Form Layout window, simply drag the miniature form to the desired location in the window.

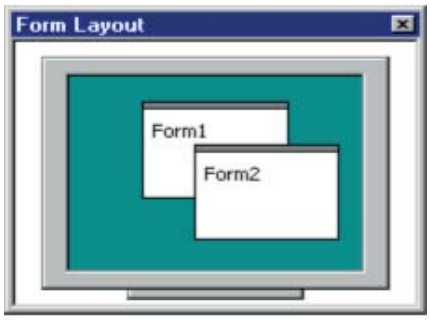


Figure 8.11 Form Layout Window

8.2.7 Menu Bar

This is where you can select actions to perform on all your project files and to access help. When a project is open extra menus of project, build and data, are shown in addition to the default menu selection of File, edit, View, Debug, tools, window and Help.

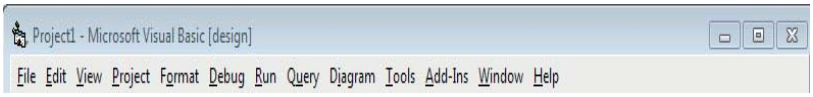


Figure 8.12 menu Bar for standard exe. Project.

8.2.8 Toolbar:it gives easy access to the menu-bar you use frequently

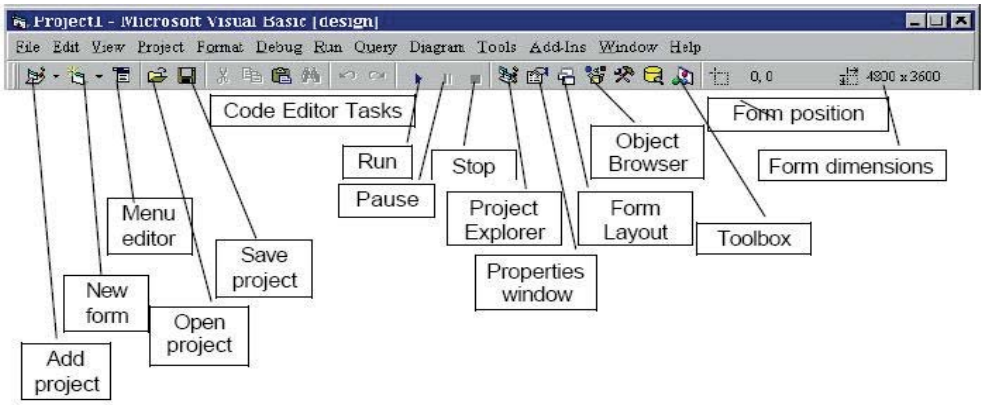


Figure 8.13 Toolbar

Application activities 8.2

1. Name five items on VB toolbox.
2. List two boxes that make up Visual Basic IDE?

8.3 Visual basic controls

a) Uses of properties window

Learning activities (8.3.1)

1. Design a form and call it "NDI UMUNYARWANDA" and have the three labels with three textbox. As well as the properties of label are:

    - **Name:** lblfirstname, lbllastname, and lblnationarity respectively.
    - **Appearance:** run time with 3D effects.
    - **Caption:** First Name, last Name and Nationality. Respectively
    - **Font:** (font face **Bold**, font style **Time New roman**, size **14** and effects)
    - **Forecolor:** It returns foreground color of a label
    - **Height**: 1000.
    - **Width**: 2000
    - **Backcolor:** Active border.
    - **Borderstyle:** Fixed Single-With border)
    - **Backstyle:** Opaque
    - And the textbox as
    - **Name:** txtfirstname, txtlastname, and txtnationarity respectively.
    - **Caption:** Null in all
    - **Font:** (font face **Bold**, font style **Time New roman**, size **14** and effects)
    - **Height**: 730.
    - **Width**: 3000

8.3.1 Definition of form.

Form is the shape, visual appearance, or configuration of an object. In a wider sense, the **form** is the way something is or happens. **Form** may also refer to: **Form** (document), a document (printed or electronic) with spaces in which to write or enter data.

8.3.2 How to design a form in Visual Basic

To design a form in visual basic use the controls which are available in the Tool Box in the left side of the project window.

Use the following steps for each control that you want to add on the form:

Step 1: In the Tool Box, choose a control that you want andClick on it.

Step 2: Drag and draw a control in the form using mouse.

Step 3: Change state for Specific control like size etc.

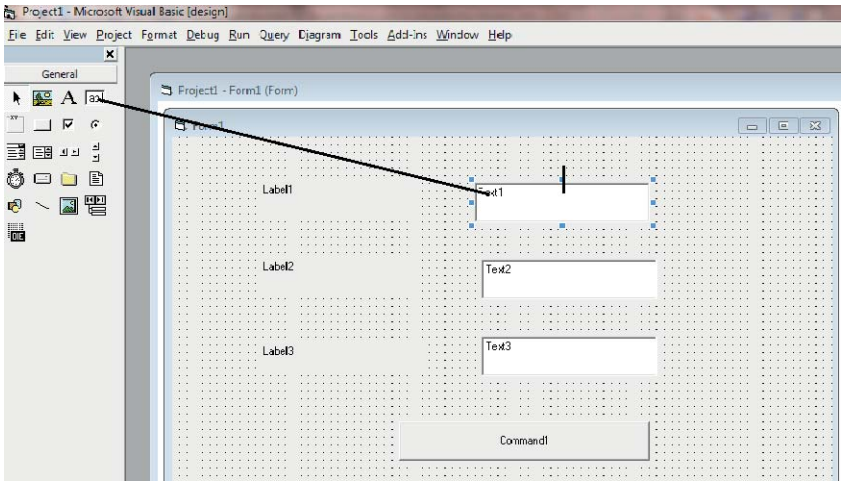


Figure 8.14 Adding a Text Box to a form.

8.3.3 properties Window

Definition of Properties Window.

Properties Window is used to change the state of each control using particular properties associated with each control.

How to use Properties Window.

Before writing an event procedure for the control to response to a user's input, you have to set certain properties for the control to determine its appearance and how it will work with the event procedure. You can set the properties of the controls in the properties window or at runtime.

Consider the following picture

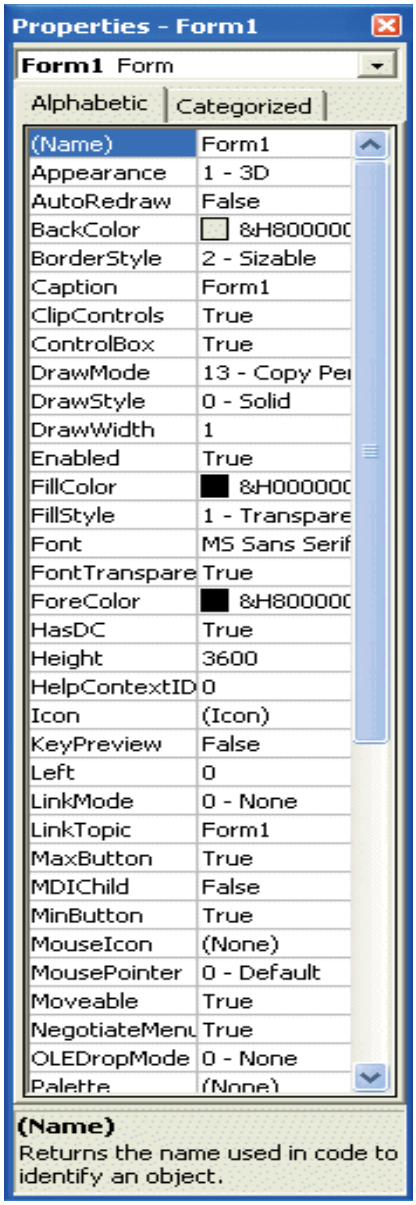


Figure 8.15 ***the Properties Window***

Figure *8.15* above is a typical properties window for a form. You can rename the form caption to any name that you like best. In the properties window, the item appears at the top part is the object currently selected (in Figure *8.15*, the object selected is Form1). At the bottom part, the items listed in the left column represent the names of various properties associated with the selected object while the items listed in the right column represent the states of the properties. Properties can be set by highlighting the items in the right column then change them by typing or selecting the options available. For example, in order to change the caption, just highlight Form1 under the name Caption and change it to other names. You may also try to

alter the appearance of the form by setting it to 3D or flat. Other things you can do are to change its foreground and background color, change the font type and font size, enable or disable minimize and maximize buttons and etc.

Application activity 8.3.1

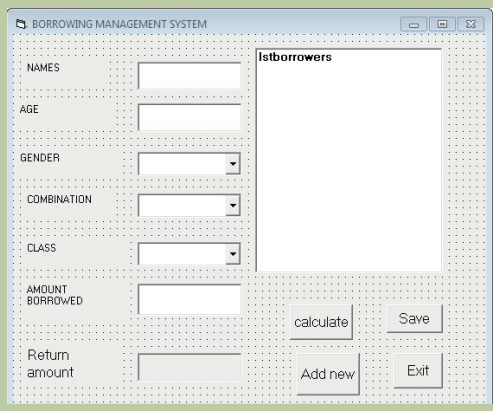
Q1. Which property used to change state of a control?

a) ***Design a form using a control.***

Learning activities (8.3.2)

Observe the figure below:

Using the toolbox design the above form.



- **A control:** is a tool you use to create objects on a Visual Basic form. You select controls from the toolbox and use the mouse to draw objects on a form. You use most controls to create user interface elements, such as command buttons, image boxes, and list boxes.
- **Properties of a control.**

Before writing an event procedure for the control to respond to an event, you have to set certain properties for the control to determine its appearance and how it will work with the event procedure. You can set the properties of the controls in the properties window or at design time.

Other things you can do are to change its foreground and background color, change the font type and font size, enable or disable minimize and maximize buttons etc.

Here are the important points about setting up control properties:

- Caption Property of a control should be clearly defined so that a user knows what to do with control
- Use a meaningful name for property as it is easier to write and read the event procedure and easier to debug or modify the program later.
- One more important property is whether to make the control enabled or not.
- You should also consider making the control visible or invisible at design time, or when should it become visible or invisible.

8.3.4 Common controls used in Visual Basic:

a. Form Form

Form is used when you start Visual Basic, a default form (Form1) with a standard grid (a window consisting of regularly spaced dots) appears in a pane called the Form window. You can use the Form window grid to **create the user interface** and to line up interface elements.

Steps to create Form

Step 1: Click Project on menu and choose **Add form** from the list of options

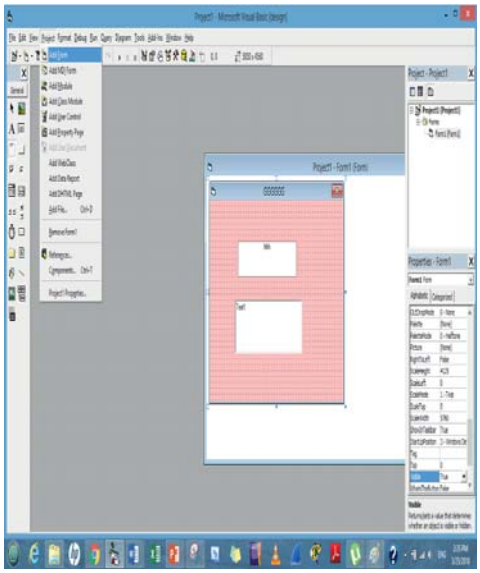


Figure 8.16 adding a form in project.

Step 2: Click Form and click Open

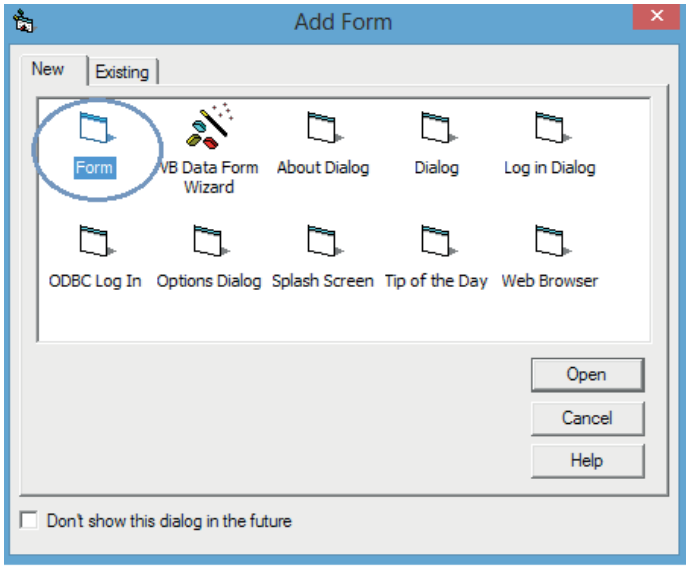


Figure 8.17 diolog box to add a form.

Step 3: Set the Properties for a form

- **Name:** Enter the name of the form. It returns the name used to identify form.
- **Enabled:** form controls can be enabled or disabled on initial execution with the enabled property.
- **Appearance:** It sets whether or not object is painted at run time with 3D effects.
- **Caption:** Enter the caption associated with the form. It sets the text displayed as the title of the form.
- **Backcolor:** Choose background color of textbox .It sets the background color used to display textbox.
- **Borderstyle:** Choose border style of the form
- **Fillcolor:** It sets the colour used to fill in shapes, circles and boxes.
- **Fillstyle:** It returns the fill style of a shape
- **Backstyle:** Choose background style; Opaque or transparent
- **Font:** It returns the font of a form.
- **Scaleheight:** It sets the number of units for the vertical measurement of form.
- **Scaleleft:** It returns the horizontal coordinates for the left edge of form.
- **Visible** shows or hide a control on a form. It sets value that determines whether an object is visible or hidden

b. Label

Label 

The label is a very useful control for Visual Basic, as it is not only used to provide instructions and guides to the users, it can also be used to display outputs. One of its most important properties is **Caption**. Using the syntax **label.Caption**, it can display text and numeric data. You can change its caption in the properties window and also at runtime.

The following are the steps to add a label on the form.

Step 1: Click on the label icon  on the toolbox

Step 2: Drag and drop a label control in the form

Step 3: Set Properties for label

- **Name:** Enter the name of the label control. It returns the name used to identify object.
- **Alignment**: It returns/sets the alignment of a label
- **Appearance:** It sets whether or not object is painted at run time with 3D effects.
- **Caption:** Enter the caption associated with the label. It sets the text displayed on the form.
- **Font:** It returns the font of object (font face, font style, size and effects)
- **Forecolor:** It returns foreground color of a label
- **Height**: Enter the height of the label.
- **Backcolor:** Choose background color. It sets the background color used to display label.
- **Borderstyle:** Choose border style of the label(None-no border or Fixed Single-With border)
- **Backstyle:** Choose background style; Opaque or transparent

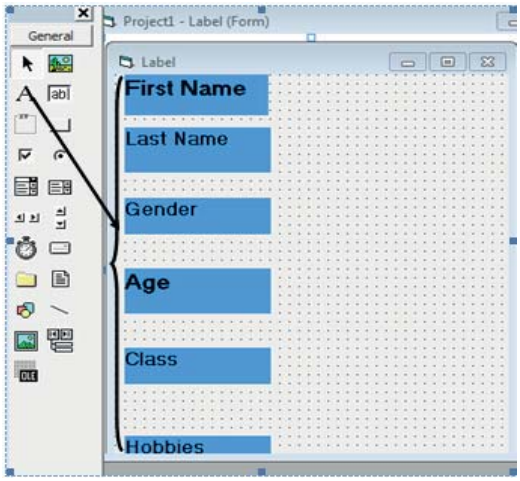


Figure 8.18 Labels drawn on form

c. Text Box

The text box is the standard control that is used to receive input from the user as well as to display the output. It can handle string (text) and numeric data but not images or pictures. String in a text box can be converted to a numeric data by using the function Val(text).

. Steps to add Textbox

Step 1: Click on the Textbox icon  on the toolbox

Step 2: Drag and drop a Textbox control in the form

Step 3: Set Properties for Textbox such as Name, alignment, appearance, backcolor etc.

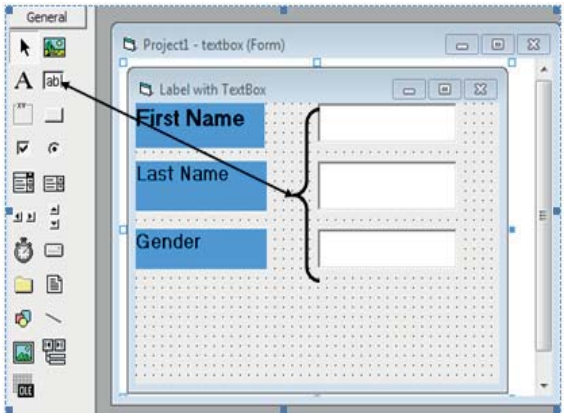


Figure 8.19 Textboxes drawn on form

d. Frame

If you want to create a group of controls that work together, you must first create a frame for the controls. (To do this, use Frame, a Visual Basic toolbox control.) Next, place your controls inside the frame so that they can be processed as a group in your program code and moved as a group along with the frame

Steps to add Frame:

Step 1: Click on the Frame icon on the toolbox

Step 2: Drag and drop a Frame control in the form

Step 3: **Set Properties for Frame: on this step, you set preferred properties**

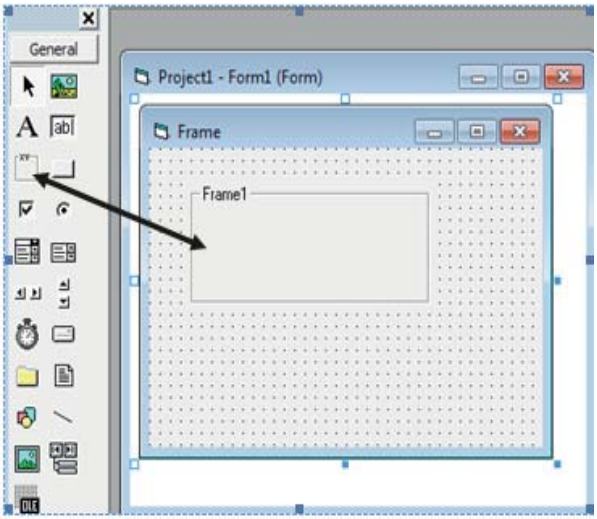


Figure 8.20 Frame on form

e. Radio Buttons/ Option Buttons

The Option Box control lets the user selects one of the choices. However, two or more Option Boxes must work together because as one of the Option Boxes is selected, the other Option Boxes will be unselected. In fact, only one Option Box can be selected at one time. In computer terminology we call this 'mutual exclusion'. Therefore, two option box controls are said to be mutually exclusive of each other. When an option box is selected, its value is set to "True" and when it is unselected; its value is set to "False".

Steps to add Option Button

Step 1: Click on the OptionButton icon on the toolbox

Step 2: Drag and drop aOptionButton control in the form

Step 3: Set Properties for Option Button

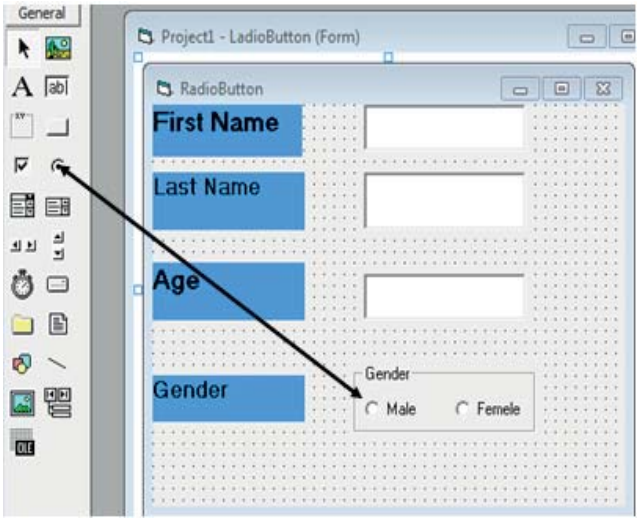


Figure 8.21 Option buttons grouped in a frame

f. Check box control

The Check Box control lets the user to select or unselect an option. When the Check Box is checked, its value is set to 1 and when it is unchecked, the value is set to 0. You can include the statements Check1.Value=1 to mark the Check Box and Check1. Value=0 unmark the Check Box, and use them to initiate certain actions. For example, the program will change the background color of the form to red when the check box is unchecked and it will change to blue when the check box is checked.

Steps to add CheckBox

Step 1: Click on the CheckBox icon on the toolbox

Step 2: Drag and drop a CheckBox control in the form

Step 3: Set Properties for CheckBox

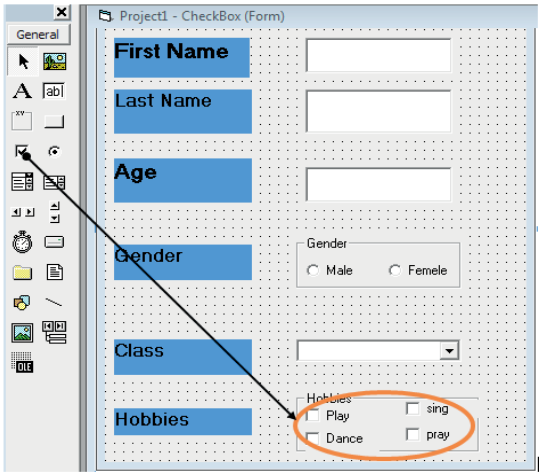


Figure8.22 Check boxes

g. Command button

Command button is a very important control as it is used to execute commands. It displays an illusion that the button is pressed when the user clicks on it. The most common event associated with the command button is the Click event.

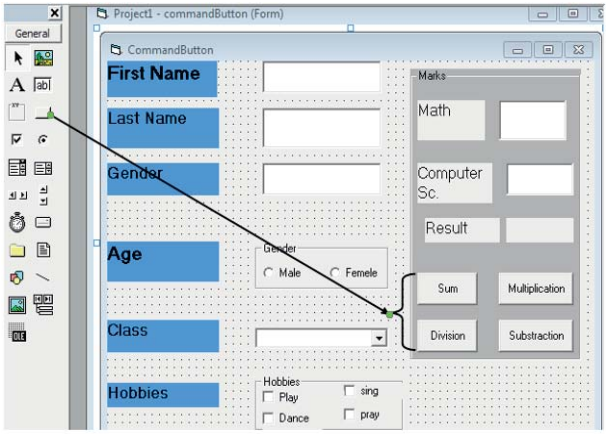


Figure 8.23 Command buttons

h. List Box

The function of the List Box is to present a list of items where the user can click and select the items from the list. In order to add items to the list, we can use the **AddItem** method.

List Box contains a list of options from which user can choose. In windows the **Font List box** is an example of the use of a list box. The Selected item in a ListBox is

given by the *Text**property.*** The ***sorted*** property determines whether the items in the list box will be sorted or not.

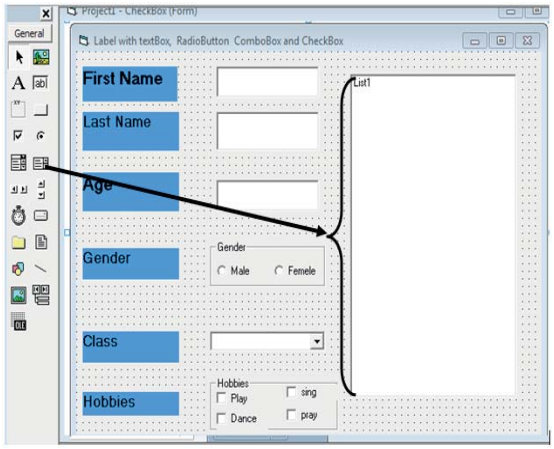


Figure 8.24 list box in a form.

Adding items to list

a) Change property list from properties window: Open the list properties window and choose list property. Write a list of items to appear in the list box

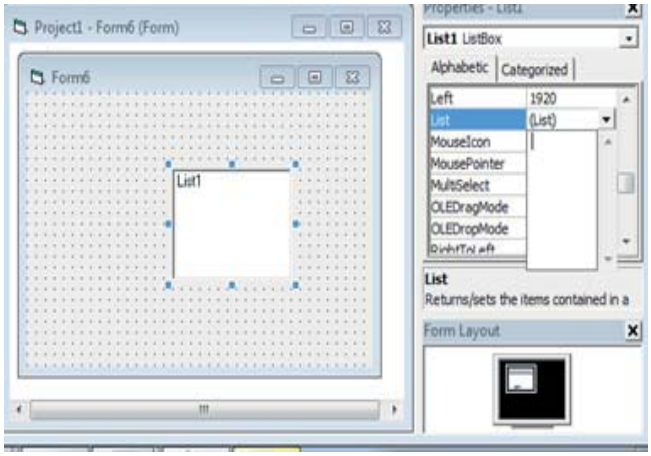


Figure 8.25 Add items to list.

i. Combo Box

The function of the Combo Box is also to present a list of items where the user can click and select the items from the list. However, the user needs to click on the small arrowhead on the right of the combo box to see the items which are presented in a drop-down list. In order to add items to the list, you can also use the **AddItem** method.

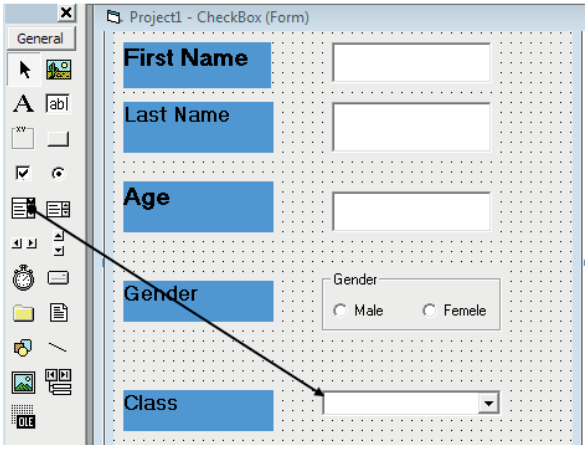


Figure 8.26 Combo box

j. Picture Box

The Picture Box is one of the controls that used to handle graphics. You can load a picture at design phase by clicking on the picture item in the properties window and select the picture from the selected folder. You can also load the picture at runtime using the **LoadPicture** method. For example, the statement will load the picture grape.gif into the picture box.

```
Picture1.Picture=LoadPicture("C:\Images\grape.gif")
```

Picture Box is a control used to display images on a VB page. The Picture Box control also supports few functionality of generating advanced drawing.

k. Image box

The Image Box is another control that handles images and pictures. It functions almost identically to the picture box. However, **there is one major difference, the image in an Image Box is stretchable, which means it can be resized**. This feature is not available in the Picture Box. The syntax is as follows:

l. Timer

When we need to perform tasks at regular interval we can use Timer**.** Open a new project and place a timer object on your form. Then locate a label at the center of theform and adjust the size as shown in the figure below. For such a program a better look will beestablished by sizing your form as a pop up window.

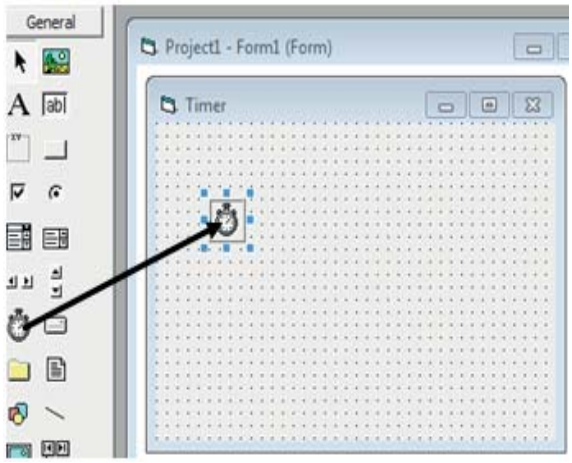


Figure 8.27 Timer

m. File System Controls

File System Controls are set of controls which help us to add file handling capabilities to our program. They are used together to provide an interface for accessing and exploring drives, folders and files. File system controls can be drive list box, directory list box and file list box.

1) **Drive List Box:** The Drive ListBox is for displaying a list of drives available in your computer. When you place this control on the form and run the program, you will be able to select different drives from your computer.

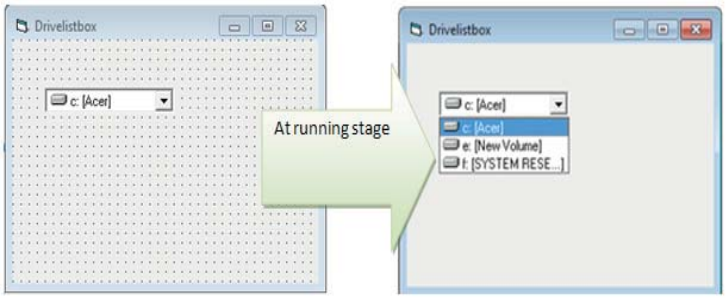


Figure 8.28 Use of DriveList control

2) **Directory List Box:** is for displaying the list of directories or folders in a selected drive. When you place this control into the form and run the program, you will be able to select different directories from a selected drive in your computer.

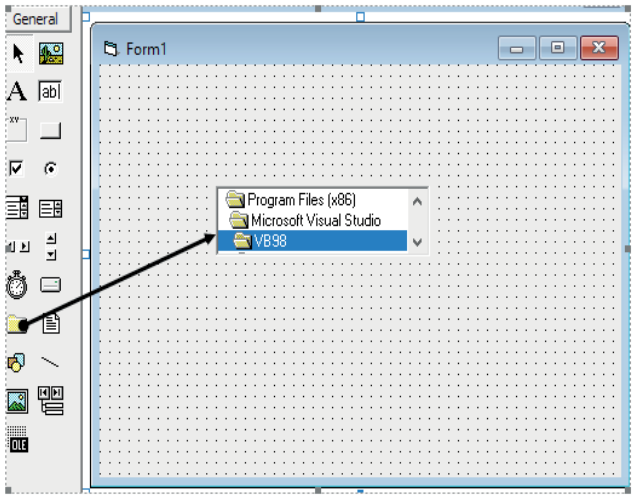


Figure 8.29 Directory List Box

3) File List Box: This control displays a list of files in the current folder.
These are important controls even though there are plenty of controls used in VB.

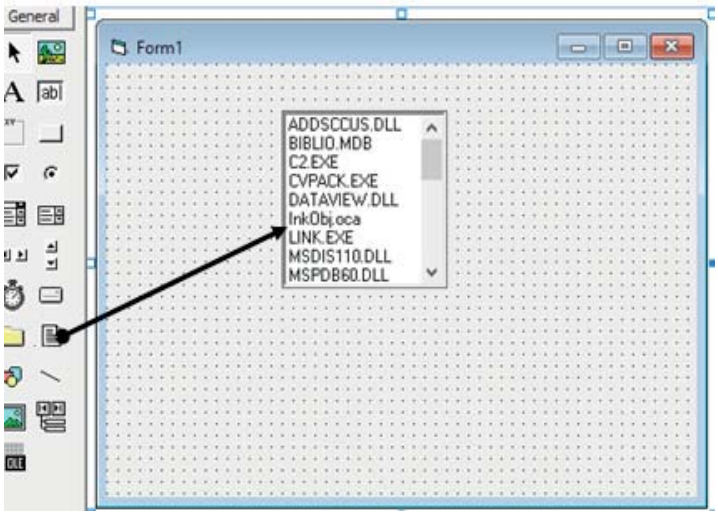


Figure 8.30 File listbox.

g. ADO

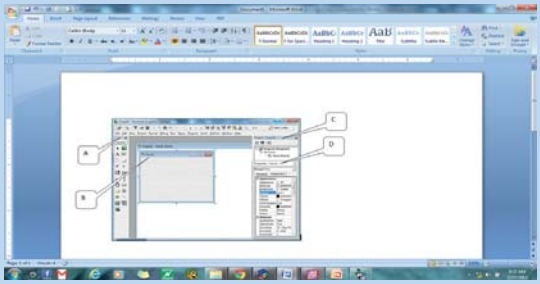
ADO (ActiveX Data Objects)is the preferred method for accessing non-Jet tables. ADO and DAO are interchangeable for most things and you are unlikely to experience real timing differences with the two methods. There are some things that ADO cannot do with an Access database that DAO can do.DAO is native to Jet, and on several operations, it might perform a bit faster on Jet than ADO. DAO will contain some methods and properties that are only relevant to Jet, not other databases.

h. DAO

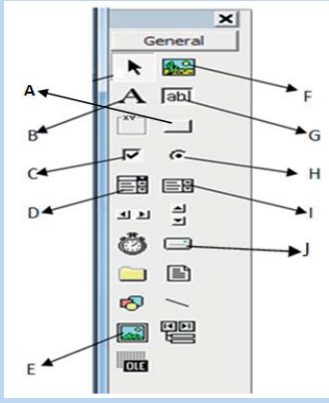
DAO(Data Access Objects) was the first object-oriented interface that exposed the Microsoft Jet database engine (used by Microsoft Access) and allowed Visual Basic developers to directly connect to Access tables as well as other databases - through Open Database Connectivity **(**ODBC).

Application activities 8.3.2

1. Name the following parts of the window labeled A,B,C and D



2. Name and give the function of labeled parts of the following Toolbar..



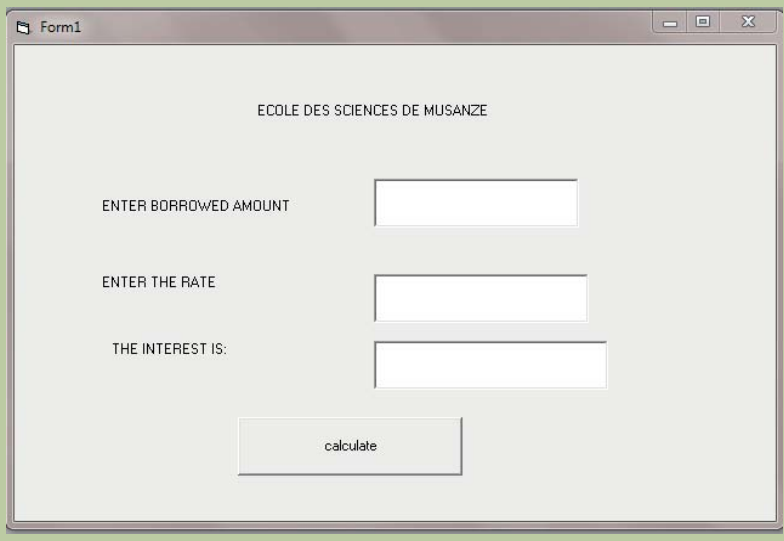
3. Design a form "Rectangle" that contains a round rectangle.
4. What is the most common Event associated with the Command Button?

8.4. Planning and Developing a Visual Basic program.8

8.4.1 The process of Planning and Developing a Visual Basic program

Learning activity 8.4.1

Consider and observe the following form that contains different controls and help a school to calculate the monthly interest according to the borrowed money per month.



1. What is the role of each control on the form.
2. Which cause the command button named **calculate** to find and display the interest
3. What happen if you do not find the simple interest?
4. What can you do if run a program and you are informed that the program contains errors?

Developing a VB program is mainly done in three steps namely setting up user interface, defining the properties, and creating code.

a. Draw the interface

At this step, you will be using the object to design the interface of your application; the controls will be taken from the tool box by dragging it from there to the form designer.

b. Set properties

At this stage, you will be setting up properties for your form and controls. Those properties are set from the properties window.

c. Write the event code.

Coding is to be done in the code window and you get there by double clicking the object you want to code. Now we double click on the form1, the source code Window for the form1 appears:

Private Sub Form_Load()
End Sub
You just have to write your code between two statements. In order to display the output, you have to add Form1.show

Example 8.4.1: VB application to display the message: "**welcome to the world of programming:"**

Private sub Form_load()
Form1.show
Print"welcome to the world of programming"
End sub.

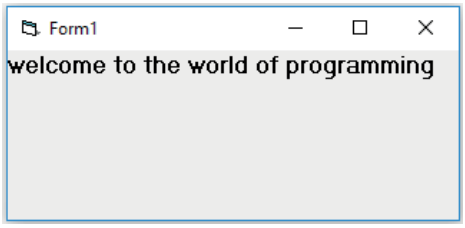


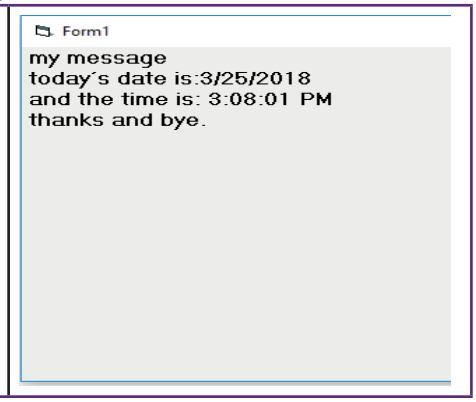
Figure 8.31 running form

Example 8.5.1.: application to display date, time and message

Codes:

```
Private Sub Form_Load()
Form1.Show
Print " my message"
Print " today's date is:" & Date
Print " and the time is: " & Time
Print " thanks and bye."
End Sub
```

Output:



Example 8.4.2: ***Build an application to calculate the volume of the cylinder***

First build the interface as it is shown on the right side

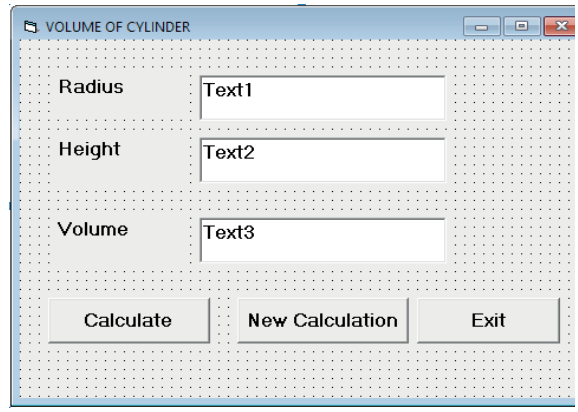


Figure 8.32 designed a form of volume of cylinder

Procedure:

- Put the pointer on the form 1, right button, properties, go to name and replace form1 by frmvolumecylinder.
- Go to **caption**; and replace form1 by **Volume of cylinder**
- Go now to toolbox, then choose label (3 times) and if toolbox is not there, go to view then choose toolbox
- Click once on the label one, right button; **properties** and replace label1 **name** by **lblradius** and the **caption** by the **"Radius"**
- Click on the label2, properties and replace the label2 name by **lblheight** and the **caption** by **"Height"**
- Click on label3, properties and replace the name by **lblarea** and caption by "**Volume"**.
- Then go to toolbox, choose now the textbox (3 times) starting by the first, go to properties and remove.
- Go to text name and replace the text1, text2 and text3 by txtradius, txtheight and txtvolume respectively.
- Go back to toolbox and choose command buttons (3 times)
- Go to properties and replace names command1, command2 and command3 by cmdcalculate, cmdnew and cmdexit respectively captions by Calculate, Newcalculation and Exit.

Command "Calculate"

Double click on command button "Calculate" to write the code to calculate the area when you click on it at running time.

Codes:

```
Private Sub cmdcalculate_Click()
txtvolume.Text = 3.14 * Val(txtradius.Text ^ 2) * Val(txtheight.Text)
End Sub
```

Command "New Calculation"

Double click on the command New calculation to write the code to let you make other calculation when click on it.

Codes:

```
Private Sub cmdnew_Click()
txtradius.Text = ""
txtheight.Text = ""
txtvolume.Text = ""
End Sub
```

Command "Exit"

And finally Exit command to write the code to end the program when you click on it.

Codes:

```
Private Sub cmdexit_Click()
Unload Me
End Sub
```

Output:

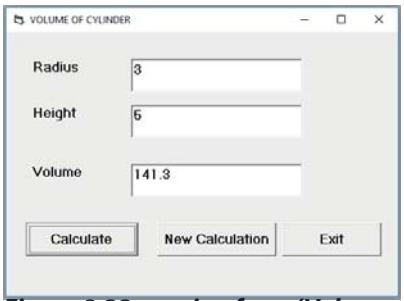


Figure 8.33 running form (Volume of Cylinder)

Application activity 8.4.1

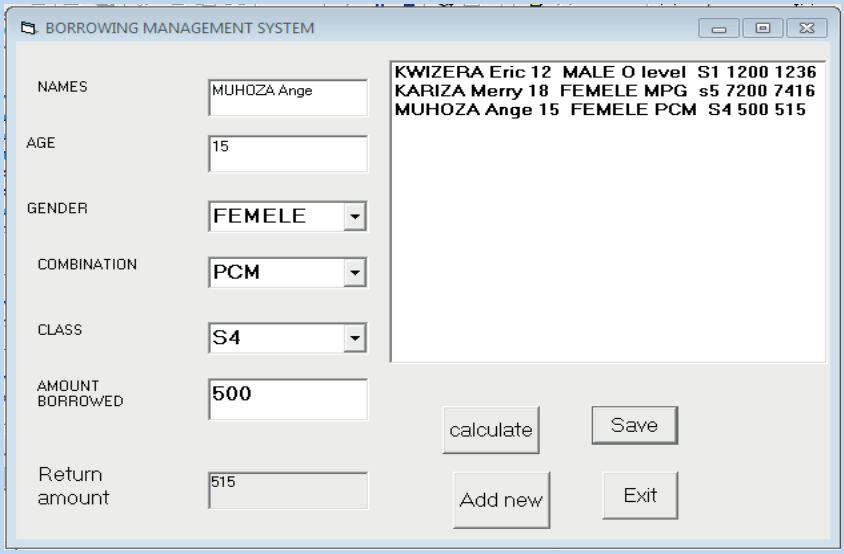


Figure 8.331 Form receiviig inputs and processing them

1. A. Design a form like above using all possible control and its proportional properties.
2. Coding a program.

8.4.2. Debugging Your Code and Handling Errors

Application activities 8.4.2

UMWARI is a student in senior five, once planning and developing a visual basic program but in running stage it does not run correctly.

a. Is it possible to correct it? Explain.
b. Give appropriate term of correct the challenges faces.
c. Identify why program does not runand the possible solutions.

8.4.2 Debugging your Code and Handling Errors

A. Program error: is a result of bad code in some program involved in producing the erroneous result

B. Handling errors

Error handling should be used to process only exceptional situations, despite the fact that there is nothing to prevent that programmer from using errors as an alternate form of program control.

These errors can be grouped into three categories:

- Syntax errors
- Run-time errors
- Logic errors

- **Syntax errors** are grammatical errors in the formulation of statements and are picked up by the interpreter while you are typing in the code (providing the syntax checking option - under environment options - is set to `yes`).
- **Run-time errors** these are errors that cannot be detected until the program is running. The syntax of the statements is correct, but once executed they cause a error situation to arise. Examples of run-time errors are attempted division by zero or trying to access a non-existent object.
- **Logic errors** these are errors that cause the program to behave incorrectly. They generally arise through failure on the part of the programmer to arrive at a correct algorithm for the task. Typical problems might be incorrect ordering of statements, failure to initialise or re-initialise a variable, assignment to an incorrect variable, use of '<' instead of '<=', use of '`and`' instead of '`or`', or omission of a crucial step in the processing. Logic errors may hide in a program even when it appears to work - they may only surface under certain conditions. This is why careful testing is so important.

c. Debugging

Debugging is a process by which you find and resolve errors in your code. To debug code in Visual Basic, consider the ideas suggested below. These techniques can also be applied in different sequences.

- Print the code, if you find it easier to read code on paper instead of softcopy.
- Run the application to find trouble spots:

From the Run **menu**, choose '**`Start`**' to begin running the application. Run until an error stops execution, or halt execution manually when you suspect an error by choosing '`Break`' from the Run menu. Resolve all compile errors and run-time errors. From the Run **menu**, choose '**`Continue`**' to continue running the application.

Once bugs are found try out bug fixes and then make edits by testing individual lines of new or debugged code in the Debug window. Search and replace code

for all occurrences of an error, checking other procedures, forms, or modules with related code. From the `Run` menu, choose '`Restart`' to reset application variables and properties and restart the application from the beginning.

Debugging tools.

These are tools designed to help in stopping the execution of a program at specific points, detecting run-time and logic errors and understanding the behavior of error-free code.

8.4.4 Building an executable file

You can make an executable file (.exe) from Visual Basic using the following procedure.

To make an executable file in Visual Basic

1. From the **File** menu, choose **Make** *projectname* **.exe** where *projectname* is the application name for the project.

2. Type a file name, or browse through the directories and select an existing file name to overwrite an existing executable with a newer version.

3. By clicking the **Options** button, you can also specify a number of version-specific details about the executable file in the **Project Properties** dialog box.

4. If you want to modify the version number of the project, set the appropriate **Major**, **Minor**, and **Revision** numbers. Selecting **Auto Increment** will automatically step the **Revision** number each time you run the **Make** *projectname* **.exe** command for this project.

5. To specify a new name for the application, under **Application**, type a new name in the **Title** box. If you want to specify a new icon, choose one from the list.

6. You can also enter version-specific commentary on a variety of issues under the **Version Information** box (comments, company name, trademark and copyright information, and so on) by selecting a topic from the list box and entering information in the text box.

7. Choose **OK** to close the **Project Properties** dialog box, and then choose **OK** in the **Make** *appname* **.exe** dialog box to compile and link the executable file.

You can run the executable file like any other Windows-based application: double-click the icon for the executable file.

8.4.5. Deploying a VB Project

The Visual Basic Package and Deployment Wizard makes it easy for you to create the necessary cab files and setup programs for your application. Like other wizards, the Package and Deployment Wizard prompts you for information so that it can create any exact configuration.

There are three ways of starting the Package and Deployment Wizard:

- Run it from within Visual Basic as an add-in. If run the wizard as an add-in, first set the necessary references in the Add-In Manager to load the wizard. When the wizard as an add-in, Visual Basic work with the project currently open. If work with another project, either open that project before starting the add-in, or use the wizard as a stand-alone component.
- As a stand-alone component from outside the development environment. Run the wizards as a stand-alone component, prompted choose the project on which working with.
- Start it in silent mode by launching it from a command prompt. "Running the Wizard in Silent Mode" in this topic for more information.

After you start the wizard, a series of screens prompt you for information about your project and let you choose options for the package. Each screen explains how it is to be used, including which information is optional, and what information must be entered before move to the next screen. If it display information on screen, press F1 or click the Help button.

Note: Save and compile project before running the Package and Deployment Wizard.

In most cases, the Package and Deployment Wizard create a package that is ready for deployment. However, if customize packaging process further or provide functionality not supported by the Package and Deployment Wizard, modify the Setup Toolkit Project.

To start the Package and Deployment Wizard from within Visual Basic

1. Open the project you want to package or deploy using the wizard.

   Note If you are working in a project group or have multiple projects loaded, make sure that the project you want to package or deploy is the current project before starting the wizard.

2. Use the Add-In Manager to load the Package and Deployment Wizard, if necessary: Select **Add-In Manager** from the **Add-Ins** menu, select **Package and Deployment Wizard** from the list, then click **OK**.

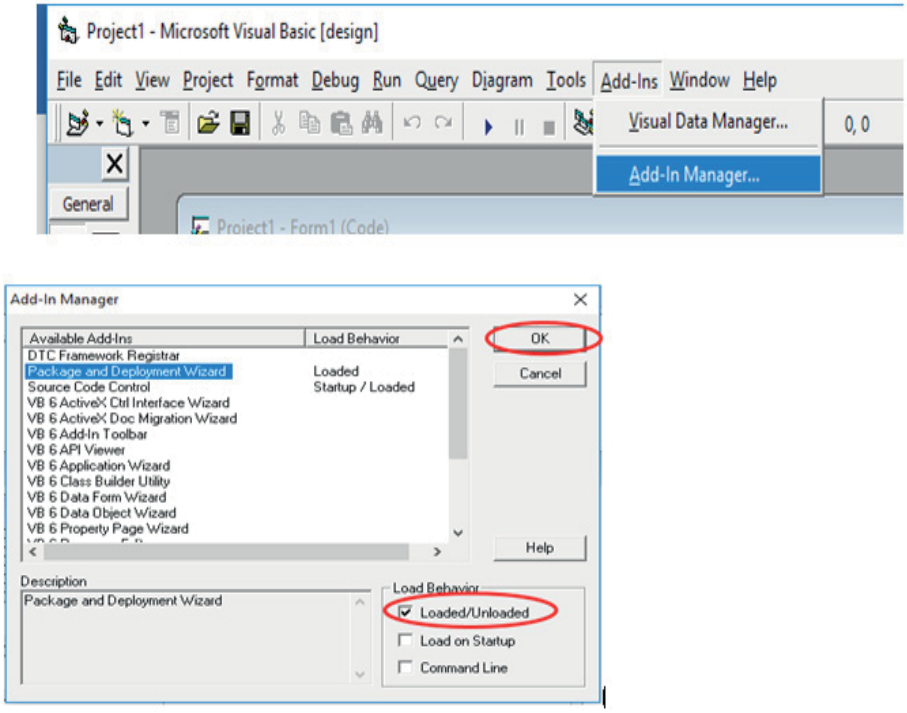


Figure 8.34 Deployment Wizard process.

3. Select **Package and Deployment Wizard** from the **Add-Ins** menu to launch the wizard.

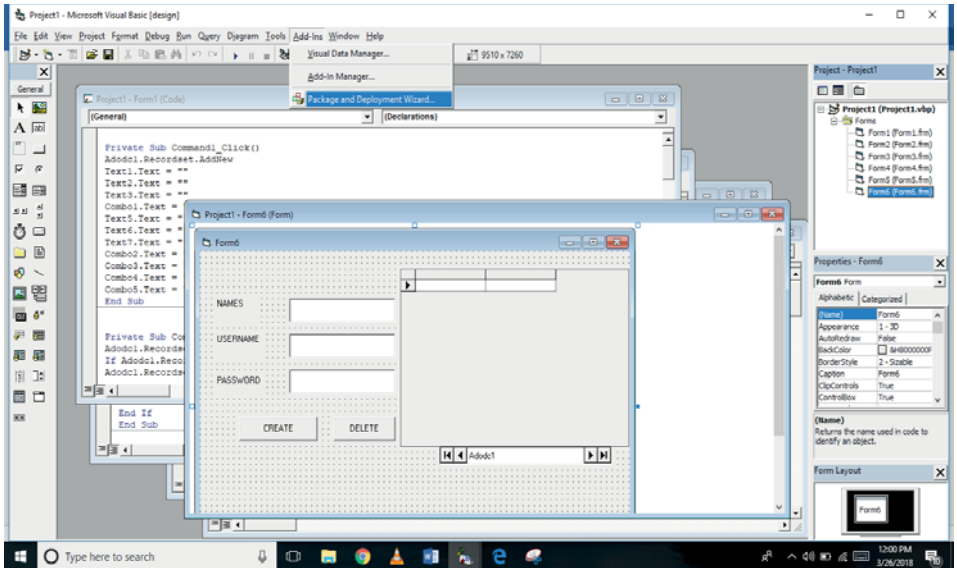


Figure 8.35 package and Deployment Wizard process.

4. On the main screen, select one of the following options:

    - If you want to create a standard package, Internet package, or dependency file for the project, click **Package**.
    - If you want to deploy the project, click **Deploy**.
    - If you want to view, edit, or delete scripts, click **Manage Scripts**.

    For an introduction to these options, see "The Package and Deployment Wizard."

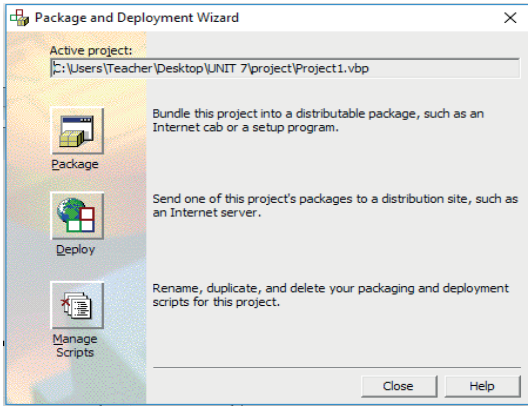


Figure 8.36 Package and Deployment Wizard.

5. Proceed through the wizard screens.

To start the Package and Deployment Wizard as a stand-alone component

1. If the project you want to package is open, save it and close Visual Basic.
2. Click the **Start** button, and then click **Package and Deployment Wizard** from the Visual Basic submenu.
3. In the **Project** list on the initial screen, choose the project you want to package.

    Note You can click **Browse** if your project is not in the list.
4. On the main screen, select one of the following options:

    - If you want to create a standard package, Internet package, or dependency file for the project, click **Package**.
    - If you want to deploy the project, click **Deploy**.
    - If you want to view, edit, or delete scripts, click **Manage Scripts**.
5. Proceed through the wizard screens.

Running the Wizard in Silent Mode

Using scripts, you may package and deploy your project files in *silent mode*. In silent mode, the wizard runs without your having to attend it to make choices and move through screens. The wizard packages and deploys your project using the settings contained in a script.

Silent mode is especially useful if you are packaging and deploying as part of a batch process. For example, early in the development of your project, you may use the Package and Deployment Wizard to package your project and deploy it to a test location. You can later create a batch file to perform the same packaging and deployment steps periodically as you update your project.

To package and deploy in silent mode

1. Open MS-DOS prompt.
2. Type the name of the wizard executable, pdcmdln.exe, followed by the path and file name of Visual Basic project, and the appropriate command line arguments.
3. PDCmdLn.exe C:\Project1\Project1.vbp /p "Internet Package"
4. /d Deployment1 /l "C:\Project1\Silent Mode.log"

   Note: Can perform packaging and deployment in a single silent session by specifying both the /p and the /d arguments, as shown in the example above. Otherwise, use either /p or /d.

Argument	Description
/p *packagingscript*	Type **/p** followed by the name of a previously saved packaging script to package the project silently according to the specified script.
/d *deploymentscript*	Type **/d** followed by the name of a previously saved deployment script to deploy the project silently according to the specified script.
/l *path*	Specifies that the wizard should store all output from the wizard, such as error messages and success reports, to a file rather than displaying them on the screen. Type **/l** followed by the path and file name of a file in which output should be stored. If the file does not exist, the wizard creates it. **Tip** If you do not choose to log output using this argument, the wizard will display a dialog box to notify when packaging or deployment is finished. In order to see this dialog box, you may minimize or close other windows.

/e path	Specifies a path for the project's executable file, if it is different from the path for the project. This argument allows you to package in silent mode when you are working in a multi-developer environment. You might use this option if development and packaging occur on different computers.

Table 8.1 packaging and deployment options

Note Any file or script name that includes spaces should be enclosed in quotation marks, as shown in the example above.

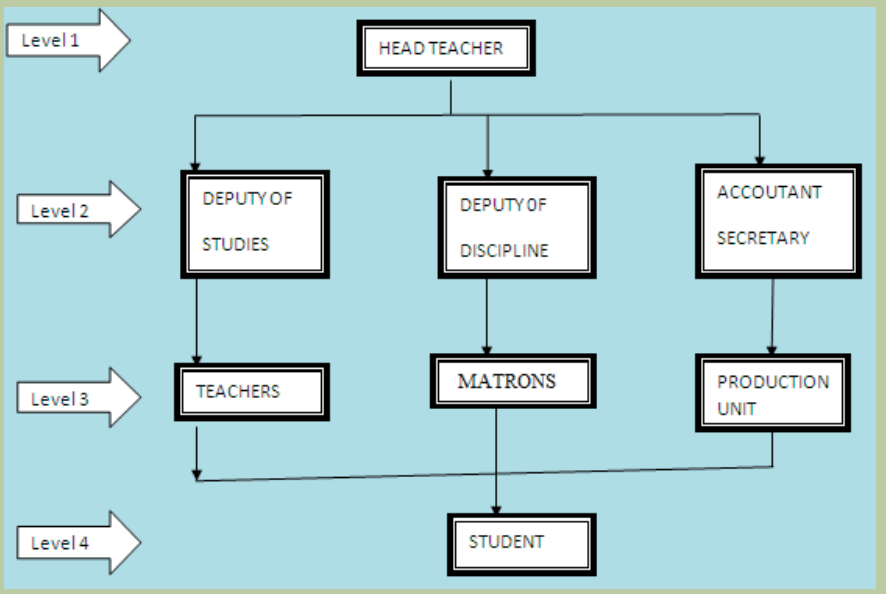
Application activities 8.4.2

1. Describe three types of errors made in Visual basic.
2. Define the term debugging.
3. Design a simple calculator which makes addition, division, multiplication and subtraction of two numbers.
4. Hirwa Eric is a programmer, when he attempts to run his VB code, it generates an error.
   a. Explain why this program generate an error?
   b. Give the strategies Eric must follow in order to solve this problem.

8.5 WORKING WITH MENUS AND DIALOG BOXES

Learning activities 8.5

Observe the following organization structure of a girls' school and answer the following questions:



1. As a student of S5 how many levels you pass through in order to reach to head teacher?
2. Why is it necessary to pass at each level?
3. Design the above organizational structure into a VB form window use of menus.

Windows applications provide groups of related commands in Menus. These commands depends on the application, but some-such as Open and Save are frequently found in applications. Menus are intrinsic controls. Visual Basic provides an easy way to create menus with the modal Menu Editor dialog. The below dialog is displayed when the Menu Editor is selected in the Tool Menu. The Menu Editor command is grayed unless the form is visible. And also you can display the Menu Editor window by right clicking on the Form and selecting Menu Editor.

8.5.1 Multiple Document Interface (MDI).

The Multiple Document Interface (MDI) is another class of the form that allows you to open Windows within a parent container window to simplify the exchange of information among documents, all under the same roof.

Applications such as Microsoft office Word and Microsoft office excel are the best examples of MDI, where many documents can be opened simultaneously within the main document.

To use windows applications that can open multiple documents at the same time and allow the user to switch among them with a mouse-click.

Each document is displayed in its own window and all document windows have the same behavior.

The main Form, or MDI form, isn't duplicated, but it acts as a container for all the windows, and it is called the **parent Window**.

The windows in which the individual documents are displayed are called **Child windows.**

An MDI application must have at least two Form, the parent form and one or more child forms.

Each of those forms has certain properties. These can be many child forms contained within the parent form, but there can be only one parent form.

The parent form may not contain any controls, while the parent form is open in design mode, the icons on the toolbox are not displayed, but you can't place any controls on the form.

To create an MDI application, at least two forms are needed in the application: One is the parent or container form and the second is the child form or the form contained within the parent.

A single child is required for the simple MDI Projects.

Procedures:

- Select New project from the file menu
- In the case you already have a form ,then in the properties window set the name property to frmchild and its caption to MDI_Child
- Right-click on the forms folder in the project window
- Click on add from the MDI form to create the MDI Parent form.
- Select MDI form from the Add MDI form dialog
- In the Properties window set the name property to frmMDI and the caption property to MDI_Parent.
- On the project menu, select project1.properties, set the startup object list to

frmMDI. In the Case you omit this; the application will start with the child form.

Select the frmchild from the project Explorer

- **Set the form's MDI child property to true in the properties window.** This will cause this form, which is the child, to rest inside the MDI Parent container.
- Save project as from the file menu and save the form as **MDI.frm** and project as **MDI.vbp**

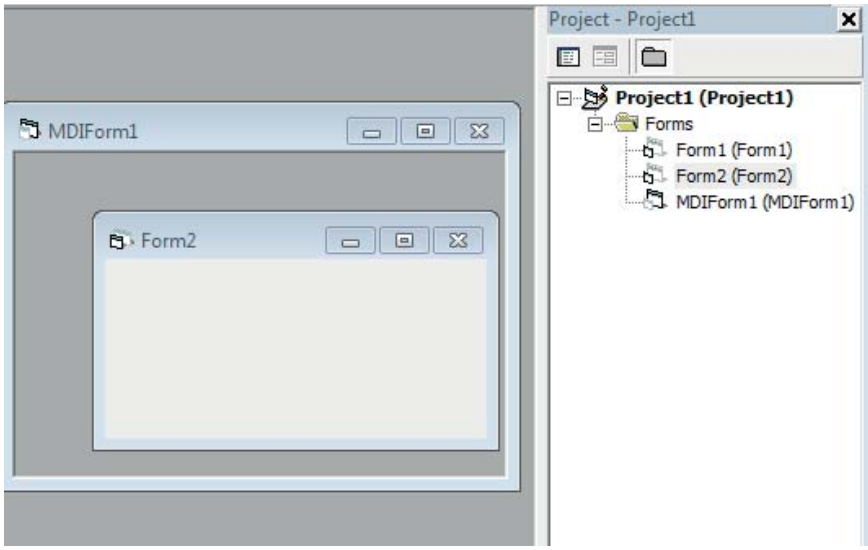


Figure 8.37 MDI child form

8.5.2 Creating Menus

Menus, which are located on the **menu bar** of a form, contain a list of related commands. When you click a menu title in a Windows-based program, a list of menu commands should always appear in a well-organized list.

Most menu commands run immediately after they are clicked. For example, when the user clicks the **Edit** menu **Copy** command, Windows immediately copies information to the Clipboard. However, if ellipsis points (…) follow the menu command, Visual Basic displays a dialog box that requests more information before the command is carried out.

This section includes the following topics:

- Using the Menu Editor
- Adding Access and Shortcut Keys
- Processing Menu Choices

8.5.3 Using The Menu Editor

The Menu Editor is a Visual Basic dialog box that manages menus in your programs. With the Menu Editor, you can:

- Add new menus
- Modify and reorder existing menus
- Delete old menus
- Add special effects to your menus, such as access keys, check marks, and keyboard shortcuts.

See the Figure below for Menu editor Window:

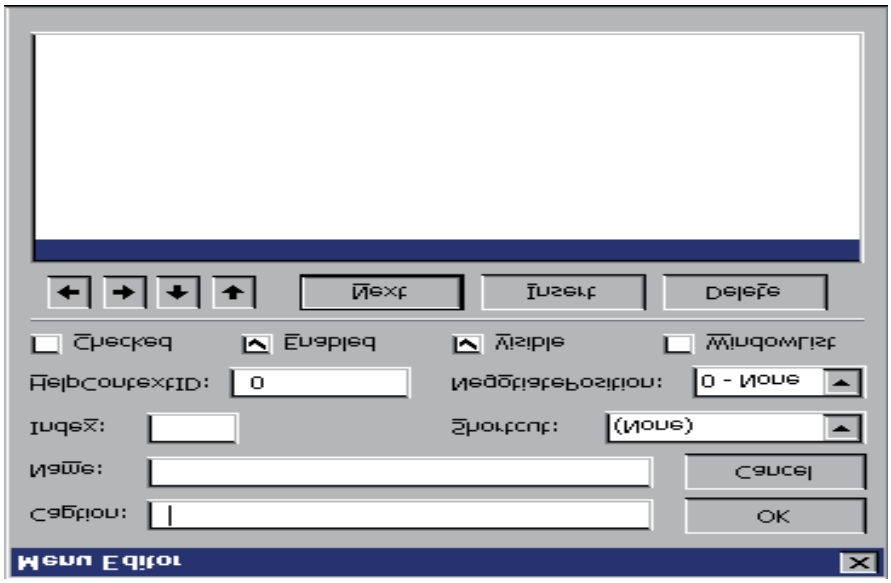


Figure 8.38 The Menu Editor Window.

Creating Menu Command Lists

To build lists of menu commands, you first need to create the menus and then add them to the program menu bar.

To create a list of menu commands on a form

4. Click the form itself (not an object on the form).
5. On the Visual Basic toolbar, click the Menu Editor icon, or select **Menu Editor** from the **Tools** menu.
6. In the **Caption** text box, type the menu caption (the name that will appear on the menu bar), and then press TAB.

7. In the **Name** text box, type the menu name (the name the menu has in the program code). By convention, programmers use the *mnu* object name prefix to identify both menus and menu commands.
8. To add the menu to your program menu bar, click **next**. The Menu Editor clears the dialog box for the next menu item. As you build your menus, the structure of the menus and commands appear at the bottom of the dialog box.
9. In the **Caption** text box, type the caption of your first menu command.
10. Press tab, and then type the object name of the command in the **Name** text box.
11. With this first command highlighted in the menu list box, click the right arrow button in the Menu Editor. In the **Menu** list box, the command moves one indent (four spaces) to the right. Click the right arrow button in the Menu Editor Dialog box to move items to the right, and click the left arrow button to move items to the left.
12. Click **Next**, and then continue to add commands to your menu.

The position of list box items determines what they are:

List box item	Position
Menu title	Flush left
Menu command	One indent
Submenu title	Two indents
Submenu command	Three indents

To add more menus

1. When you're ready to add another menu, click the left arrow button to make the menu flush left in the **Menu** list box.

2. To add another menu and menu commands, repeat Steps 3 through 9 in the preceding procedure.

3. When you're finished entering menus and commands, click **OK** to close the Menu Editor. (Don't accidentally click **Cancel** or all your menu work will be lost.) The Menu Editor closes, and your form appears in the programming environment with the menus you created.

Adding Event Procedures After you add menus to your form, you can use event procedures to process the menu commands. Clicking a menu command on the form in the programming environment displays the event procedure that runs when the menu command is chosen. You'll learn how to create event procedures that process menu selections in Processing Menu Choices.

8.5.4 Adding Access and Shortcut Keys.

Visual Basic makes it easy to provide access key and shortcut key support for menus and menu commands.

Access and Shortcut Keys

The access key for a command is the letter the user can press to execute the command when the menu is open. The shortcut key is the key combination the user can press to run the command without opening the menu. Here's a quick look at how to add access and shortcut keys to existing menu items:

- Add an access key to a menu item Start the Menu Editor. Prefix the access key letter in the menu item caption with an ampersand (&).
- Add a shortcut key to a menu command and Start the Menu Editor. Highlight the command in the menu list box. Pick a key combination from the Shortcut drop-down list box.

Creating Access and Shortcut Keys

You can create access keys and shortcut keys either when you first create your menu commands or at a later time.

The following illustration shows the menu commands associated with two menus, **File** and **Clock**. Each menu item has an access key ampersand character, and the **Time** and **Date** commands are assigned shortcut keys. See figure below.

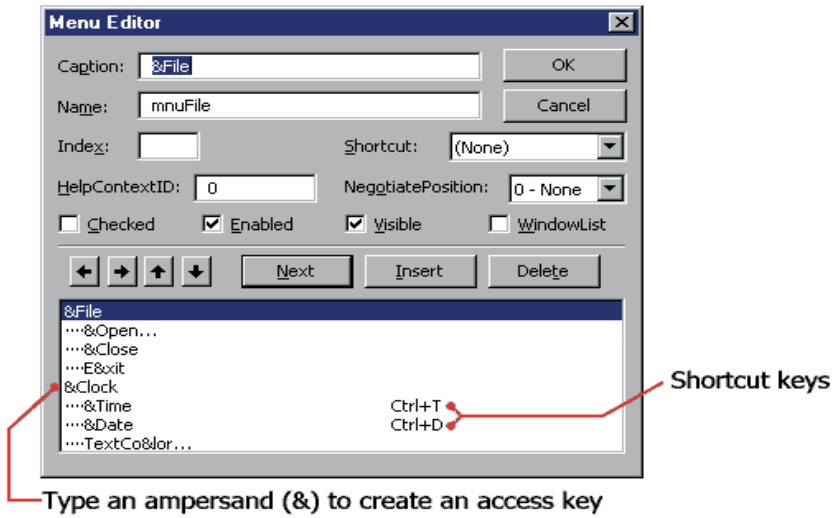


Figure 8.39 Menu Editor Window showing how to create shortcut keys

8.5.5 Processing Menu Choices

After you place menu items on the menu bar, they become objects in the program. To make the menu objects do meaningful work, you need to write event procedures for them. Typically, menu event procedures:

- Contain program statements that display or process information on a form.
- Modify one or more object properties.

For example, the event procedure for a command named **Time** might use the **Time** keyword to display the current system time in a text box.

Processing the selected command might require additional information (you might need to open a file on disk, for example). If so, you can display a dialog box to receive user input by using a common dialog box. You'll learn this technique in the next section.

Disabling a Menu Command

In a typical Windows application, not all menu commands are available at the same time. In a typical **Edit** menu, for example, the **Paste** command is available only when there is data on the Clipboard. When a command is disabled, it appears in dimmed (gray) type on the menu bar. You can disable a menu item by:

- Clearing the **Enabled** check box for that menu item in the Menu Editor.
- Using program code to set the item's Enable property to False. (When you're ready to use the menu command again, set its Enable property to True.).

Application activity 8.5

Design a program which have five menus (Home,Insert,Page Layout, review and View) and sub_menu in home (File, open and Edit) and the open File option display form1.

You should be able to open the above form using shortcut(Ctrl+F).

UNIT 9

VARIABLES, OPERATORS, EXPRESSIONS AND CONTROL STRUCTURES.

UNIT 9: VARIABLES, OPERATORS, EXPRESSIONS AND CONTROL STRUCTURES.

Key Unit Competency

To be able to use variables, operators, expressions and control structures in a Visual Basic program.

INTRODUCTORY ACTIVITY

The Visual Basic program below asks the user to enter the names, year of birth and the sex and displays the following messages: "Good Morning Sir, you are .... years old" if the sex is Male and "Good morning Madam, you are ...years old" Analyze the following VB source code and answer corresponding questions:

```
Private Sub Command1_Click()
Dim names, sex As String
Dim year As Integer
names = Text1.Text
sex = Text2.Text
year = Val(Text3.Text)
m = 2018 - year
If sex = "Male" Then
Print "Good morning Sir , you are"; m; " years old "
ElseIf sex = "Female" Then
Print "Good morning Madam ,you are"; m; " years old "
Else
Print sex; "  sex does not exist!"
End If
End Sub
```

1. List the variables used in the source code above
2. What is the type of Visual Basic Data used in this source code?
3. Describe operators used in the source code
4. What is the control structure used in the source code?
5. Draw the corresponding interface and give an output of the above source code

9.1. Types of Visual Basic Data

ACTIVITY 9.1

In every program, a programmer declares variables with their data types.

1. Explain why the declaration of variables is important during the writing of programs
2. What are the data types used in programs?
3. Discuss the storage size and range of values of data types used in programs.

There exist many types of variables, but the most used in Visual Basic are:

- An elementary data type, such as Boolean, Long, or Decimal
- A composite data type, such as an array or structure
- An object type, or class, defined either in current application or in another application

All the above types are grouped in two major data types: numeric data type and non-numeric data type.

9.1.1 Numeric data types

Numeric data types are types of data that consist of numbers, which can be computed mathematically with various standard operators such as add, minus, multiply, divide among others.

Table 9.1: Numeric data types

Type	Storage	Range of values	Prefix
Byte	1 byte	0 to 255 or 0 to 2^8-1	byt
Integer	2 bytes	-32,768 to 32,767 or -1/2(2^{16}) to ½(2^{16})-1 (signed)	Int
Long	4 bytes	-2,147,483,648 to 2,147,483,648 or -1/2(2^{32})to ½(2^{32})-1 (signed)	Lng
Single	4 bytes	-3.402823E+38 to -1.401298E-45 for negative values 1.401298E-45 to 3.402823E+38 for positive values	sng
Double	8 bytes	-1.79769313486232E+308 to -4.94065645841247E-324 for negative values 4.94065645841247E-324 to 1.79769313486232E+308 for positive values.	Dbl
Currency	8 bytes	-922,337,203,685,477.5808 to 922,337,203,685,477.5807 or -1/2(2^{64})to ½(2^{64})-1 (signed)	cur

Decimal	12 bytes	+/- 79,228,162,514,264,337,593,543,950,335 or +/-2^{96} if no decimal is use +/- 7.9228162514264337593543950335 (28 decimal places).	dec
Variant (numeric)	16 bytes	Any value as large as Double	Var

9.1.2 Non-numeric data types

Non- numeric data types are types of data that cannot be manipulated mathematically using standard arithmetic operators. The non- numeric data types comprise of text or string data types, the Date data types, the Boolean data types, Object data type and Variant data type.

Table 9.2: Non-numeric data types

Data type	Storage	Range	Prefix
String(fixed length)	Length of string	1 to 65,400 characters	str
String (variable length)	Length +10 bytes	0 to 2 billion characters	
Date	8 bytes	January 1,100 to December 31,9999	dte
Boolean	2 bytes	True or False	bln
Object	4 bytes	Any embedded object	Obj
Variant(text)	Length + 22 bytes	Same as variable-length string	

Suffixes for literal

Literals are values that you assign to data. In some cases, we need to add a suffix behind a literal so that VB can handle the calculation more accurately. For example, we can use num=1.3089# for a Double.

In some cases, a literal can be forced to a particular data type; for example, when assigning a particularly large integral literal value to a variable of type decimal, the following example produces an error:

```
Dim myDecimal as Decimal
myDecimal = 10000000000000000000  ' This causes a compiler error.
```

The error results from the representation of the literal. The Decimal data type can hold a value this large, but the literal is implicitly represented as a Long, which cannot.

To make the previous example work, you can append the D type character to the literal, which causes it to be represented as a Decimal:

Dim `MyDecimal` AsDecimal = 1000000000000000000D

So, a suffix needs to be added behind a literal so that VB can handle the calculation more accurately. If a suffix is not added behind a literal, it generates a syntax error. For example, num = 2.3807**#** is used for **Double** data type**.**

The Table 9.3 below shows Visual Basic Data Types and their suffixes or Appended type character

Table 9.3: Visual Basic Data Types and their suffixes

Data type	Suffix or Appended type character
Boolean	None
Integer	%
Long (Integer)	&
Single (Floating)	!
Double(Floating)	#
Currency	@
Date	None
Object	None
String	$
Variant	None

Note: String literals are normally enclosed within double quotations while **date** and **time** literals are enclosed within two **#** sign. Strings can contain any characters, including numbers. The following table summarizes the data types with their enclosing characters and appended type characters

Table 9.4: data types with their enclosing characters and appended type characters

Data type	**Enclosing character**	**Appended type character**
Boolean	(none)	(none)
Byte	(none)	(none)
Char	“	C
Date	#	(none)
Decimal	(none)	D or @
Double	(none)	R or #
Integer	(none)	I or %
Long	(none)	L or &
Short	(none)	S

Single	(none)	F or !
String	“	(none)

Examples:

- Strname = “ Hakizimana Petero”
- Telnumber = “ 250783297650”
- Firstday = #01-Jan-06 #
- ArrivalTime = # 12:00 am#

APPLICATION ACTIVITY 9.1

1. What would happen if a programmer uses one of VB keywords (reserved words) as a variable identifier?
2. In a program, if a user declares an integer variable and enters a string like “book”, explain the nature of the results produced by such a program
3. Study the following VB source code, draw its corresponding interface and then give an output.

```
Private Sub Form_Load()
Form1.Show
Name1 = “ UWIHANGANYE”
phone = “ +250783297650”
Jour = #7/29/1996#
BithTime = #7:20:00 PM#
Text1.Text = Name1
Text2.Text = phone
Text3.Text = Jour
Text4.Text = BithTime
End Sub
```

9.2 Variables

ACTIVITY 9.2

1. Outline different rules of naming a variable in Visual Basic
2. Explain general format of variable declaration
3. Explain general format of variable initialization
4. Draw interface and write a VB program which asks a user for entering a radius of a circle to display the area of a circle after running the program as shown in figure 9.1. Consider pi as the constant whose numeric value 22/7.

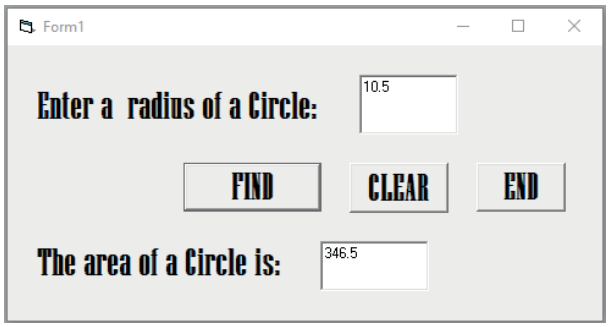


Figure 9.1 Output of a program displaying area of a given circle

A variable is a name assigned to a data storage location. It can also be defined as area allocated by the computer memory to hold data. Variables are used in Visual Basic to store various kinds of data during program execution.

9.2.1 Variable name

In Visual Basic Program, each variable must be given a name. To name a variable in Visual Basic, there is a set of rules to be followed.

The rules used in naming variables in Visual Basic are:

- No more than 40 characters
- Variables must include letters, numbers and underscores (**)**. No punctuation, spaces or other characters are permitted.
- A variable name must begin with a letter
- The name cannot be a **reserved word (words needed by Visual Basic).** For example, End is a reserved word in **Visual Basic** and therefore should not be used as a variable name.

Examples of valid and invalid variable names are displayed in the table below:

Table 9.5: examples of valid and invalid variable names

Valid name	Invalid name	Reason
My_house	My.house	The dot character is used to separate name of a file and its extension. So it is a reserved character
Today	2Years	The first character must be a letter not a number
First_Name_Last_name	Her&hermother	& is not acceptable

9.2.2 Variable declaration

They are three ways for declaration of a variable.

1. Default Declaration

If the variables are not implicitly or explicitly declared, they are assigned the **variant (**a variable that can hold data of any size or format**)** type by **default**. The variant data type is a special type used by visual basic that contains numeric, string or date data.

For example: the following VB program finds the sum of two given numbers:

```
Private Sub Command1_Click()
Dim n1, n2, sum As Variant
n1 = Val(Text1.Text)
n2 = Val(Text2.Text)
n3 = n1 + n2
Text3.Text = n3
End Sub
```

2. Implicit Declaration

To **implicitly** declare a variable, use the corresponding suffix. For example:

- **TextValue$= my school** creates a string variable
- **Amount%=1000** creates an integer variable**.**

3. Explicit Declaration

In a good programming practice, variables have to be explicitly declared. In this variable declaration way, Visual Basic will take care of ensuring consistency in upper and lower case letters used in variable names.

To **explicitly** declare a variable, its scope has to be firstly determined. There are four levels of scope are **procedural level, static, form and module level, and last global level**

These levels of scope progress from the narrowest (block) to the widest (namespace), where *narrowest scope* means the smallest set of code that can refer to the element without qualification.

a) Procedure level

With procedure level, variable are declared using the Dim statement

Example:

```
Dim MyName As String
Dim MyInt As Integer
```

This is given to variables declared using the **Dim reserved word** within a **block** of statemet

such as an **if … Then … Else statement**, or a loop statement.

For example:

```
Private Sub Command1_Click()
Dim fname As String  'procedure level variable
fname = Text1.Text
If fname = "Paul" Then
Dim Newname As String ' block level variable this can't be accessed out of if...then else
Newname = "Jean Paul"
Text2.Text = Newname
Else
Newname = fname
Text2.Text = Newname
End If
End Sub
```

Here, the variable can only exist and be used within the **If … Then … Else** statement. It is a local variable visible only inside the block where it has been declared.

b) Procedure level scope, static

In procedure level ,static, variables declared in this manner do not retain their value once a procedure terminate, within a procedure level, static variables are declared using the **Dim** statement

For example:

```
Private Sub End_click()
Static age As Integer
Static MyName As string
End Sub
```

Note that

- Dim can only exist and be used within the procedure or function. It is a local variable in that procedure or function.
- Procedure level variables declared in this manner do not retain their value once a procedure terminates.
- To make a procedure level variable retain its value upon existing the procedure, replace the **Dim** keyword with **Static**

c) Form(module) level scope

The variable is declared in the **General Declarations** section at the start of the module. Form (module) level variables retain their values and are available to all procedures within

that form. Module level variables are declared in the declarations part of the general object in the form's (module's) code window. The Dim keyword is used.

Example1: Dim MyDate as Date

Example2: a program which requires three numbers entered from keyboard then calculates their sum and average.

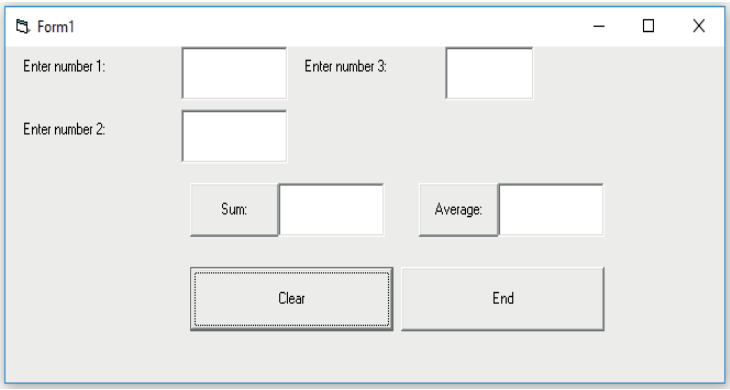


Figure 9.2: user interface for the application

Requirements to perform the asked question;

Design interface containing five Text box with their corresponding labels and four commands representing sum, agerage clear and exit respectively least but not last assign properties to each control and last attach code

VB Source code can look as follows :

```
Dim sum As Integer  'module level variable
Private Sub add_Click()
Dim num1, num2, num3 As Integer 'procedure level variable
num1 = Val(Text1.Text)
num2 = Val(Text2.Text)
num3 = Val(Text3.Text)
sum = num1 + num2 + num3
Text4.Text = sum
End Sub
Private Sub average_Click()
Dim average As Integer 'procedure level variable
average = sum / 3
Text5.Text = average
End Sub
Private Sub Command3_Click()
Text1.Text = ""
```

Text2.Text = “”
Text3.Text = “”
Text4.Text = “”
Text5.Text = “”
End Sub
Private Sub Command4_Click()
Unload Me
End Sub

Output for the asked question:

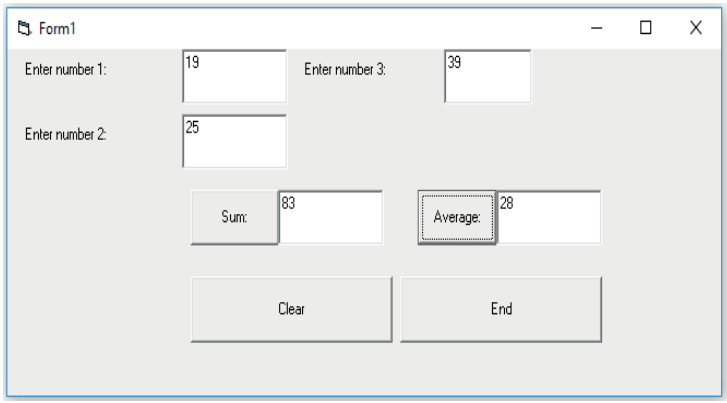


Figure 9.3: running the application

d) Global level scope

This is given to variables declared using the **public** reserved word within a code module. The variable is declared in the General Declarations section at the start of the module. Global level variables retain their value and are available to all procedures within an application. It is advisable to keep all global variables in one module. Use the **Global or public** keyword.

Examples: Global MyInt as Integer or

Public MyInt as Integer

Notice that several variables can be declared in one statement without having to repeat the data type. In the following statements, the variables i, j, and k are declared as type Integer, l and m as Long, and x and y as Single:

Dim i, j, k As Integer
‘All three variables in the preceding statement are declared as Integer.
Dim l, m As Long, x, y As Single
‘In the preceding statement, l and m are Long, x and y are Single.

Variable initialization

Once a variable is declared, it does not have a defined value, hence it cannot be used until it is **initialized** by assigning it a value.

Thus, variable initialization is the process of providing or assigning value to the variable

Syntax: Variablename = value

Where **Variablename** is the descriptive name of the variable, **=** is the assignment operator and **value** is the value that a variable will contain.

For examples:

```
Private Sub Command1_Click()
Dim Number1As Integer
Dim FirstName As string
DimNumber1As Single
Number1 = 5
FirstName = "Steve"
LoanAmount = 67.38
Print Number1
Print FirstName
Print LoanAmount
End Sub
```

9.2.4 Declaring Constants

Constant is value in memory that does not change during program execution. For example, in mathematics, **pi** is a constant whose numeric value is 22/7.

The declaration of a constant goes immediately with its initialization.

Syntax: Const constant name As data type = value

Examples:

```
Private Sub Form_Load()
Form1.Show
Dim circumference As Single
Const r As Integer = 8
Const Pi As Single = 3.14159265358979
circumference = 2 * r * Pi
Print "The circumference of a circle is:"; circumference
End Sub
```

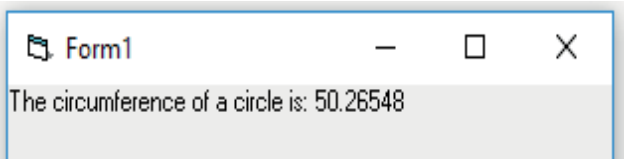


Figure 9.4: displaying program output on the form

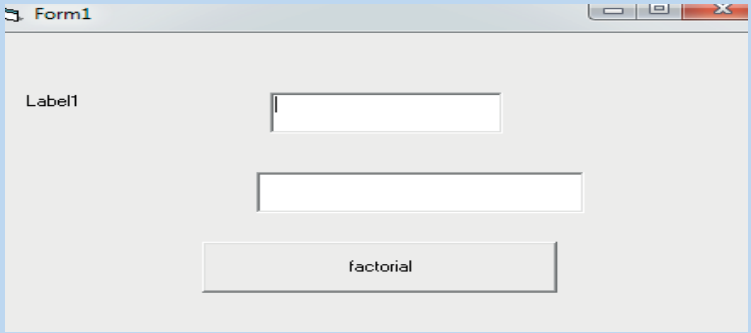
9.3 Scope of a variable

Learning Activity 9.1

As any other programming language, Visual Basic application acts on data stored in variables. Using internet and library textbooks, discuss different scopes of variables.

APPLICATION ACTIVITY 9.2

1. Study the given VB source code of a program that requires a positive integer and displays its factorial. To avoid possible memory overflow, replace the "fact" variable data type with an appropriate data type whose size can hold a large value.




```
Private Sub Command1_Click()
 Dim fact As Integer, n As Byte
 Do While n > 1
 fact = fact * n
 n = n - 1
Loop
 Text2.Text = fact
End Sub
```

2. Write a VB program that asks to enter a radius of a sphere and display its volume. Declare Pi as a constant variable and assign to it 22/7 value.
3. Write a VB program that requests the user to enter 2 numbers and displays their sum, difference and product.

The scope of a declared element is the set of all code that can refer to it without qualifying its name. It is an area of influence and lifetime for a variable. This means a period of existence dependent on where and how the variable is declared. An element can have scope at one of the following levels:

Table 9.6: Scope of variables

Level	Description
Block/private scope	Available only within the code block in which it is declared
Procedure scope	Available to all code within the procedure in which it is declared
Module /global scope	Available to all code within the module, class, or structure in which it is declared

In Microsoft Visual Basic for Applications, the three scopes available for variables are private, module, and public.

9.3.1 Private (Local) Scope

A local variable with private scope is recognized only within the procedure in which it is declared. A local variable can be declared with a **Dim or Static** statement.

- When a local variable is declared with the **Dim** statement, the variable remains in existence only as long as the procedure in which it is declared is running. Usually, when the procedure is finished running, the values of the procedure's local variables are not preserved, and the memory allocated to those variables is released.
- A local variable declared with the **Static** statement remains in existant the entire time Visual Basic is running. Local variable retains its value after the module has finished executing.

9.3.2 Module (Global) Scope

A variable that is recognized among all of the procedures on a module sheet is called a "module-level" variable. A module-level variable is available to all of the procedures in that module, but it is not available to procedures in other modules. A module-level variable remains in existence while Visual Basic is running until the module in which it is declared is edited. Module-level variables can be declared with a Dim or Private statement at the top of the module above the first procedure definition.

Example:

```
Dim A As Integer      ' Module-level variable.
Private B As Integer  ' Module-level variable.

Sub Example1()
  A = 100
  B = A + 1
End Sub

Sub Example2()
  MsgBox "The value of A is " & A
  MsgBox "The value of B is " & B
End Sub

Sub Example3()
   Dim C As Integer   ' Local variable.
   C = A + B
   MsgBox "The value of C is " & C
End Sub

Sub Example4()
   MsgBox A  ' The message box displays the value of A.
   MsgBox B  ' The message box displays the value of B.
   MsgBox C  ' The message box displays nothing because C was a local variable.
End Sub
```

9.3.3 Public scope

Public variables have the broadest scope of all variables. A public variable, like a module-level variable, is declared at the top of the module, above the first procedure definition. A public variable cannot be declared within a procedure. A public variable is always declared with a "Public" statement. A public variable may be declared in any module sheet.

Example:

```
Public SalesPrice As Integer
   Public UnitsSold As Integer
   Public CostPerUnit As Integer
   Sub CDSales()
     Dim X as String
     SalesPrice = 12
     UnitsSold = 1000
     CostPerUnit = 5
     X = "yes"
```

APPLICATION ACTIVITY 9.3

Write a Visual Basic program which computes the sum and average of the three given integers using keyboard.To achieve this, follow the instructions below:

i. The program uses two different procedures addition () and moyenne ()
ii. Declare the 3 variables as local variables in addition () procedure
iii. Declare sum as global variable to hold the sum calculated by addition() procedure
iv. The moyenne () procedure calculates the average of three given integers and display it.

9.4 Operators and expressions in Visual Basic

ACTIVITY 9.4

1. Enumerate the various operators used in C++ programming language
2. What is the value of the following expression if x=10 and y=5?
3. Evaluate the following expression if A=10, B=5, C=0 and D=22(A AND B)OR(C OR D) AND A

In programming context, an **operator** is a symbol or a keyword that instructs a compiler to evaluate mathematical or logical expressions.A Visual Basic expression is a combination of **operators** and **operands**.

For example, in the expression: energy = mass * light_speed ^2

- Energy, mass ,light_speed and 2 are the operands while
- = is the assignment operator, * is multiplication operator and ^ the exponentiation arithmetic operator.

9.4.1 Arithmetic operators

The most common operators in Visual Basic are the arithmetic operators. Table 9.4 below gives a summary of arithmetic operators supported in Visual Basic.

Table 9.7: Arithmetic operators

Operator	Description	Example
^	Exponentiation	6^ 2=36
*	Multiplication	8*10=80
/	Division	18/3=6

\	Integer division(**truncates**)	10\3=3(Only the integer portion of the result is considered)
Mod	Modulus	15Mod4=3
+	Addition	9 + 4 = 13
-	Subtraction	5-3=2

Note that:

- Parentheses () can change the precedence
- To concatenate two strings, use the & symbol or the + symbol:
  txtSample.Text=”My country“+“is”+ “Rwanda”
  myname=”John” & “Mugabo”

9.4.2 Relational (Comparison) operators

Normally, they are used to compare data values and then the result helps to decide what action to take. The result of comparison operation is a Boolean value (**True or False**).

Table 9.8: Relational (Comparison) operators

Operator	Meaning	Example
==	Equal to	txtItem == “ Video”
>	Greater than	txtValue > 20.45
<	Less than	txtValue < 50
>=	Greater than or equal to	txtValue >= 60.5
<=	Less than or equal to	txtValue <= 20
<>	Not equal to	txtPrice <> 10

Logical operators

Table 9.9: Logical operators

Operator	**Operation**
Not	Negation
And / AndAlso	Logical and
Or/ OrElse	Logical or
Xor	Exclusive Or

AND: An expression has a true value if both operands are true, and false value elsewhere

OR: An expression has **a true** value if at least one of the operand is **true** and false elsewhere.

Xor: An expression has a **true** value if both operand are different and has a **false** value if

both operands are the same.

NOT: Negates Boolean operand

Table 9.10: Example of logical operators

Operand	Logical Not expression	
X	Logical Not operator	Output of expression
True	Not X	False
False	Not X	True

Table 9.11: Implementation of Logical Not operators

Operands		Logical expressions					
X	Y	**Logical And**		**Logical Or**		**Logical Xor**	
True	True	X And Y	True	X Or Y	True	X Xor Y	False
True	False	X And Y	False	X Or Y	True	X Xor Y	True
False	True	X And Y	False	X Or Y	True	X Xor Y	True
False	False	X And Y	False	X Or Y	False	X Xor Y	False

The following example of VB program illustrates the Not ,And,Or,and Xor operators

```
Private Sub Form_Load()
Form1.Show
Dim a, b, c, d, e, f, g, x, y As Boolean
x = Not 23 > 14
y = Not 23 > 67      ' The preceding statements set x to False and y to True
a = 23 > 14 And 11 > 8
b = 14 > 23 And 11 > 8 ' The preceding statements set a to True and b to False.
c = 23 > 14 Or 8 > 11
d = 23 > 67 Or 8 > 11 ' The preceding statements set c to True and d to False.
e = 23 > 67 Xor 11 > 8
f = 23 > 14 Xor 11 > 8
g = 14 > 23 Xor 8 > 11 ' The preceding statements set e to True, f to False, and g to False.
Print x
Print y
Print a
Print b
Print c
Print d
Print e
Print f
Print g
End Sub
```

The output of the above source code is:

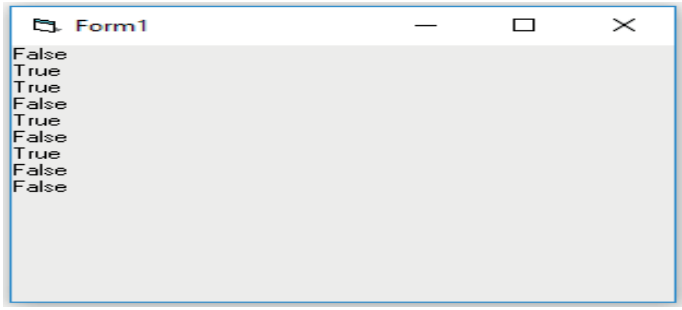


Figure 9.5: Program output

Note that The **AndAlso Operator** is very similar to the **And** operator, in that it also performs logical conjunction on two **Boolean** expressions. The key difference between the two is that **AndAlso** exhibits short-circuiting behavior. If the first expression in an **AndAlso** expression evaluates to **False**, then the second expression is not evaluated because it cannot alter the final result, thus **AndAlso** returns **False.**

Similarly, the OrElse Operator performs short-circuiting logical disjunction on two Boolean expressions. If the first expression in an OrElse expression evaluates to, then the second expression is not evaluated because it cannot alter the final result, and OrElse returns True.

9.4.3 Bitwise operators

Bitwise operations evaluate two integral values in binary (base 2) form. They compare the bits at corresponding positions and then assign values based on the comparison. Bitwise operators are similar to logical operators only that they are specifically used to manipulate binary digits.

Table 9.12: Bitwise operators

Operator	Description
Not	Bitwise Not
And	Bitwise And
Or	Bitwise Or
Xor	Bitwise exclusive Or (Xor)

Example:

Consider the expression below:

x = 5 Or 6

5 in binary form = 101

6 in binary form = 110

101 And 110 = 100

The **bitwise And** operator compares the binary representations, one binary position (bit) at a time. If both bits at a given position are 1, then a 1 is placed in that position in the result. If either bit is 0, then a 0 is placed in that position in the result. The result is treated as decimal. The value 100 is the binary representation of 4, so x = 4.

The **bitwise Or** operator takes a 1 and assigns to the result bit if either or both of the compared bits is 1.

Example:

Consider the expression below:

x = 5 Or 6

101 (5 in binary form)

110 (6 in binary form)

111 (The result, in binary form)

The result is treated as decimal. The value 111 is the binary representation of 7, so x =7.

- **Bitwise Not** takes a single operand and inverts all the bits, including the sign bit, and assigns that value to the result. This means that for signed positive numbers, **Not** always returns a negative value, and for negative numbers**, Not** always returns a positive or zero value.

Example:

Dim x As Integer

x = Not 5

101 (5 in binary form) , its negation is 010

The result is treated as decimal. The value 010 is the binary representation of 2, so x =2.

- **Bitwise Xor** assigns a 1 to the result bit if exactly one of the compared bits (not both) is 1.

Example:

Dim x As Integer

x = 5 Xor 6

101 (5 in binary form)

110 (6 in binary form)

011 (The result, in binary form)

The result is treated as decimal. The value 011 is the binary representation of 3, so x =3.

- The **AndAlso** and **OrElse** operators do not support bitwise operations.

The following example of VB program illustrates the use of bitwise operators

```
Private Sub Form_Load()
Form1.Show
Dim x, y, z, w, v, s As Integer
x = 5
y = 7
s = Not x
z = x And y
w = x Or y
v = x Xor y
Print " The value of s is:"; s
Print " The value of z is:"; z
Print " The value of w is:"; w
Print " The value of v is:"; v
End Sub
```

Output is as follow:

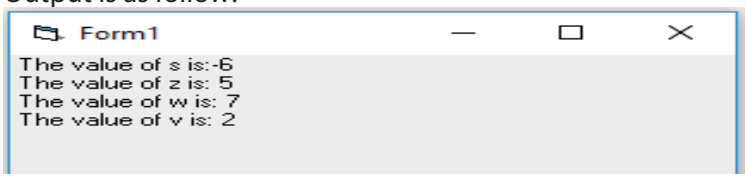


Figure 9.6: Program output

9.4.4 Precedence of operators

When several operations occur in an expression, each part is evaluated and resolved in a predetermined order called *operator precedence*.

Precedence Rules:

When expressions contain operators from more than one category, they are evaluated according to the following rules:

- The arithmetic and concatenation operators have greater precedence than the comparison, logical, and bitwise operators.
- All comparison operators have equal precedence, and all have greater precedence than the logical and bitwise operators, but lower precedence than the arithmetic and concatenation operators.
- The logical and bitwise operators have lower precedence than the arithmetic, concatenation, and comparison operators.
- Operators with equal precedence are evaluated left to right in the order in

which they appear in the expression

Just like in BODMAS, the order of precedence can be changed by use of parenthesis.

Table 9.13: Operator precedence

Operator	Description	Precedence	
^	Exponentiation	Left to right	Highest
*	Multiplication	Left to right	
/	Division	Left to right	
Mod	Modulus	Left to right	
+	Addition	Left to right	
-	Substraction	Left to right	
&	String concatenation	Left to right	
=	Equal to	Left to right	
<>	Not equal to	Left to right	
<	Less than	Left to right	
>	Greater than	Left to right	
<=	Less than or equal to	Left to right	Lowest
>=	Greater than	Left to right	
Not	Logical not	Left to right	
And,AndAlso	Logical and	Left to right	
Or,OrElse	Logical and	Left to right	
Not	Bitwise Not	Left to right	
And	Bitwise And	Left to right	
Or	Bitwise Or	Left to right	
Xor	Bitwise Xor	Left to right	
=	Assignment	Right to left	

Note that:

- The string concatenation operator (&) is not an arithmetic operator, but in precedence it is grouped with the arithmetic operators.
- When operators of equal precedence appear together in an expression, for example multiplication and division, the compiler evaluates each operation as it encounters it from left to right. The following example illustrates this.

```
Private Sub Form_Load()
Form1.Show
Dim x, y, z As Integer
x = 96 / 8 / 4 - 8 * 4 / 2
```

```
y = (96 / 8) / (4 - 8) * 4 / 2
z = 96 / (8 / 4) - 8 * 4 / 2
Print " The value of x is:"; x
Print " The value of y is:"; y
Print " The value of z is:"; z
End Sub
```

Output:

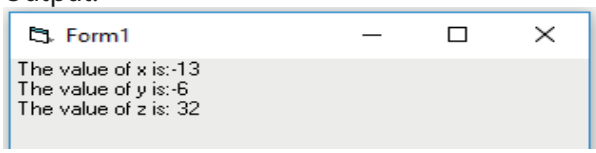


Figure 9.7: Program output

APPLICATION ACTIVITY 9.4

1. Study the following program and determine the output after its execution

```
Private Sub Form_Load()
 Form1.Show
Dim x ,y,z ,q,p,k As Integer
Dim i, j,a As Boolean
    x = 42
     y=7
     z = 24
    i = (x<=35) And (z = 24)
    j = (x =35) Or (y < 10)
    a = Not (x>30) Xor (y = 7)
    q = x And y
     p = x Or z
     k = x Or z
Print i
Print  j
Print a
Print q
Print p
Print k
     End sub
```

2. Perform bitwise And, Or and Xor on the following variables
   a. S = 25, T = 31
   b. X = 50, Y = 18

9.5. Decision structures in Visual Basic

Learning Activity 9.5

1. Discuss the use of decision control structures (If…Then, If… Then… Else, Nested If … Then…. Else …Statements) in Visual Basic programming
2. Using Visual Basic, write a program that prompts a user to enter a student's score in Mathematics. If the score is above 45%, the program should display "Pass" otherwise it should display "Fail"

To implement an application in visual basic, you need several coding structures to control the program flow. The control structures are used to control the flow of execution of a program they are used in a program to repeat a set of operations (looping control structures), and for making decisions according to the conditions when a program is executing (decision control structures). A decision structure will test a condition and then perform operations depending on whether yes or not the condition is met (True or false).

9.5.1 Types of decision control structures

Decision control structures also known as selection control statements are conditional logic used when there is one or more options or alternatives to chose from. There are four types of decision control structures namely **if…then, if…then else, nested if ….then else, and select case**

A. If ….. Then statement

If .... Then statement is used for making a decision in program.

The If...Then selection structure performs an indicated action only when the condition is True; otherwise the action is skipped

The syntax of **If ….Then** structure is :

If *condition* **Then**
VB statement(s)
End If

Example: To set the Maths mark to grade "A" we can use the following statement:

```
If txtMaths >= 80 Then
   textGrade = " A"
End if
```

B. If …..Then ….Else statement

The If...Then...Else selection structure allows the programmer to specify that different actions are to be performed when the condition is True than when the condition is False. If the condition is true, the first part of the if …then is executed. However if the condition is false the block of the else statements is executed

The syntax of the If …Then …Else conditional statement is as follows:

If *(Boolean expression/condition)* **Then**

*VB statement(s)***Else**

VB statement(s)

End If

Example: a program to check whether the entered number is odd or even

```
Dim n As Integer
a = Val(Text1.Text)
If n Mod 2 = 0 Then
Text2.Text = "The number you entered is even"
Else
Text2.Text = "The number you entered is  odd "
End If
End Sub
```

We can also use the same example by using MsgBox function

```
        x = val(Text1.Text)
If x Mod (2) = 0 Then
    MsgBox "This is an even number you entered"
Else
        MsgBox "You entered the odd number"
End If
```

C. Nested If ….Else statement

Nested If...Then...Else statement test for multiple cases by placing If...Then...Else selection structures inside If...Then...Else structures

The syntax of the Nested If …Else

```
If (Boolean expression/condition) Then
  VB statement(s)
```

Else if (Boolean expression/condition)
VB statement(s)
Else
VB statement(s)
End If

In using branching statements, be aware that each IF and Else If in a block is tested sequentially. The first time an If test is met, the code associated with that condition is executed and the If block is exited. If a later condition is also True, it will never be considered.

Example:

Write a program to read marks (per cent) of university students and put these students into classes according to their marks.

Classes with their corresponding marks:

- From 80 to 100: First class honours
- From 70 to 79: Upper second class honours
- From 50 to 69: Lower second class honours
- From 40 to 49: Pass
- Below 40: Fail

Solution:

User interface

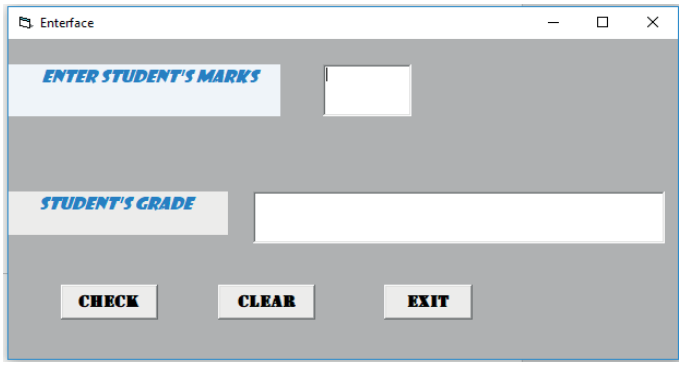


Figure 9.8:User interface

Source code

Private Sub check_Click()
Dim txtmarks As Integer
txtmarks = Val(txtmark.Text)
If txtmarks >= 80 And txtmarks <= 100 Then
textClass = "The student got First Class Honours"

```
ElseIf (txtmarks >= 70 And txtmarks < 80) Then
textClass = "The student got Upper second class Honours"
ElseIf (txtmarks >= 50 And txtmarks < 70) Then
textClass = "The student got Lower second class Honours"
ElseIf (txtmarks >= 40 And txtmarks < 50) Then
textClass = "The student got Pass"
ElseIf (txtmarks >= 40 And txtmarks < 50) Then
textClass = "The student got Pass"
ElseIf (txtmarks >= 0 And txtmarks < 40) Then
textClass = "The student got Fail"
Else
textClass = "Invalid input"
End If
End Sub
Private Sub clear_Click()
txtmark = ""
textClass = ""
End Sub
Private Sub exit_Click()
Unload Me
End Sub
```

Output:

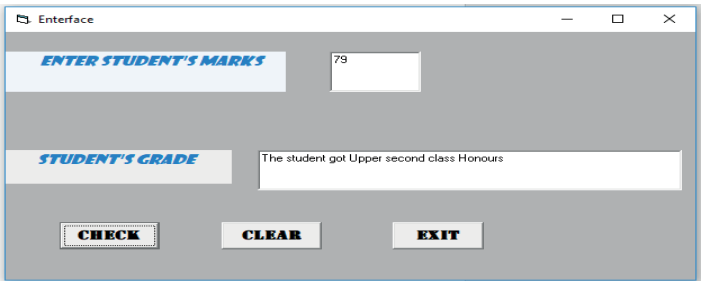


Figure 9.9: Running program

APPLICATION ACTIVITY 9.5

1. Create a simple program which will ask a user to input a value.
   - If the value is 1 it should display the text "The day is Monday"
   - If the value is 2 it should display the text "The day is Tuesday"
   - Do this for all the days of the week.
   - Use the Nested If…Then …Else structure to achieve the above.

2. Write a VB program that would enable the user to enter student marks in three subjects. The program should calculate mean marks and determine whether the student has passed if the pass mark is 50%
3. Observe the following interfaces and write a visual basic program using various mathematical functions like sqrt, sqr, sin, cos, tan.\

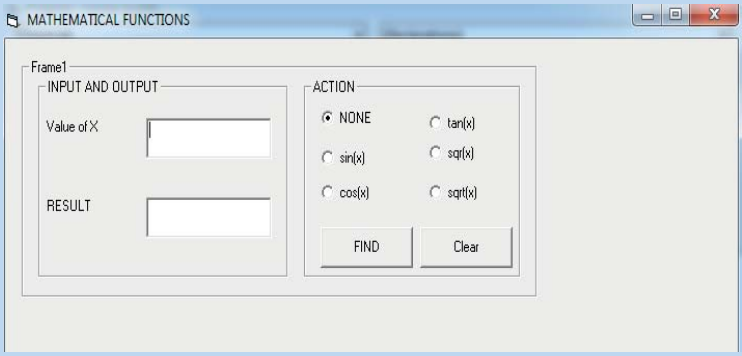


Figure 9.10: user interface for the application

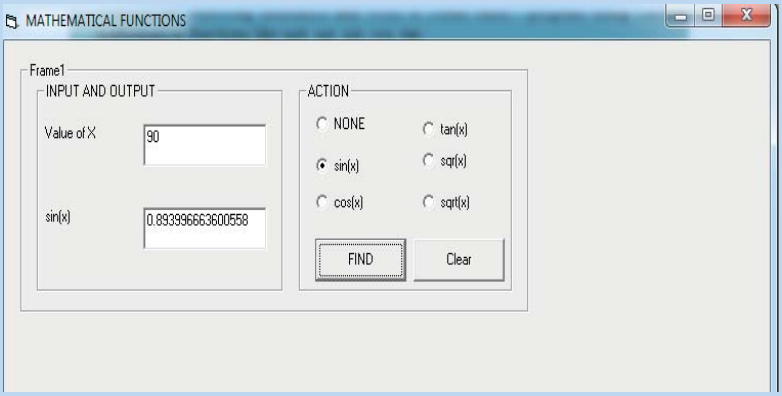


Figure 9.11: Running program

Hint: Once the user enters any value in the first Textbox and selects any option, the output will be displayed in second TextBox, if the user enters a negative number and selects the sqrt option he/she will be informed that the square root of a negative number does not exist and once he or she does not select any option he/she will be informed to select any of available option.

D. Select case statement

Learning Activity 9.6

1. Discuss the use of Select case statement as decision control in Visual Basic programming
2. Create a simple program which will ask a user to input a value.
   - If the value is 1 it should display the text "The month is January"
   - If the value is 2 it should display the text "The month is February"
   - If the value is 3 it should display the text "The month is March"
   - If the value is 4 it should display the text "The month is April"
   - Do the same for all months of a year
   - Use the Select case statement to achieve the above

Select...Case structure is an alternative to nested If...Then...Else for selectively executing a single block of statements from among multiple block of statements. If the "**case**" matches the condition is selected and all instructions within the matching case are executed. Select... case is more convenient to use than the If...Else...End If.

The general format is:

Select case expression
Case value 1
Block of visual basic statements
Case value2
Block of visual basic statements
Case value3
Block of visual basic statements
Case value4
Case Else
Block of visual basic statements
End Select

Each possible value of the item is checked using a Case statement followed by one or more Visual Basic statements that will be carried out if there is a match for that value. The final part of the structure is usually a Case Else statement to be executed if none of the above values match. This is useful for informing the user that he /she is out of cases.

Example 1: A Visual Basic program that displays the Grade according to the marks entered by students:

Marks	Grade
0 - 10	U
11 - 20	S
21 - 30	F
31 - 40	E
41 - 50	D
51 - 60	C
61 - 74	B
75 - 100	A

Interface:

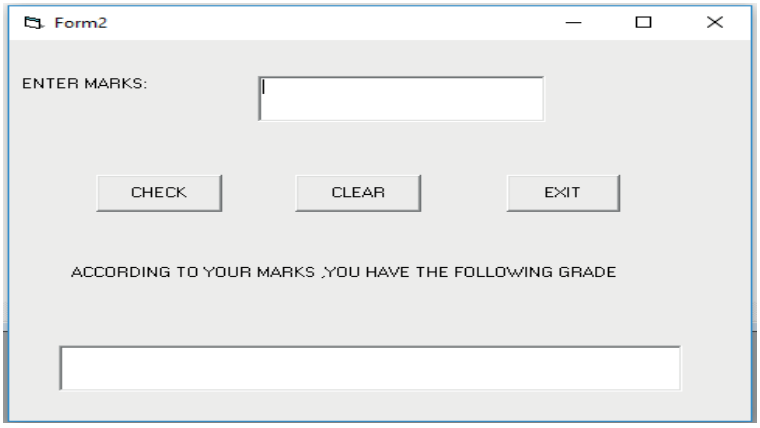


Figure 9.12: User Interface

Source codes:

```
Private Sub Command1_Click()
Dim x As Integer
x = Val(Text1.Text)
Select Case x
Case 0 To 10
Text2.Text = "Grade U"
Case 11 To 20
Text2.Text = "Grade S"
Case 21 To 30
Text2.Text = "Grade F"
Case 31 To 40
Text2.Text = "Grade E"
```

Case 41 To 50
Text2.Text = "Grade D"
Case 51 To 60
Text2.Text = "Grade C"
Case 61 To 70
Text2.Text = "Grade B"
Case 71 To 100
Text2.Text = "Grade A"
Case Else
Text2.Text = "No Grade because you Entered Invalid marks"
End Select
End Sub
Private Sub Command2_Click ()
Text1.Text = ""
Text2.Text = ""
End Sub
Private Sub Command3_Click ()
Unload Me
End Sub

Output:

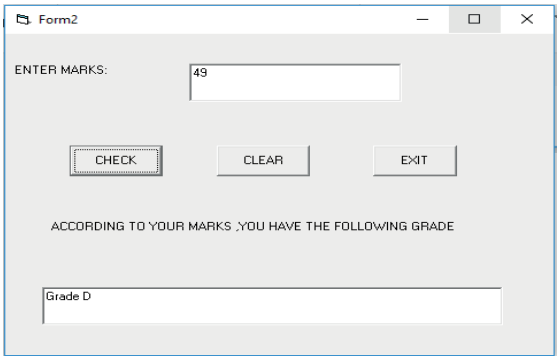


Figure 9.13: Running the program

Example 2: Application to assign grade (comment) to students depending upon the marks they've got.

Dim x As Single

Private Sub Compute_Click()
x = marks.Text
Select Case x
Case 0 To 49
txtgrade.Text = "Need to work hard"

```
        Case 50 To 59
        txtgrade.Text = "Average"
        Case 60 To 69
        txtgrade.Text = "Above average"
        Case 70 To 79
        txtgrade.Text = "Good"
        Case Else
        txtgrade.Text = "Excellence"
        End Select
    End Sub
```

Or one can use

User interface

VB source code

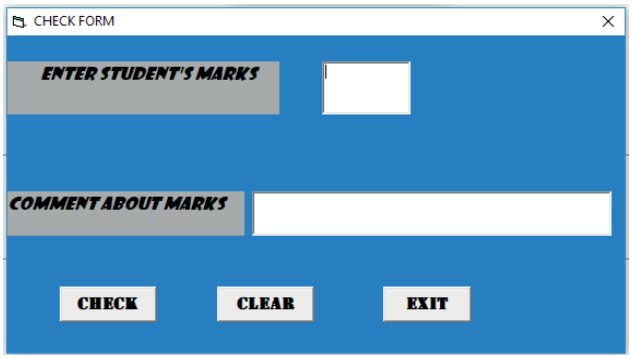


Figure 9.14: User Interface

```
Dim x As Single
    Private Sub check_Click()
    x = mark.Text
    Select Case x
    Case Is >= 100
    textGrade.Text = "Wrong input!"
    Case Is >= 80
    textGrade.Text = "Excellence Student.Keep it Up!"
    Case Is >= 70
    textGrade.Text = " Very Good!"
    Case Is >= 60
    textGrade.Text = "Above average!"
    Case Is >= 50
    textGrade.Text = "Average student!"
    Case Is >= 0
```

```
        textGrade.Text = "You need to work harder!"
        Case Else
        textGrade.Text = "Wrong input!"
        End Select
        End Sub
Private Sub clear_Click()
mark = ""
textGrade = ""
        End Sub
Private Sub exit_Click()
Unload Me
End Sub
```

Out put:

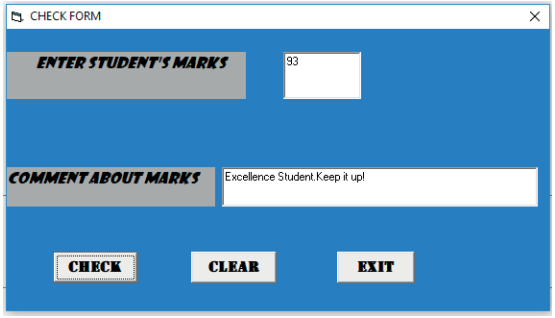


Figure 9.15: Program output

APPLICATION ACTIVITY 9.6

1. Write a VB program that displays the comment according to the entered student's grade

Grade	Comment
A	Excellent
B	Good
C	Fair
D	Poor
E	Fail

2. Write a VB program that allows a user to enter two floating point number from the keyboard and it requests a user to enter his/her preference in order to perform one operation from (addition, subtraction, multiplication and division) operation's menu. The program will display the result of an operation.

Interface of your program should be similar to the following:

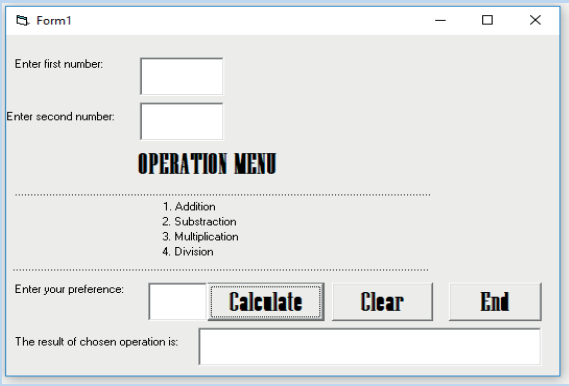


Figure 9.16: user Interface

9.6 Repetition structures

Learing Activity 9.7

3. With an example, explain the use of the following loops in Visual Basic programming:
   - Do While Loop
   - Do Until…. Loop
   - While loop
   - For loop
4. Write a program that prints the numbers divisible by 5 and 3 in the range from 1 to 20
5. Using Do … Loop While, write an application that prints not divisible by 5 in the range from 1 to 10

Repetitions also known as **iterations** or **loops** are structures that allow a statement or group of statements to be carried out repeatedly while some conditions remain true (or for a specified number of times)

They are two basic iteration structures:

a. **Pre-test iterations:** in these loops, the condition to be met occurs at the beginning of the loop and the code will not run at all if the condition is never met. This can be implemented using: **For … Next, Do While …. Loop, Do Until … Loop and While … Wend loop**

b. **Post –test iterations:** in these loops, the condition to be met is at the end of the loop so that the code always runs at least once. This can be implemented using the: **Do … Loop While, Do … Loop Until**

Visual Basic supports two types of loops which are indeterminate loops and determinate loop

9.6.1 Indeterminate loops

These are loops which repeat until a predetermined goal is achieved. Repeating until some initial conditions change. The Indeterminate loops supported in VB are the following:

i. **Do While … Loop**

The general format is:

Do while (condition)
VB statement(s)
Loop

When Visual Basic executes this Do While … Loop, it first tests condition. If condition is False, it skips past all the statements. If it's True, Visual Basic executes the statements and then goes back to the Do while statement and tests the condition again. Consequently, the loop can execute any number of times, as long as condition is True. The statements never execute if initially is False.

Example: a visual basic program to find the sum of number from 0 to 100

```
Private Sub Form_Load()
 Form1.Show
 Dim num As Integer, Total As Integer
 num = 0
 Total = 0
 Do While num <= 100
 Total = Total + num
 num = num + 1
 Loop
 Text1.Text = Total
End Sub
```

Output:

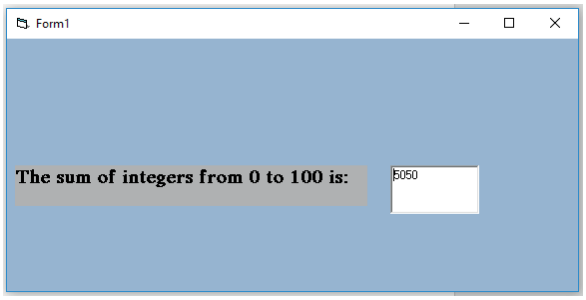


Figure 9.17: Loop results

ii. Do Until … Loop

Unlike the **Do While...Loop** and **While...end** loop structures, the **Do Until... Loop** structure tests a condition for falsity. Statements in the body of a **Do Until...Loop** are executed repeatedly as long as the loop-continuation test evaluates to False.

The general format is:

Do until condition
Block of one or more statements
Loop

Example: a simple program to find even numbers from 10 to 20

```
Private Sub Form_Load()
Form1.Show
Dim number As Long
number = 10
Do Until number >= 20
number = number + 2
Print number
Loop
End Sub
```

The outpout is as follow:

Figure 9.18: Loop results

iii. While …. Wend

A **While...Wend** statement behaves like Do **While...Loop** statement. It executes a set of VB statements till the condition evaluates to true.

The general format is:

```
While condition
 One or more VB statements
 Wend
```

For example: Generate the sum of first 10 integers

```
Private Sub Form_Load ()
 Form1.Show
Dim I As Integer, Sum  As Integer
 I = 0
 Sum = 0
```

```
While I <= 10
Sum = Sum + I
I = I + 1
Wend
Print "The sum of first 10 integers is"; Sum
End Sub
```

The outpu of above code

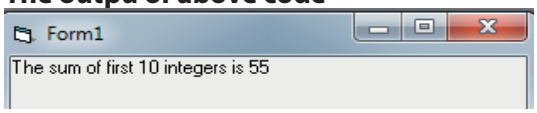


Figure 9.19: ***Loop output***

iv. Do … Loop While

The **Do...Loop** While statement first executes the statements and then test the condition after each execution**.**

Syntax

```
Do
    Block of one or more VB statements
    Loop While condition
```

For Example: Loop counts from 1 to 20 using do ....loop while

```
Private Sub Command1_Click()
Dim number As Long
number = 0
Do
number = number + 1
Print number
Loop While number < 20
End Sub
```

The output of above source code is as follow:

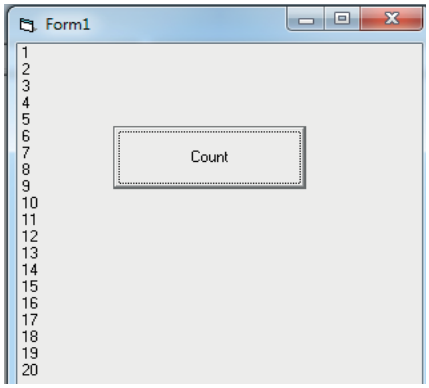


Figure 9.20: Loop output

The programs executes the statements between Do and Loop While structure in any case. Then it determines whether the number is less than 20. If so, the program again executes the statements between Do and Loop While, else exits the Loop.

v. Do … Loop Until

The general format is:

Do
Block of one or more VB statements
Loop Until condition

Example: program to count multiples of three up to 15

```
Private Sub Form_Load()
Form1.Show
Dim Counter =0 As Integer
Do
Counter = Counter + 3
Print Counter
Loop Until Counter = 15
End Sub
```

The output of above source code is as follow:

Figure 9.21: Loop results

APPLICATION ACTIVITY 9.8

1. By using:
   - Do While…. Loop
   - While …Wend loop
   - Do … Loop Until

Write a VB program that computes the sum of even numbers from 0 to 50.

2. Write a VB program that displays the product of even numbers and product of odd numbers between 1 to 15 numbers.

9.6.2 Determinate loop

ACTIVITY 9.9

1. Give and explain the syntax of for… Next loop in VB
2. Write a program that reads a number from user and displays the table of that number.

   2 X 1 = 2

   2 X 2 = 4

   2 X 3 = 6

   . . . . .

   2 X 9 = 18

   2 X 10 = 20

This loop repeats a set of operations a fixed number of times. The common used terminate loop is **For ….next** loop.

The general format is:
Fo*r variable = start* **To** *end*
VB statements to be repeated
Next *variable*

In this syntax statement, **For**, **To**, and **Next** are required keywords, and the assignment operator is required. First, you replace **variable** with the name of a numeric variable that keeps track of the current loop count. Next, you replace s**tart** and **end** with numeric values that represent the starting and stopping points for the loop. The line or lines between the **For** and **Next** statements are the instructions that repeat each time the loop executes.**For... Next** loop structure handles all the details of counter-controlled repetition

Example 1: Suppose you want to print the numbers from 1 to 10.

The following lines of codes can be used in procedure.

```
Private Sub Form_Load()
Dim i As Integer
Form1.Show
For  i = 1 To 10  'step1 is by default
Print i
Next i
End Sub
```

Output:

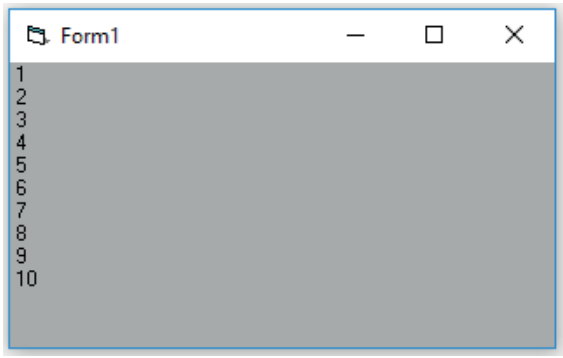


Figure 9.22:Loop output

Example 2: VB application to display the multiplication tables of 5

User interface:

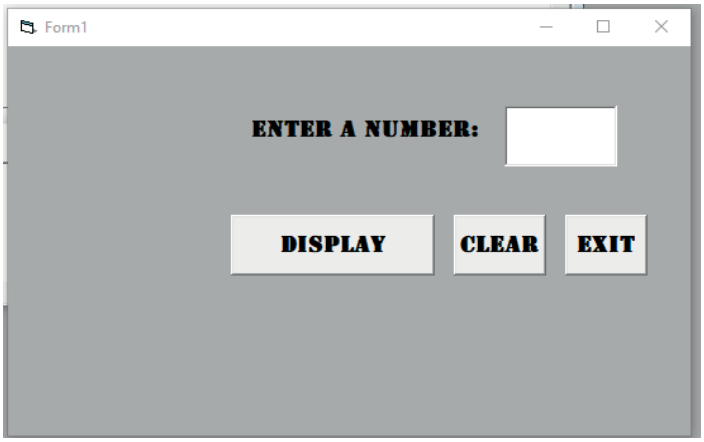


Figure 9.23: User interface

Source code:

```
Private Sub Command1_Click()
Dim a As Integer
a = Val(Text1.Text)
For i = 1 To 10
Print a; "*"; i; "="; a * i
Next i
End Sub
Private Sub Command2_Click()
Text1.Text = ""
Form1.Cls
End Sub
```

Private Sub Command3_Click()
Unload Me
End Sub

Output:

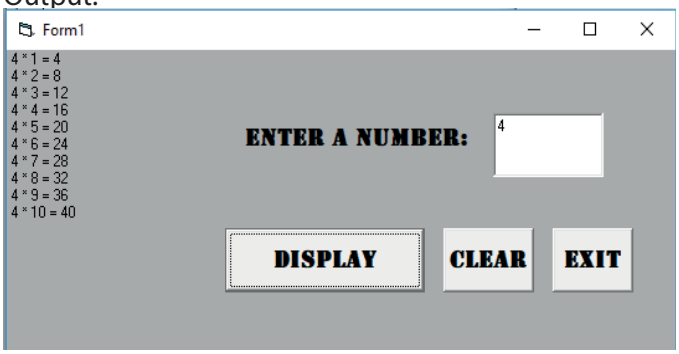


Figure 9.24: Output of the program

Creating Loops with a Custom Counter

You can create a loop with a counter pattern other than 1, 2, 3, 4, and so on. First, you specify a different start value in the loop. Then, you use the Step keyword to increment the counter at different intervals. In this case the syntax for a **For...Next** loop looks like the following:

For counter = initial To End Step [Increment]

One or more VB statements

Next [counter]

The arguments **counter**, **start**, **end**, and **increment** are all numeric. The increment argument can be either positive or negative.

If increment is positive, start must be less than or equal to end or the statements in the loop will not execute. If increment is negative, start must be greater than or equal to end for the body of the loop to execute. If **step** isn't set, then increment defaults to 1.

Example: Print the numbers from 1 to 50 by the increment of 3

Private Sub Form_Load()
Form1.Show
Dim intnum As Integer
For intnum = 1 To 50 Step 3
Print intnum
Next intnum
End Sub

The VB source code above prints this sequence of numbers: 1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49

Specifying Decimal Values

You can also specify decimal values in a loop.

For example:

For i = 1 To 2.5 Step 0.1
Print i
Next i

The program above prints this sequence of numbers as follow:

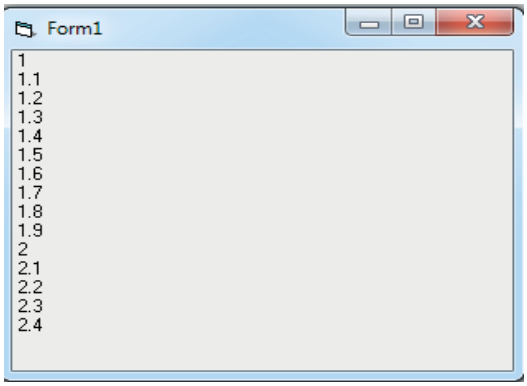


Figure 9.25: Output

Note that:

The **exit** statement allows you to exit directly from **For Loop** and **Do Loop. Exit For** can appear as many times as needed inside a For loop, and **Exit Do** can appear as many times as needed inside a Do loop (the **Exit Do** statement works with all versions of the Do Loop syntax).

Sometimes the user might want to get out from the loop before the whole repetitive process is executed; the command to use is **Exit For** to exit a **For.....Next** Loop or **Exit Do** to exit a **Do… Loop**, and you can place the Exit For or Exit Do statement within the loop.

The exit keyword in loop control statement is used as break Jump control statement in other programming languages to terminate prematurely the loop.

a) Exit For

The format is:

For counter= start To end step (increment)

Statements

Exit for

Statement

Next counter

Example:

```
Private sub Form Load_( )
Form1.show
Dim n as Integer
For n=1 to 10
If n > 6 Then Exit For
Print n
Next n
End Sub
```

Output: 1 2 3 4 5 6

b) Exit Do

The format is:

```
Do While condition
Statements
Exit Do
Statements
Loop
```

Example:

```
Private sub Form _Load( )
Form1.show
Dim x As Integer
X=0
Do While x < 10
Print x
x = x + 1
If x = 5 Then
Print "The program is exited at x=5"
Exit Do
End If
Loop
End Sub
```

The output: 0 1 2 3 4 the program is exited at x=5

Nested Loops

- **The nested loops** are the loops that are placed inside each other.
- The most inner loop will be executed first, then the outer ones.
- Nested loops are very useful when there is a requirement to generate different kinds of patterns as output.

Example: Program to generate the output given below:

1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

Solution:

```
Private Sub Form_Load()
Label1.FontSize = 24
For i = 1 To 5
 For j = 1 To i
   Label1.Caption = Label1.Caption + Str(i) + " "
  Next j
  Label1.Caption = Label1.Caption + vbCr
Next i
End Sub
```

For Each loop

The For Each loop is a scope that defines a list of statements that are to be repeated for all items specified within a certain collection/array of items. The For Each loop, as compared to the For loop, can't be used to iterate from a range of values specified with a starting and ending value.

Syntax of a For Each Loop:

For each variable in Items
Vb codes
Next variable

Where:

Variable is the iterating variable which is used to iterate through the elements of the collection or array

Items: a collection or array of items

Next: closing statement for the loop. Optionally you can specify the **Iterator** variable

Example: a program to display elements of the list

```
Private Sub Form_Load()
Form1.Show
Dim x(7) As Long, a As Variant
x(0) = 30
x(1) = 40
x(2) = 50
x(3) = 60
x(4) = 70
```

```
x(5) = 80
x(6) = 90
x(7) = 100
For Each a In x
  Print a
Next a
End Sub
```

The output is

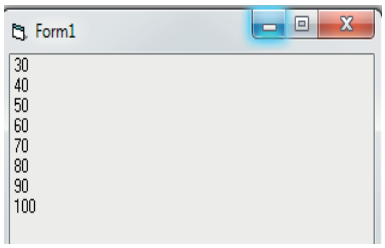


Figure 9.26: program output

APPLICATION ACTIVITY 9.9

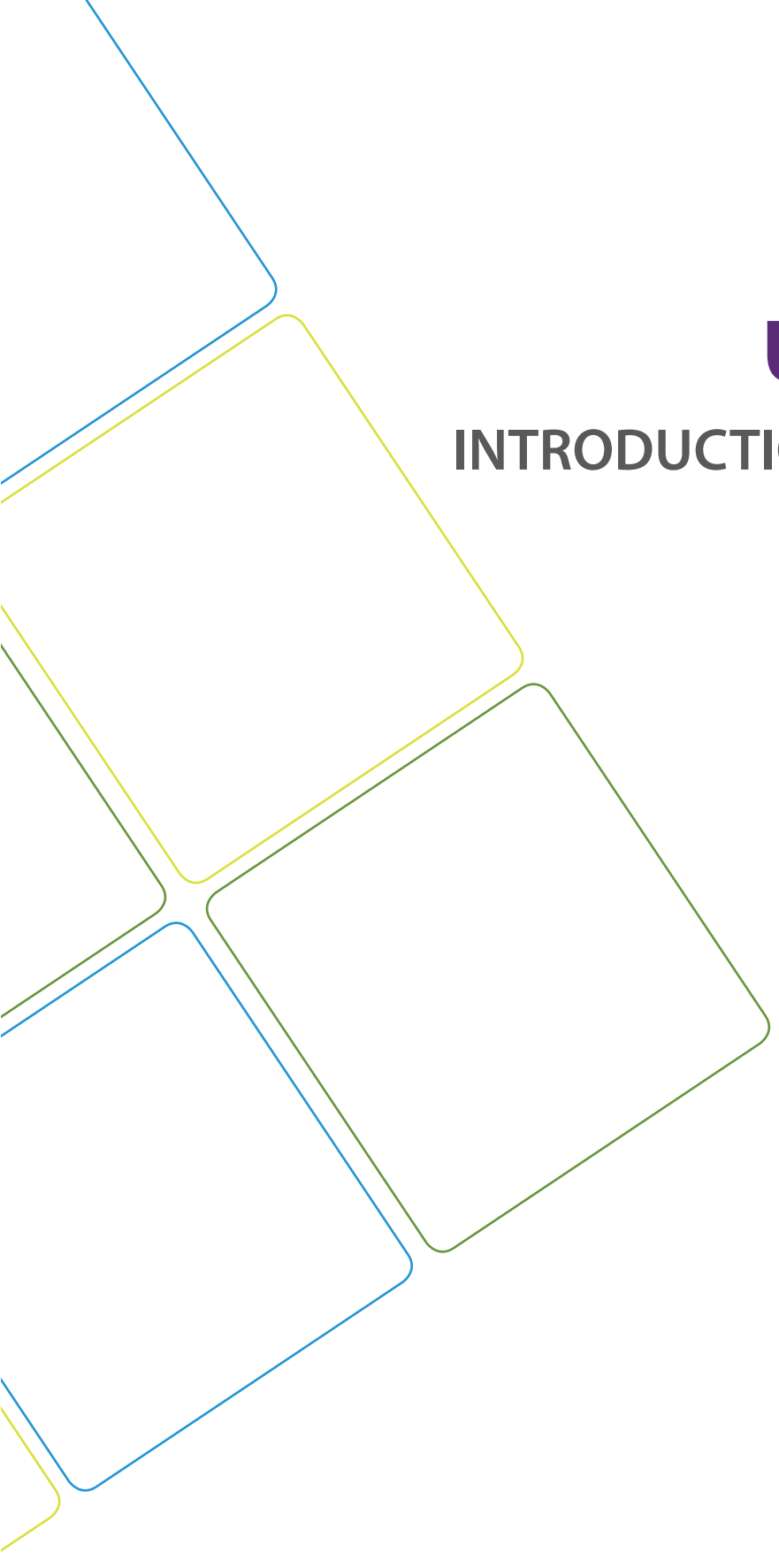
1. Study the following program in Visual Basic and write its output:

```
 Private Sub Form_Load()
Form1.Show
Dim firstCounter, secondCounter, thirdCounter As Integer
For firstCounter = 1 To 2
Print " Hello My Friend! "
Print " "
    For secondCounter = 1 To 3
    Print " Welcome to VB tutorial! "
        For thirdCounter = 8 To 2 Step -2
        Print "Happy Learning!"
         Next thirdCounter
     Next secondCounter
Next firstCounter
Print " "
Print " Thanks a lot! "
End Sub
```

2. Write a VB program to display the following sequence of numbers 100,80,60,40,20,0
3. Write a VB application that prints the numbers divisible by 5 or 3 but not both in the range from 1 to 20

END UNIT ASSESSMENT

1. Write a VB program to calculate the values of the following arithmetic operation if the user inputs two numbers M and N into two separate text boxes
   a. M^N
   b. M\N
   c. M Mod N
2. Build an application to calculate the volume of the cylinder
3. Differentiate nested if to switch case decision control structures
4. Write a VB application that prints the numbers divisible by 5 or 3 in the range from 1 to 20
5. Write a VB program that displays odd numbers between 1 to 30 numbers
6. Design an application to compute the sum of the first "N" natural numbers.
7. Draw an interface and write a VB application to check if a given number is prime or not.



UNIT 10

INTRODUCTION TO JAVA

UNIT 10: INTRODUCTION TO JAVA

Key Unit Competency:

to be able to create, build and run a java console program

Introductory activity

1. Explain the following terms in computer programming:
   a) Computer Programming b) Programming language
   c) High-level language d) Low-level language
   e) Programming paradigms
2. Discuss the following OOP concepts:
   a) Abstraction c) Encapsulation
   b) Polymorphism d) Inheritance
3. Explain the following concepts in programming
   a) Compile and run program
   b) Programming Environment c) Basic Syntax d) Data Types
   d) Variables e) Keywords f) Basic Operators
   g) Decision Making h) Loops

10.1. Concepts of Java Programming

Learning Activity 10.1

1. Define a java in Programming language
2. Discuss on invention and advantages of java.

10.1.1. Definition

Java is a computer programming language. It enables programmers to write computer instructions using English-based commands instead of having to write in numeric codes. It is known as a high-level language because it can be read and written easily by humans

10.1.2. Invention

Java is a general purpose, high-level programming language developed by Sun Microsys-

tems. The Java programming language was developed by a **small team** of engineers, known as the *Green Team*, who initiated the language in **1991**. The Java language was originally called *OAK,* and at the time it was designed for handheld devices and set-top boxes.

OAK was unsuccessful and in 1995 Sun Microsystems changed the name to Java, which has built-in support to create programs with a Graphical User Interface **(GUI),** utilizes the Internet, creates client-server solutions, and modified the language to take advantage of the burgeoning (developing quickly) World Wide Web.

The latest release of the Java Standard Edition is Java SE 8. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

The new J2 versions were renamed as Java SE, Java EE, and Java ME respectively. Java is guaranteed to be **Write Once, Run Anywhere.**

A variant of Java programs called applets can be embedded inside a web page and execute on the computer that is viewing the page, automatically and in a secure environment.

10.1.3. Advantages of Java

Advantages	Explanations
Simple	Java is designed to be easy to learn. With the understanding of the basic concept of OOP Java, it would be easy to master.
Secure	With Java's secure feature, it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
Architecture-neutral	Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
Portable	Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
Robust	Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.

Multithreaded	With Java's multithreaded feature, it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
Interpreted	Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
High Performance	With the use of Just-In-Time compilers, Java enables high performance.
Distributed	Java is designed for the distributed environment of the internet.
Dynamic	Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

Application Activity 10.1.

Discuss the difference between Java Programming and C/C++ programming languages in terms of advantages.

10.2. Platforms of JAVA Program

Learning Activity 10.2.

Search and discuss about the following in Java Environment:

1. Platform and Platform independent
2. Java Virtual Machine (JVM)
3. Java Runtime Environment (JRE) vs. Java Development Kit (JDK)

10.2.1 JRE & JDK & JVM

The Java platform is the name given to the computing platform from Oracle that helps users to run and develop Java applications. The platform does not just enable a user to run and develop a Java application, but also features a wide variety of tools that can help developers work efficiently with the Java programming language.

The platform consists of two essential pieces of software:

- **Java Runtime Environment (JRE):** Thisis the environment within which the java virtual machine runs. JRE contains Java virtual Machine (JVM), class libraries, and other files excluding development tools such as compiler and debugger. Note: The JRE takes care of running the Java code on multiple platforms, however as developers, we are interested in writing pure code in Java which can then be converted into Java byte-code for mass deployment

- **Java Development Kit (JDK),** which is needed to develop those Java applications and applets(a small dynamic Java program that can be transferred via the Internet and run by a Java-compatible Web browser). If you have installed the JDK, you should know that it comes equipped with a JRE as well. JDK contains everything that JRE has along with development tools Such as compiler debugger etc.

- The Java platform consists of the Java Application Programming Interfaces (APIs) and the Java Virtual Machine (JVM). Java APIs are libraries of compiled code that you can use in your programs.

- They let the programmers add ready-made and customizable functionality to save them programming time.

- Any program uses Java API to print a line of text to the console.

- The console printing capability is provided in the API ready for programmers to use; programmers supply the text to be printed. Java programs are run (or interpreted) by another program called the Java VM, for the native operating system. Any computer system with the Java VM installed can run Java programs regardless of the computer system on which the applications were originally developed.

For example, a Java program developed on a Personal Computer (PC) with the Windows NT operating system should run equally well without modification on Macntosh computer running Mac OSx operating system, and vice versa.

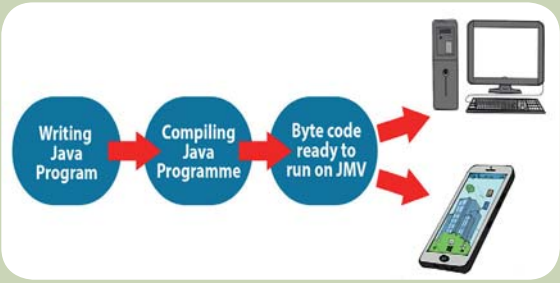
Application activity 10.2.

Discuss the difference between JVM vs JRE vs JDK

10.3 Write, Compile and run a java Program

Learning Activity 10.3.

1. Using internet, Search and install step by step the environment of java.
2. Observe and interpret the following picture:



10.3.1 Create a java program

Before writing and running the simple Java program, there is a needto install and to configure the Java platform. Which is available free of charge from the Java web site.(for example NetBeans IDE 8.0 or NetBeans IDE 8.2)

The first application, `HelloWorldApp`, will simply display the greeting "Hello World!"To create this program, the following steps must be respected

Create an IDE (Integrated Development Environment) project: Creating an IDE project means creating an environment in which to build and run the applications. Using IDE projects eliminates configuration issues normally associated with developing on the command line. Build or running an application can be done by choosing a single menu item within the IDE.

a) Launch the NetBeans IDE.

- On Microsoft Windows systems, use the NetBeans IDE item in the Start menu.
- On Solaris OS and Linux systems, execute the IDE launcher script by navigating to the IDE's bin directory and typing. /netbeans.
- On Mac OS X systems, click the NetBeans IDE application icon.

b) Choose **File** | **New Project...**.

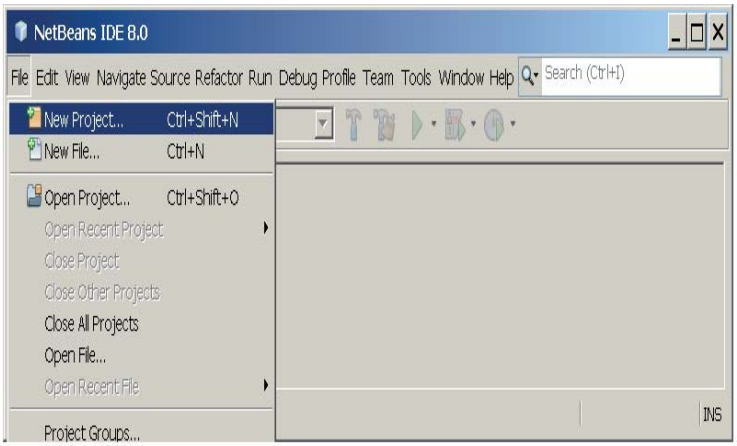


Figure 10.1 NetBeans IDE with the File | New Project menu item selected.

In the **New Project** wizard, expand the **Java** category and select **Java Application** as shown in the following figure:

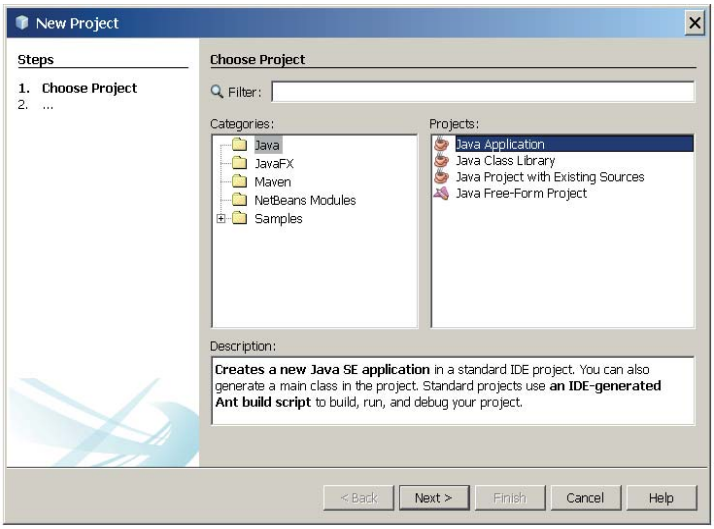


Figure 10.2 NetBeans IDE, New Project wizard, Choose Project page.

In the **Name and Location** page of the wizard, do the following (as shown in the figure below):

- In the Project Name field, type HelloWorldApp.
- In the Create Main Class field, type helloworldapp.HelloWorldApp.
- Then click the button Finish in the figure below.

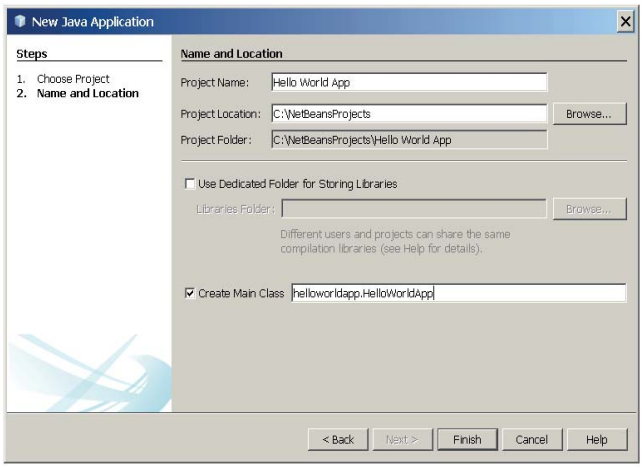


Figure 10.3 New java application window

The project is created and opened in the IDE. In the next figure, there are the main components of the environment

- The Projects window, which contains a tree view of the components of the project, including source files, libraries that the code depends on, and so on.
- The Source Editor window with a file called HelloWorldApp.java open.
- The Navigator window, used to quickly navigate between elements within the selected class.

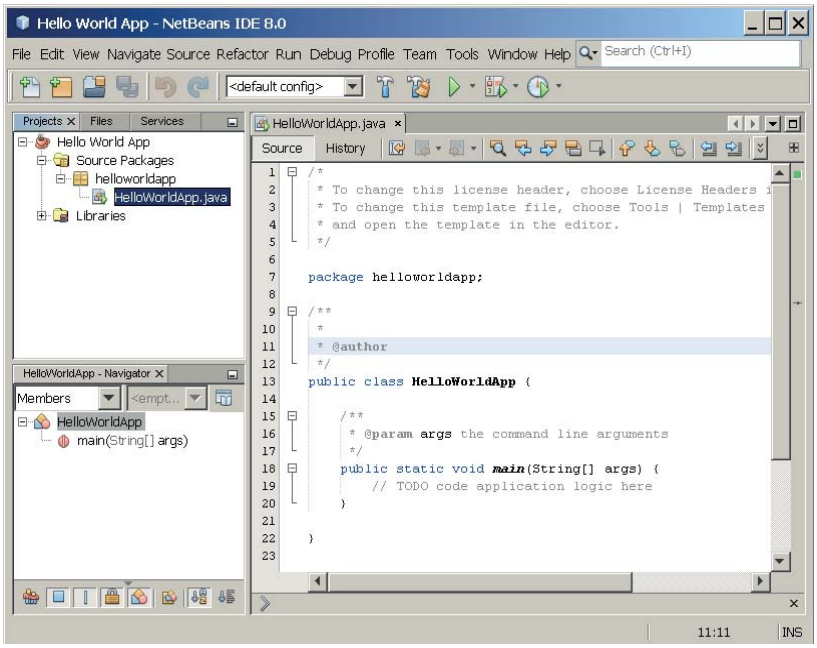


Figure 10.4 NetBeans IDE with the HelloWorldApp project open

Add Code to the Generated Source File: A source file contains code, written in the Java programming language, that programmers can understand. As part of creating an IDE project, a skeleton source file is automatically generated and the source file is then modified to add the "Hello World!" message. When this project was created, the **Create Main Class** checkbox was left selected in the **New Project** wizard. The IDE has therefore created a skeleton class for the programmer. The «Hello World!» message is added to the skeleton code by replacing the line:

// TODO code application logic here

With the line:

System.out.println("Hello World!"); // Display the string.

Optionally, you can replace these four lines of generated code:

```
/**
 *
 * @author
 */With these lines:
/**
 * The HelloWorldApp class implements an application that
 * simply prints "Hello World!" to standard output.
 */
```

These four lines are a code comment and are not affecting how the program runs. Later sections of this unit explain the use and format of code comments.

Notice that Java programming is case sensitive. It means that "a" is different from "A", **`helloWorldApp`** is also different from **`HelloWorldApp`**.

After adding the code, the remaining action is to Save the changes by choosing File | Save.

The file should look something like the following:

```
/* * To change this template, choose Tools | Templates* and open the template in the editor. */
packagehelloworldapp;
/**
 * The HelloWorldApp class implements an application that
 * Simply prints "Hello World!" to standard output.
 */
public class HelloWorldApp {
  /**
   * @paramargs the command line arguments
   */
```

```
public static void main(String[] args) {
System.out.println("Hello World!"); // Display the string.
  }
}
```

10.4. Compile the Source File into a .class File

The IDE invokes the Java programming language *compiler* (`javac`), which takes the source file and translates its text into instructions that the Java virtual machine can understand. The instructions contained within this file are known as *byte codes*.To compile the source file, choose **Run** | **Build Project (Hello World App)** from the IDE›s main menu.

The Output window opens and displays output similar to what is seen in the following figure:

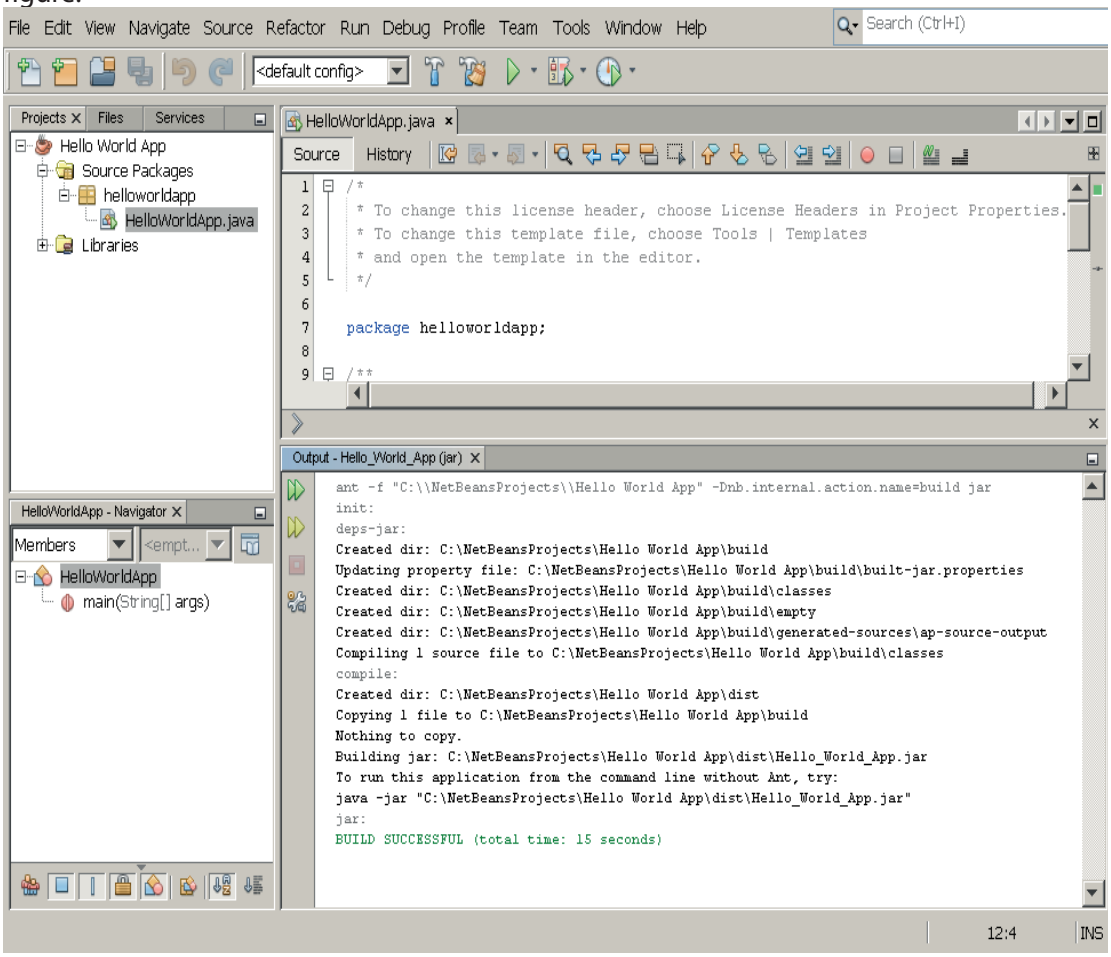


Figure 10.5. Output window showing results of building the HelloWorld project.

If the build output concludes with the statement `BUILD SUCCESSFUL`, congratulations!, The

program is successfully compiled. If the build output concludes with the statement `BUILD FAILED`, there is probably a syntax error in the code.

Errors are reported in the Output window as hyperlinked text. To navigate to the source of an error, double-click such a hyperlink. After fixing the error, once again choose **Run** | **Build Project**.

10.5 Running the Program

The IDE invokes the Java application launcher tool(java), which uses the Java virtual machine to run your application.

From the IDE's menu bar, choose **Run** | **Run Main Project**.

The result is displayed in the next figure.

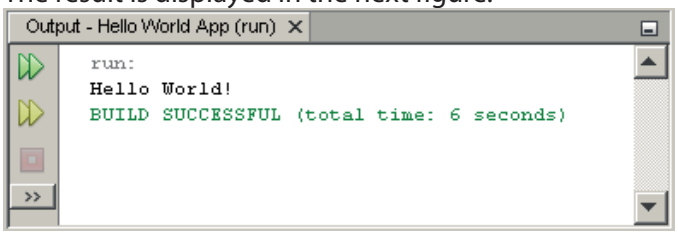


Figure 10.6. Output for the "Hello World" java program

After successful compilation and run, the program prints "Hello World!" to the Output window (along with other output from the build Scripts.

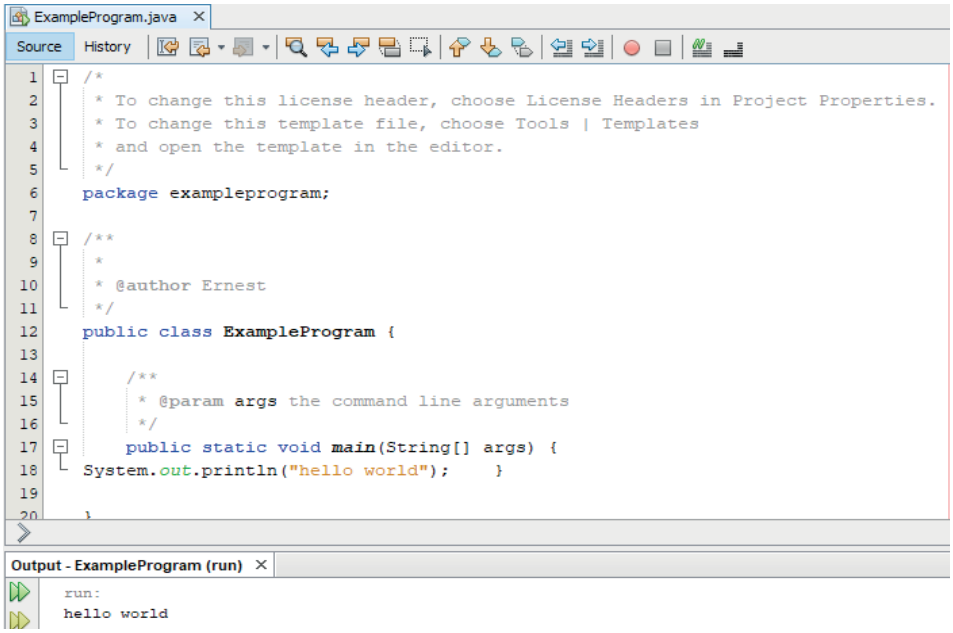


Figure 10.6 Hello world program

10.5.1 Syntax of a java program

```
class class_name
{
 public static void main(String[] args)
 {
// java code goes here
 }
}
```

Explanation of the syntax

i. **class** : class keyword is used to declare classes in Java

ii. **public** : It is an access specifier. Public means this function is visible to all.

iii. **static** : static is again a keyword used to make a function static. To execute a static function you do not have to create an Object of the class. The main() method here is called by JVM, without creating any object for class.

iv. **void** : It is the return type, meaning this function will not return anything.

v. **main** : main() method is the most important method in a Java program. This is the method which is executed, hence all the logic must be inside the main() method. If a java class is not having a main() method, it causes compilation error.

vi. **String[] args** : This represents an array whose type is String and name is args. We will discuss more about array in Java Array section.

Application activity 10.3

1. Type The program prints "Hello World!", then compile and run it.
2. Add a print statement to display the sentence "How are you?" Compile and run the program again.
3. Add a comment to the program (anywhere), recompile, and run it again. The new comment should not affect the result.
4. Write a program in Java that after compilation and execution will display the messages: "Unity and Peace are the engines of development in Rwanda" and "Rwanda is a Knowledge Based Society".

10.5.2 Variables and Data types in Java

Learning activity 10.4.

1. Discuss about data type in term of Definition and Classification
2. Using internet search a variable in java and how is declared.
3. Given two integer x= 5; y= 4; write a java program which is calculate and display the sum of these two integers.

The declaration of variables necessitates always specifying their types. Variables cannot exist without types and vice versa. One of the most powerful features of a programming language is the ability to define, declare and to manipulate variables.

For example, String message;
This statement is a declaration, because it declares that the variable named **message** is of type **String**. Each variable has a type that determines what kind of values it can store

A. Data type

Data type is a classification of data which tells the compiler or interpreter how the programmer intends to use the data. Most programming languages support various types of data, for example: real, integer or Boolean.

Classification of Java Data type

i. Primitive Java Data Types

There are eight primitive data types supported by Java and are predefined by the language and named by a keyword. They are presented in the following table.

Type	Description
Byte	• Byte data type is an 8-bit signed two's complement integer • Minimum value is -128 (-2^7) • Maximum value is 127 (inclusive)(2^7 -1) • Default value is 0 • Byte data type is used to save space in large arrays, mainly in place of integers, since a byte is four times smaller than an integer. Example: byte a = 100, byte b = -50
Short	• Short data type is a 16-bit signed two's complement integer • Minimum value is -32,768 (-2^15) • Maximum value is 32,767 (inclusive) (2^15 -1) • Short data type can also be used to save memory as byte data type. A short is 2 times smaller than an integer • Default value is 0. Example: short s = 10000, short r = -20000

Int	• Int data type is a 32-bit signed two's complement integer. • Minimum value is - 2,147,483,648 (-2^31) • Maximum value is 2,147,483,647(inclusive) (2^31 -1) • Integer is generally used as the default data type for integral values unless there is a concern about memory. • The default value is 0 Example: int a = 100000, int b = -200000
Long	• Long data type is a 64-bit signed two's complement integer • Minimum value is -9,223,372,036,854,775,808(-2^63) • Maximum value is 9,223,372,036,854,775,807 (inclusive)(2^63 -1) • This type is used when a wider range than int is needed • Default value is 0L Example: long a = 100000L, long b = -200000L
Float	• Float data type is a single-precision 32-bit IEEE 754 floating point • Float is mainly used to save memory in large arrays of floating point numbers • Default value is 0.0f • Float data type is never used for precise values such as currency Example: float f1 = 234.5f
Double	• Double data type is a double-precision 64-bit IEEE 754 floating point • This data type is generally used as the default data type for decimal values, generally the default choice • Double data type should never be used for precise values such as currency • Default value is 0.0d Example: double d1 = 123.4
Boolean	• Boolean data type represents one bit of information • There are only two possible values: true and false • This data type is used for simple flags that track true/false conditions • Default value is false Example: boolean one = true
Char	• char data type is a single 16-bit Unicode character • Minimum value is (0) • Maximum value is (or 65,535 inclusive) • Char data type is used to store any character Example: char letterA = 'A'

Table 10.1. Primitive data types

ii. Reference Data types or Objects

Reference variables are created using defined constructors of the classes. They are used to access objects. These variables are declared to be of a specific type that cannot be changed. For example, employee, puppy, etc.

- Class objects and various types of array variables come under reference data type.
- Default value of any reference variable is null.
- A reference variable can be used to refer any object of the declared type or any compatible type.

Example: Animal animal = new Animal("giraffe");

B. Variable in Java

A *variable* is a storage location (like a house, a pigeon hole, a letter box) that stores a piece of data for processing. It is called *variable* because you can change the value stored inside. More precisely, a *variable* is a *named* storage location, that stores a *value* of a particular data *type*.

The syntax for declaration of variable is as follow:
data_type variable_name;

Examples

To declare an integer variable named x, simply type:
int x
String message = "Hello!";
int hour = 11;
int minute = 59;

Normally, the used names for variables should help to guess what will be the different values to assign to them.

Example
String firstName, lastName;
int hour, minute;

Sometimes a variable is initialized by assigned to it for the first time value before it can be used or it a variable can be assigned assign a value later, as in the previous example. It is also possible to declare and initialize on the same line.

Printing variables

The value of a variable can be displayed by using the functions **print or println**. The following statements declare a variable named firstLine, assign it the value "Hello, again!", and

display that value.
String firstLine = "Hello, again!";
System.out.println(firstLine);

Displaying a variable means displaying the value of that variable. To display the name of a variable, you have to put it into quotes.
System.out.print("The value of firstLine is ");
System.out.println(firstLine);

For this **example**, the output is:
The value of firstLine is Hello, again!

Conveniently, the syntax for displaying a variable is the same regardless of its type.

For example:
int hour = 11;
int minute = 59;
System.out.print("The current time is ");
System.out.print(hour);
System.out.print(":");
System.out.print(minute);
System.out.println("");

The output of this program is: The current time is 11:59.

Application activity 10.4

Write a program where you declare the variables firstname, lastname, age, level and com bination. This program should display like this: John Ntwali is 18 years old and in senior 5 of MPC

10.5.3 Types of Variables in Java

Learning activity 10.5

1. Is a variable declared visible always in all parts of the program? Explain why and give the different types.
2. Discuss about Java Expression.

In terms of visibility of variables in a program, there are three **types of** in Java namely local variable, instance variable and class variable**.**

1. Local Variables

This type of variable has the following characteristics:

- Local variables are declared in methods, constructors, or blocks.

Local variables are created when the method, constructor or block is entered and the variable will be destroyed once it exits the method, constructor, or block.

- Access modifiers cannot be used for local variables.
- Local variables are visible only within the declared method, constructor, or block.
- Local variables are implemented at stack level internally.
- There is no default value for local variables, so local variables should be declared and an initial value should be assigned before the first use.

Example: Here, *age* is a local variable. This is defined inside *pupAge()* method and its scope is limited to only this method.

```
Public class Test{
public void pup Age(){
int age =0;
age= age +7;
System.out.println("Puppy age is : "+ age);
}
Public static void main(String args []){
Test test=new Test();
test.pupAge();}}
```

2. Instance Variables

The variables defined within a class or method is called instance variables because each instance of the class (that is, each object of the class) contains its own copy of these variables. Thus, the data for one object is separate and unique from the data for another. An instance variable can be declared public or private or default (no modifier).To avoid that the variable's value is changed out-side its class, it should be declared as private. Public variables can be accessed and changed from outside of the class. The syntax is shown below given below:
private int doors;

Whereby:

"private" is an access modifier

"int" is data type

doors is an identifier

These variables belong to the *instance of a class*, thus an object. And every instance of that class (object) has it's own copy of that variable. Changes made to the variable don't reflect in other instances of that class.

3. Class or static variables

These are also known as *static member variables* and there›s only one copy of that variable that is shared with all instances of that class. If changes are made to that variable, all other instances will see the effect of the changes.

```
Public class Product{
Public static int Barcode;
}
```

Example:

```
public class Demo {
 static int a;
 static int b;
 static {
   a = 10;
   b = 20;
 }
 public static void main(String args[]) {

  System.out.println("Value of a = " + a);
  System.out.println("Value of b = " + b);

        }}
```

Output

```
run:
Value of a = 10
Value of b = 20
BUILD SUCCESSFUL (total time: 2 seconds)
```

10.5.4 Java keywords

Keywords are words that have already been defined for Java compiler. They have special meaning for the compiler. Java Keywords must be in your information because you cannot use them as a variable, class or a method name. Java keywords are categorized as datatypes and access modifiers

A. Data types

All predefined datatypes in java are keywords eg. int, char, bool, double, float, and String

B. Java access modifiers

Modifiers are keywords that are added to change meaning of a definition. In Java, modifiers are categorized into two types,

- Access control modifier
- Non Access Modifier

1) Access control modifier

Java language has four access modifiers to control access levels for classes, variable methods and constructors.

- **Default** : Default has scope only inside the same package
- **Public** : Public has scope that is visible everywhere
- **Protected** : Protected has scope within the package and all sub classes
- **Private** : Private has scope only within the classes

2) Non-access Modifier

Non-access modifiers do not change the accessibility of variables and methods, but they do provide them special properties. Non-access modifiers are of 5 types,

- Final
- Static
- Transient
- Synchronized
- Volatile

1) Final: Final modifier is used to declare a field as final i.e. it prevents its content from being modified. Final field must be initialized when it is declared. Final keyword can be used with a variable, a method or a class.

2) Static Modifier: Static Modifiers are used to create class variable and class methods which can be accessed without instance of a class.

3) Synchronized modifier: When a method is synchronized it can be accessed by only one thread at a time.

4) Volatile modifier: Volatile modifier tells the compiler that the volatile variable can be changed unexpectedly by other parts of the program. Volatile variables are used in case of multithreading program. volatile keyword cannot be used with a method or a class. It can

be only used with a variable.

5) Transient modifier: When an instance variable is declared as transient, then its value doesn't persist when an object is serialized

10.5.6 Java Expressions

Expressions are essential building blocks of any Java program, usually created to produce a new value, although sometimes an expression simply assigns a value to a variable. Expressions are built using values, variables, operators and method calls.

Types of Expressions

While an expression frequently produces a result, it doesn't always. There are three types of expressions in Java:

- Those that produce a value, i.e. the result of $(1 + 1)$, Expressions that produce a value use a wide range of Java arithmetic, comparison or conditional operators. For example, arithmetic operators include +, *, /, <, >, ++ and %.
- Those that assign a variable, for example $v = 10$
- Those that have no result but might have a "side effect" because an expression can include a wide range of elements such as method invocations or increment operators that modify the state (i.e. memory) of a program.

Example: This program here contains plenty of expressions

```
public class MyClass {
public static void main(String args[]) {
int x=10;// assign a variable Expressions
int y=25;// assign a variableExpressions
int z=x+y;// produce a value Expressions
System.out.println("Sum of x+y = " + z);   }
```

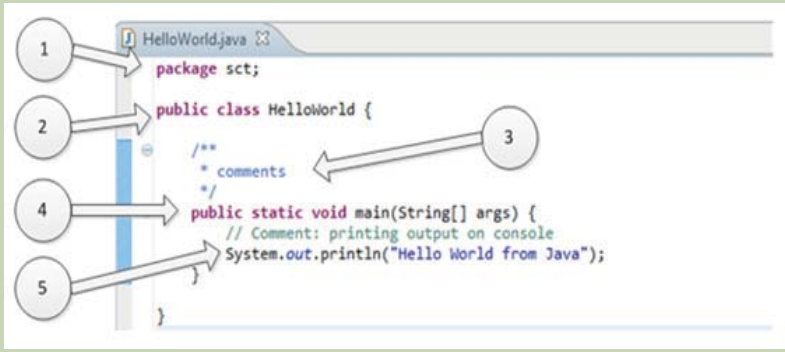
Application Activity 10.5

Look at the following three groups of statements and predict the outputs

int x = 10, y; y = x+1; System.out.print(y); **Show the answer.**	int x = 10, y; y = x+1; System.out.print(x); **Show the answer.**	int result = 4 + 5 * 6 + 2; System.out.print(result); Show the answer. int a = 5 + 7 % 2; System.out.print(a); **Show the answer.**

10.6. Elements of Java source file

Learning Activity 10.6.



1. Refer to the numbers in above figure; explain the different parts of this program.
2. Following the above figure, write a program with a class that has more than one property.

1. Class:

A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. Classes and Objects are basic concepts of Object Oriented Programming which revolve around the real life entities

Example:

```
/* Display a message */
Class Hello
{
public static void main(String[] args)
{
System.out.println("Hello World!");
}
}
```

Explanation of the above program

Java program consists of a named class(Hello)

The body of the class is surrounded by braces

Almost every Java program must have one and only one main ( ) function

The body of the function is surrounded by brackets. Statements can be combined within braces to form a block statement which can be used wherever a statement is required by the Java syntax

2. **Object:**

It is a basic unit of Object Oriented Programming and represents the real life entities. A typical Java program creates many objects, which interact by invoking methods. An object consists of:

- **State:** It is represented by attributes of an object. It also reflects the properties of an object.
- **Behavior:** It is represented by methods of an object. It also reflects the response of an object with other objects.
- **Identity:** It gives a unique name to an object and enables one object to interact with other objects.

3. **Method:**

It is a unit of program code, within a class designed to perform specific task or function. A method is a program module that contains a series of statements that carry out a task. A method is executed by invoking or calling it from another method. Any class can contain an unlimited number of methods, and each method can be called an unlimited number of times.

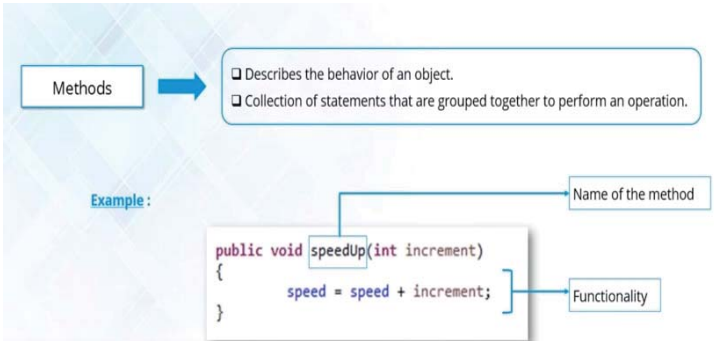
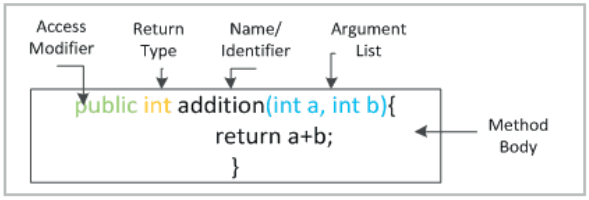


Figure 10.3 methods

- The syntax to declare method:



Within a class, a method can be used as follows:

```
class classname {
    // declare instance variables
    type var1;
    type var2;
    // ...
    type varN;
    // declare methods
    type method1(parameters) {
       // body of method
    }
    type method2(parameters) {
       // body of method
    }
       // ...
    type methodN(parameters) {
      // body of method
    }
}
```

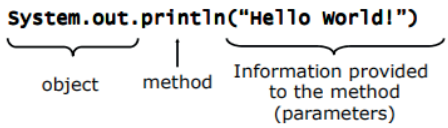
Example:

/ Display a message */*
class Hello {
public static void main(String[] args){
System.out.println("Hello World!");
} }

The identifier System.out is an object

The identifier println is one of the methods for that object

Explanation:



- A method **println(…)** is a service that the **System.out** object can perform
  - This is the object of type **PrintStream**, declared in **java.lang.System** class
- Method **println(…)** is invoked (or called)
- The method **println(…)** is called by sending the message to the **System.out** object, requesting the service

4. Statements:

A programming *statement* is the smallest independent unit in a program, just like a sentence in the English language. It performs *a piece of programming action*.

Statements are:

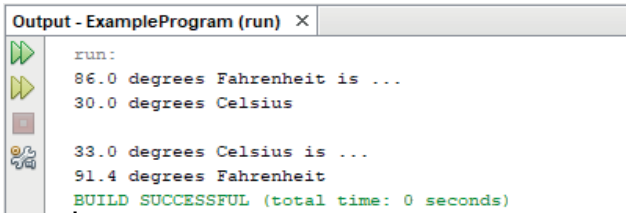
- Variable declarations: primitive data types and classes
- Operations: arithmetic, logical, bit-level, class access

- Control structures: selection, looping, etc.
- Object messages: i.e., calls to methods

A statement ends with a semi-column (;).

Example

```
class Weather {
public static void main(String[] arguments) {
float fah = 86;
System.out.println(fah + " degrees Fahrenheit is ...");
// To convert Fahrenheit into Celsius
// Begin by subtracting 32
fah = fah - 32;
// Divide the answer by 9
fah = fah / 9;
// Multiply that answer by 5
fah = fah * 5;
System.out.println(fah + " degrees Celsius\n");
Float cel = 33;
System.out.println(cel + " degrees Celsius is ...");
// To convert Celsius into Fahrenheit
// Begin by multiplying it by 9
cel = cel * 9;// Divide the answer by 5
cel = cel / 5;
// Add 32 to the answer
cel = cel + 32;
System.out.println(cel + " degrees Fahrenheit");
}}
```



Java naming conventions

Java **naming convention** is a rule to follow when naming the identifiers such as class, package, variable, constant, method etc. But, it is not forced to follow. So, it is known as convention not rule. All the classes, interfaces, packages, methods and fields of java

programming language are given according to java naming convention.

By using standard Java naming conventions, the code is made easier to read for for programmers. Readability of Java program is very important. It indicates that less time is spent to figure out what the code does.

Name	Convention
class name	should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc.
interface name	should start with uppercase letter and be an adjective e.g. Runnable, Remote, ActionListener etc.
method name	should start with lowercase letter and be a verb e.g. actionPerformed(), main(), print(), println() etc.
variable name	should start with lowercase letter e.g. firstName, orderNumber etc.
package name	should be in lowercase letter e.g. java, lang, sql, util etc.
constants name	should be in uppercase letter. e.g. RED, YELLOW, MAX_PRIORITY etc.

Syntax:

```
package details                    ──────►  import java.io.*
class className                    ──────►  class Sum
{
Data members;                      ──────►  int a, b, c;
user_defined method;               ──────►  void display();
public static void main(String args[])
{
Block of Statements;               ──────►  System.out.println("Hello Java !");
}
}
```

Example :

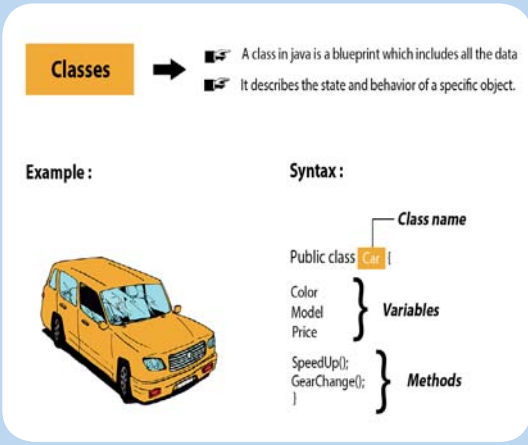
Import java.io.*;

Class Class

```
import java.io.*;
class ClassName
{
  int a=2;
  int b=3;
  void addition()
   {
    System.out.println("method");
   }
  public static void main(String args[])
  {
   int c=2;
   System.out.println(c);
  }
}
```

Package Details
Class Name
Data Member of Class
User Define Method
Main Method
Data Member in main
Block Statement
it will print the value of c

Application activity 10.6



The figure above is an example of Car object which has various properties like color; max speed etc. along with, it has functions like run and stop. How in Java program will be represented?

10.7 Flow control

Learning activity 10.7

1. Discuss about the the following flow statements in C++
   i. Conditional statements
   ii. Loop statement
2. Differentiate while from do-while loop

An execution path is a sequence of statements that the computer executes as it runs a program. A control statement tells the computer to divert onto a different path. If a program has no control statements, the JVM executes it, statement by statement, in sequential order. Often programmers refer to an execution path as a flow of control.

The statements that control the flow of execution through a method fall into two categories:

10.7.1 Conditionals and loops

10.5.1. Conditional statements

a. The IF Statement: An if statement can be followed by an optional else statement, which executes when the Boolean expression is false. The Java if statement has the

following syntax:

```
if(Boolean_expression) {
  // Executes when the Boolean expression is true
}else {
  // Executes when the Boolean expression is false
}
```

If the Boolean condition is true, the statement is executed; if it is false, the statement is skipped.

Example

```
Public class Test{
Public static void main(String args[]){
int x =30;
if( x <20){
System.out.print("This is if statement");
}else{
System.out.print("This is else statement");
}}}
```

b. The IF-ELSE Statement

An else clause can be added to an **if** statement to make it an **–elseif** statement.

The following is the syntax of an if...else statement

```
If (Boolean_expression 1) {
  // Executes when the Boolean expression 1 is true
}else if(Boolean_expression 2) {
  // Executes when the Boolean expression 2 is true
}else if(Boolean_expression 3) {
   // Executes when the Boolean expression 3 is true
}else {
   // Executes when the none of the above condition is true.
}
```

Example

```
Public class Test{
Public static void main (String args[]){
int x =30;
if( x = =10){
System.out.print("Value of X is 10");
}elseif( x = =20){
```

```
System.out.print("Value of X is 20");
}elseif( x = =30){
System.out.print("Value of X is 30");
}else{
System.out.print("This is else statement");
}}}
```

c. The SWITCH Statement in Java

A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

The syntax of switch statement is:

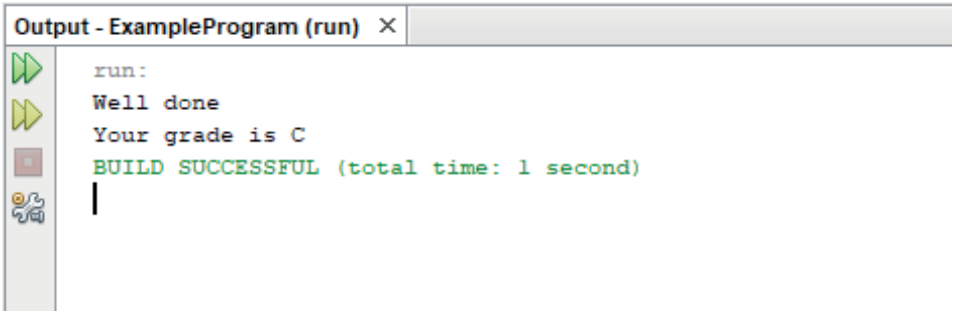
```
switch(expression) {
case value :
   // Statements
break; // optional
case value :
   // Statements
break; // optional

 // You can have any number of case statements.
default : // Optional
   // Statements
}
```

Example:

```
Public class Test{
Public static void main(String args[]){
// char grade = args[0], char At(0);
char grade ='C';
switch(grade){
case 'A':
System.out.println("Excellent!");
break;
case 'B':
case 'C':
System.out.println("Well done");
break;
case'D':
System.out.println("You passed");
```

```
case'F':
System.out.println("Better try again");
break;
default:
System.out.println("Invalid grade");
}System.out.println("Your grade is "+ grade);
}}
```



10.7.1.2 Loops in java

Looping in programming languages is a feature which facilitates the execution of a set of instructions/functions repeatedly while some condition evaluates to true. Java provides three ways for executing the loops.

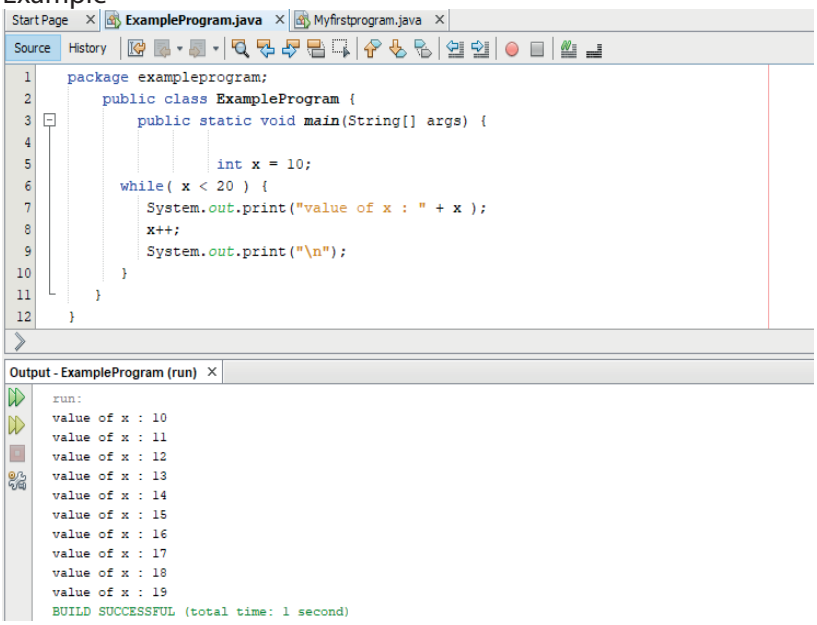
SN	Loop	Description
1	while loop	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.A while loop statement in Java programming language repeatedly executes a target statement as long as a given condition is true.
2	for loop	Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
3	do...while loop	Like a while statement, except that it tests the condition at the end of the loop body.

a. The while loop

The syntax of the while loop is as shown below;

```
initial_value
while(Boolean_expression) {
  // Statements
}
```

Example



b. Thedo...while loop

This loop is similar to the while loop, except that a do...while loop is guaranteed to execute at least one time.

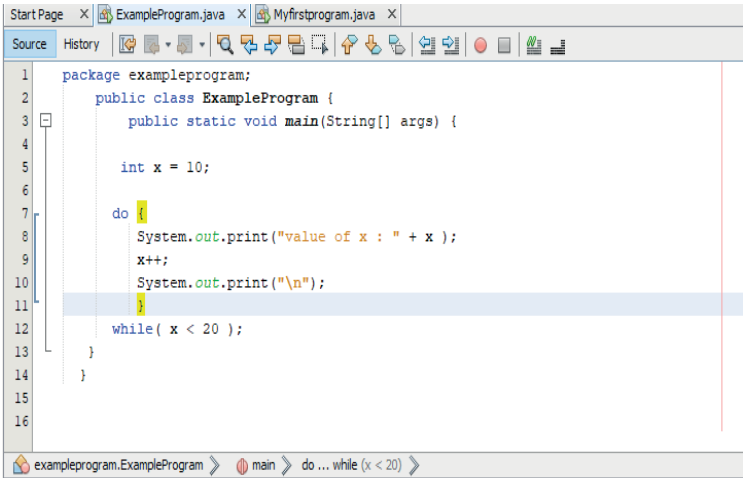
The syntax of a do...while loop is:

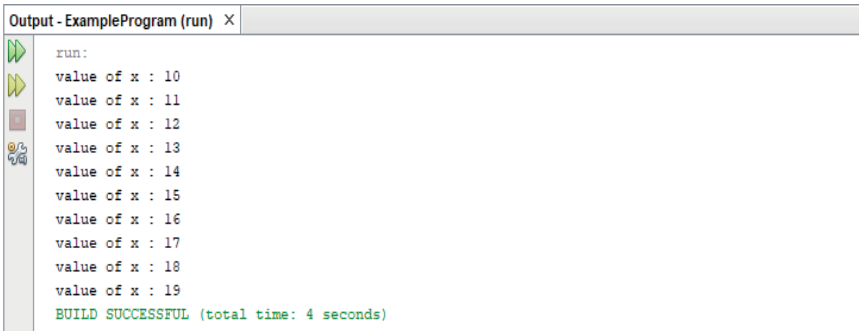
do {

// Statements

}while (Boolean_expression);

Example:





c.The for loop.

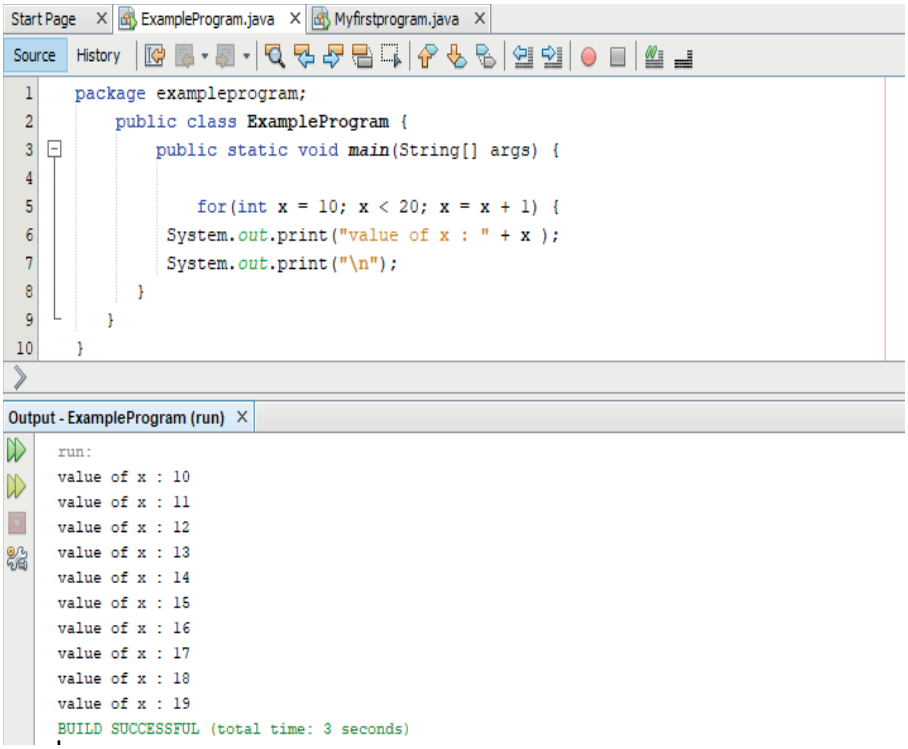
This is a repetition control structure that allows you to efficiently write a loop that needs to be executed a specific number of times.

A **for** loop is useful when you know how many times a task is to be repeated.

The syntax of a for loop is:

for(initialization; Boolean_expression; update) {
// Statements
}

Example:



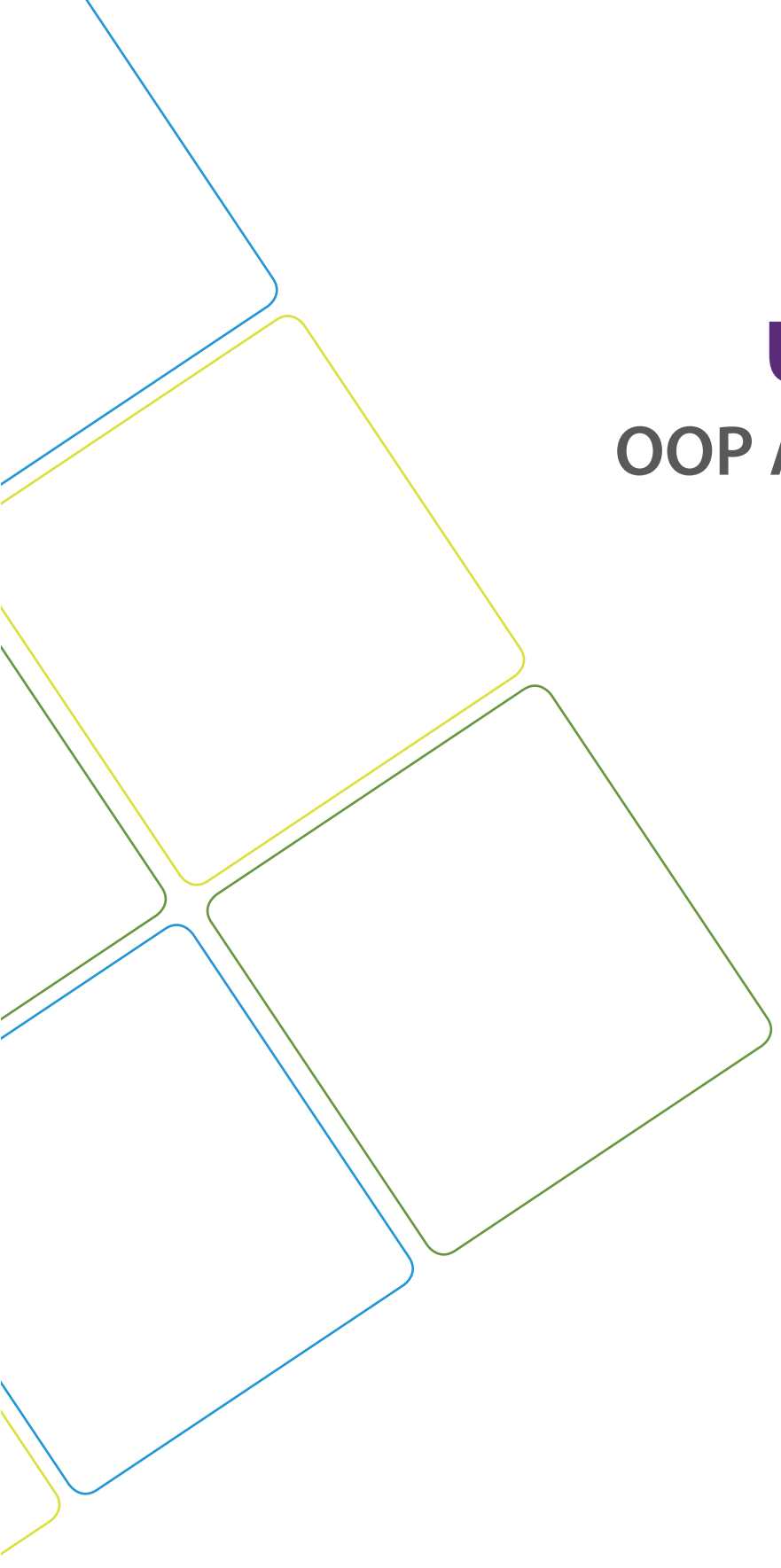
Application activity 10.7

1. State Java's selection statements and its behaviors
2. Give the syntax of selection statements already listed above and comparable their flowchart
3. Analyze the following programs and give the output will be produce
   a. // demonstrate an if-else-if ladder.

```
class Myfirstprogram {
public static void main(String args[]) {
int x; for(x=0; x<6; x++) {
if(x==1) System.out.println("x is one");
else if(x==2) System.out.println("x is two");
else if(x==3) System.out.println("x is three");
else if(x==4) System.out.println("x is four");
else
System.out.println("x is not between 1 and 4");
}}}
public class Example{
public static void main(String []args){
inta,b,c;
a = 2;
b = 3;
c = a - b;
if (c >= 0)
{System.out.println("c is a positive number");
}else
System.out.println("c is a negative number");
System.out.println();
c = b - a;
if (c >= 0)
{System.out.println("c is a positive number");
}else if (c < 0) System.out.println("c is a negative number");}}
```

END UNIT ASSESSMENT

1. In computer jargon, what's the difference between a statement and a comment?
2. Briefly what do class, object, methods, and instance variables mean?
3. Discuss about types of variables in java.
4. Write an assay for the Conditional statements in java
5. Starting with the hello world program, try out each of the following errors. After you make each change, compile the program, read the error message (if there is any), and then fix the error.
    i. Remove one of the open squiggly braces.
    ii. Remove one of the close squiggly braces.
    iii. Instead of main, write mian.
    iv. Remove the word static

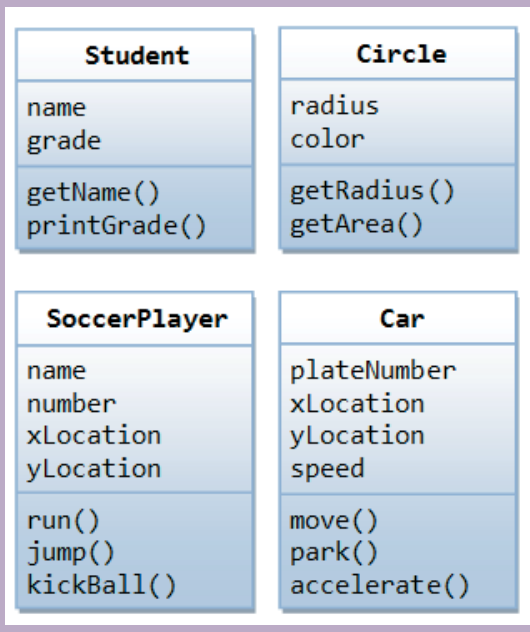


UNIT 11
OOP AND JAVA

UNIT 11: OOP AND JAVA

Key Unit Competency: To be able to use objects to manipulate data in Java program

Introductory Activity



Observe the above picture and answer the following questions.

1. Briefly, discuss the object oriented programming paradigm.
2. Find defined classes.
3. What are the data members for each class?
4. What are the member functions for each class?
5. What do you understand by instance of a class?
6. Create 2 instances of student class
7. Using Java, declare Student class in Java?

11.1. Classes vs Objects in Java

Learning Activity 11.1

```
class student {
int Age;
  public student(String name) {
  System.out.println("Student name is :" + name );
  }
public void setAge( int age ) {
Age = age;
  }
public intgetAge( ) {
    System.out.println("Student age is :" + Age );
Return Age;
  }
}
public class Learningactivity1 {
public static void main(String[] args) {
     student mystudent = new student( "Kiberinka " );
  mystudent.setAge( 18 );
    mystudent.getAge( );
    System.out.println("Variable Value :" + mystudent.Age );
  }
```

Write the above program in your computer and run it. After analyzing the result, answer the following questions:

1. Give the output of the program?
2. What are new keywords in the above program?
3. Describe the structure of the part starting with the word class.
4. List the data members contained in the class "student"
5. What are the functions found in the above program?
6. What is the role played by the keyword new in the above program?

11.1.1 Class in Java

a. Definition

A class is an entity that determines how an object will behave and what the object will contain. In other words, it is a blueprint or a set of instruction to build a specific type of object or a class is a group of objects which have common properties.

b. Syntax to declare a class:

```
class <class_name>
{
  field;
  method; }
```

c. General form of a class

A class is declared by the use of the class keyword. A simplified general form of a class definition is shown here:

```
class classname {
        access specifier type instance-variable1;
        access specifier type instance-variable2;
        ...
        type instance-variableN;
        type methodname1(parameter-list) {
        // body of method
        }
        type methodname2(parameter-list) {
        // body of method
        }
        // ...
        type methodnameN(parameter-list) {
        // body of method
        }
        }
```

11.1.2. Object in Java

An object is nothing but a self-contained component which consists of methods and properties to make a particular type of data useful. Object determines the behavior of the class. From a programming point of view, an object can be a data structure, a variable or a function. It has a memory location allocated. The object is designed as class hierarchies.

a. Syntax of an object

```
ClassName ReferenceVariable = new ClassName();
```

b. Reference variable

An object can be accessed only through a reference variable. A reference variable is declared to be of a specific type and that type can never be changed.

Reference variables can be declared as static variables, instance variables, method parameters, or local variables. A reference variable that is declared as final can't never be reassigned to refer to a different object. The data within the object can be modified, but the reference variable cannot be changed.

- **Object initialization through reference Variable:**

Initializing simply means storing data into object

Example 1: Below is the example where class student is created, *new* keyword and the reference variable are used

```
class Student{
 int id;
 String name; }
class TestStudent1{
 public static void main(String args[])
{  Student s1=new Student(); //creating an object
of Student   s1.id=101;
 s1.name="Uwineza";
                System.out.println(s1.id);          //
accessing member through reference variable
System.out.println(s1.name);                        //
accessing member through reference variable} }
```

Output:

101

Uwineza

Example: We can also create multiple objects and store information in it through reference variable

```
class Student{
 int id;
 String name;
}
class TestStudent3{
 public static void main(String args[]){
 //Creating objects
 Student s1=new Student();
 Student s2=new Student();
 Student s3=new Student();
 //Initializing objects
 s1.id=101;
 s1.name="Uwineza";
 s2.id=102;
 s2.name="Munezero";
 s3.id=103;
 s3.name="Rugwiro";
 //Printing data
        System.out.println(s1.id+"     "+s1.
name);   System.out.println(s2.id+" "+s2.
name);
 System.out.println(s3.id+" "+s3.name );
 }
}
```

Output

101 Uwineza

102 Munezero

103 Rugwiro

- **Object initialization through method in Java Language**

Two objects of Student class are going to be created and initialized with the value by invoking the insertRecord method. The state (data) of the objects are displayed by invoking the displayInformation() method.

Example: The use of method in Java program

```
class Student
{
 int rollno;
 String name;
 void insertRecord(int r, String n)
{
 rollno=r;
 name=n;
 }
 void displayInformation()
{
System.out.println(rollno+" "+name);
} }
```

```
class TestStudent4{
public static void main(String args[]){
 Student s1=new Student();
 Student s2=new Student();
 s1.insertRecord(111,"Muvara");
  s2.insertRecord(222,"Mugwaneza");   s1.
displayInformation();
 s2.displayInformation();
 } }
```

Output :

111 Muvara

222 Mugwaneza

Example: Example that maintains the records of Rectangle class.

```
class Rectangle
{
 int length;
 int width;
 void insert(int l, int w)
{
 length=l;
  width=w;
 }
 void calculateArea()
{
System.out.println(length*width);
}
}
class TestRectangle1{
 public static void main(String args[])
{
```

```
 Rectangle r1=new Rectangle();
 Rectangle r2=new Rectangle();
 r1.insert(11,5);
 r2.insert(3,15);
 r1.calculateArea();
 r2.calculateArea();
} }
```

Output

55

45

Application Activity 11.1

1. What do you understand by object instantiation?
2. What is the difference between a class and an object?
3. What is the role of a method in java program?
4. Discuss why main () method is public, static and void in java?
5. What is meant by the terms instance variable and instance method?

11. 2 Constructor in Java

Learning activity 11.2

```
class district
{
intnum;

String name;

    district()
  {
    System.out.println("Rwanda has 30 districts ");
  }
}
public class Learningactivity102 {
public static voidmain(String[] args) {
district district1 = new district();
System.out.println(district1.name);
System.out.println(district1.num);
  }
 }
```

Write the above program in your computer and run it. After analyzing the result, answer the following questions:

1. What is the output of the above program?
2. Discuss the difference between class district and the function district ()?
3. Why the function district () does not have a return type?

11.2.1. Understanding constructor in Java

a. Definition

A constructor is a special method of a class that initializes an object of that type. A constructor is an instance method that usually has the same name as the class and can be used to set the values of the members of an object, either to default or to user-defined values

A Constructor is a special type of method which is used to initialize the object. It is called constructor because it constructs the values at the time of object creation. It is not necessary to write a constructor for a class. It is because java compiler creates a default constructor if the class doesn't have any.

Every time an object is created using *new ()* keyword, at least one constructor is called. It is called a default constructor.

There are two (2) ways of creating a constructor: explicit and implicit:

- **Explicit** means that the constructor is created by the programmer.
- **Implicit** means that the creation of a constructor was done by the Java Virtual Machine or the tool, not the Programmer.

Java will provide default constructor implicitly even if the programmer doesn't write the code for constructor. It is called default constructor. The explicit is opposite to this implicit which means that the programmer has to write a constructor.

Example: The program below maintains records of employees.

```
class Employee{
   int id;  // instance variable
   String name;
   float salary;
Employee(int i, String n,float s) //
constructor
   {
id= i ;
name=n;
salary=s;
   } void insert(int i, String n, float s) {
            id= i ;
name=n;
salary=s;
   }
   void display()
{
System.out.
println(id+" "+name+" "+salary);
} }
public class TestEmployee {
public static void main(String[] args)
{
Employee e1=new Employee(
101,"Uwera",45000);
   Employee e2=new Employee(
102,"Kabera",25000);
   Employee e3=new Employee(
103,"Mugisha",55000);
   e1.display();
   e2.display();
   e3.display();
```

```
   e1.insert(201,"Umutesi",65000);
   e2.insert(202,"Kabayija",75000);
   e3.insert(203,"Mutara",85000);
   e1.display();
   e2.display();
   e3.display();
} }
```

Output

```
101 Uwera 45000.0
102 Kabera  25000.0
103 Mukiza 55000.0

201 Umutesi 65000.0
202 Kabayija 75000.0
203 Mutara 85000.0
```

There are two types of constructors in java namely default constructor (no-argument constructor) and parameterized constructor

11.2.2 Default Constructor

A default constructor is a constructor which doesn't have any parameter and which can be called with no arguments.

Syntax of default constructor:

<class_name>(){}

Example of default constructor

We are going to create the no-argument constructor in the Bike class. It will be invoked at the time of object creation.

```
class Bike1{
Bike1()// Default constructor
{
System.out.println("Bike is created");
}
public static void main(String args[]){
Bike1 b=new Bike1();
} }
```

Rule: If there is no constructor in a class, compiler automatically creates a default constructor.

Example: Default constructor that displays the default values

```
class Student3
{
int id;
String name;
void display()
{
System.out.println(id+" "+name);
}
public static void main(String args[])
{
Student3 s1=new Student3();
Student3 s2=new Student3();
s1.display();
s2.display(); }}
```

Output

```
0 null
0 null
```

Explanation: In the above class, there is no constructor created, so the compiler provides a default constructor. Here 0 and null values are provided by default constructor respectively as values for id and name.

11.2.3 Parameterized constructor in Java

A constructor which has a specific number of parameters is called parameterized constructor. A parameterized constructor is used to provide different values to the distinct objects.

Example : In this example, the constructor of Student class has two parameters of the number of parameters in the constructor can vary.

```
class Student4{
   int id;
   String name;
   Student4(int i, String n)//parameterized constructor
   {
  id = i;
   name = n;
   }
   void display()
{
System.out.println(id+" "+name);}
  }
public static void main(String args[])
{
        Student4 s1 = new Student4 (111,"Umubyeyi");
        Student4 s2 = new Student4 (222,"Shyaka");
   s1.display();
   s2.display();
  }
}
```

Output

```
111 Umubyeyi
222 Shyaka
```

11.2.4. Constructor Overloading in Java

In Java, a constructor is just like a method but without return type and it can also be overloaded like Java methods.

Constructor overloading in Java is a technique of having more than one constructor with different parameter lists. They are arranged in a way that each constructor performs a different task. They are differentiated by the compiler by the number of parameters in the list and their types.

Example: Constructor Overloading

```
class Student5
{
  int id;
  String name;
 int age;
  Student5(int i, String n)// constructor 1 has two parameters
  {
  id = i;
  name = n;
  }
  Student5(int i, String n, int a)// constructor  2 has three parameters
 {
  id = i;
  name = n;
  age=a;
  Student5 s1 = new Student5(111,"Umubyeyi");
  Student5 s2 = new Student5(222,"Shyaka",25);
  s1.display();
  s2.display();
  }
}
```

```
 }
  void display()
{
System.out.
println(id+" "+name+" "+age);
}
  }
  public static void main
(String args[]){
```

Output

```
111 Umubyeyi 0
222 Shyaka   25
```

11.2.5. Difference between constructor and method in java

There are many differences between constructors and methods. They are given in the following table.

Java Constructor	Java Method
Constructor is used to initialize the state of an object.	Method is used to expose behavior of an object.
Constructor must not have return type.	Method must have return type.
Constructor is invoked implicitly.	Method is invoked explicitly.
The java compiler provides a default constructor if you don't have any constructor.	Method is not provided by compiler in any case.
Constructor name must be same as the class name.	Method name may or may not be same as class name

Application Activity 11.2

1. Discuss the purpose of a constructor in a class?
2. Explain rules for creating a Java constructor?
3. What do understand by the Java Default Constructor?
4. Difference between constructor and method in Java?
5. Define a class Greetings, with single data member, greet that hold a string. The class should include two constructors, the default constructor which initializes the string greet to "Amakuru yawe ?" and parameterized constructor that takes a string as a parameter. Show these two constructors can be invoked using two objects in the main class.
6. Predict the output of the below Java program

```
class Student{
  int id;
  String name;
  Student(int i,String n){
  id = i;
  name = n;
  }
  Student(Student s){
  id = s.id;
  name =s.name;
  }
  void display(){System.out.println(id+" "+name);}
}

public class ApplicationActivity2 {

   public static void main(String[] args) {
    Student s1 = new Student(121,"Kwizera");
  Student s2 = new Student(s1);
  s1.display();
  s2.display();
  }
```

11.3. The use of "new" and "This" keywords in java

Learning activity 11.3

```
class Rectangle{
private int length;
private int breadth;
 Rectangle(int length){
this.length = length;
 }
 Rectangle(int length, int breadth){
this(length);
this.breadth = breadth;
 }
public intarea(){
return (length*breadth);
  }
}
public class Thsiexample {
public static voidmain(String[] args) {
        Rectangle rect = new Rectangle(5,5);

    System.out.println("The Area of rectangle is : "+
        + rect.area());
    }

}
```

Write the above program in the computer and run it, after analyzing the result, answer the following questions:

1. What is the output of the program?
2. What is the meaning of **this**.length = length;
3. What is the impact of **this**.length = length; to the output of the program
4. Discuss the role played by the statement Rectangle rect = **new** Rectangle (5,5); in the execution of the program ?

11.3.1 The use of "new" keyword

The **new** operator instantiates a class by allocating memory for a new object and returning a reference to that memory. The **new** operator also invokes the object constructor.
Instantiating a class means the same thing as "creating an object." When an object is created, an "instance" of a class is created.

Example:
*Student s1=***new** *Student ();*

The **new** operator requires a single, postfix argument: a call to a constructor. The name of the constructor provides the name of the class to instantiate. The **new** operator returns a reference to the object it created. This reference is usually assigned to a variable of the appropriate type. There are above many examples containing the **new** operator.

11.3.2 The use of "This" keywords in java

In java, this is a reference variable that refers to the current object.
This keyword is used in three (3) different circumstances:

- ***this*** can be used to refer current class instance variable.
- ***this*** can be used to invoke current class method (implicitly)
- ***this()*** can be used to invoke current class constructor.

1. *this:* to refer current class instance variable

The ***this*** keyword can be used to refer current class instance variable. If there is ambiguity between the instance variables and parameters, this keyword resolves the problem of ambiguity.

b. Understanding the problem without *this* keyword

Example: This keyword is not used

```
class Student{
int rollno;
String name;
float fee;
Student(int rollno, String name, float fee)
{
rollno=rollno;
name=name;
fee=fee;
}
void display()
{
System.out.println(rollno+" "+name+" "+fee +
"frw");}
}
class StudentThis
{
}
public static void main(String args[])
{
Student s1=new Student(111,"Umubyeyi",5000 );
Student s2=new Student(112,"Kalinda",6000 );
s1.display();
s2.display();
}
}
```

Output

```
111 Umubyeyi 5000 frw
222 Kalinda  6000 frw
```

Explanation: In the above example, parameters (formal arguments) and instance variables are same. So, **this** keyword needs to be used to distinguish local variable and instance variable.

Example: Solution of the above problem by **this** keyword

class Student *{* *int rollno;* *String name;* *float fee;* *Student(int rollno, String name, float fee)* *{* *this.rollno=rollno;* *this.name=name;* *this.fee=fee;* *}* *void display()* *{System.out.println(rollno +" "+ name +" "+ fee + "frw");}* *}* *class StudentThis {* *}* *public static void main(String args[])* *{* *Student s1=new Student(111,"Umubyeyi",5000);* *Student s2=new Student(112,"Kalinda",6000);* *s1.display();* *s2.display();* *}* *}*	**Output** 111 Umubyeyi 5000 frw 222 Kalinda 6000 frw

Notice that if local variables (formal arguments) and instance variables are different, there is no need to use this keyword.

Example: Program where this keyword is not required

```
 class Student
{
int rollno;
String name;
float fee;
Student(int r, String n, float f)
{
rollno=r;
name=n;
fee=f;
}
void display()
{System.out.println(rollno+" "+name+" "+fee +
"frw");
}
}
 class TestThis3
{
public static void main(String args[])
{
Student s1=new Student(111,"Umubyeyi",5000);
Student s2=new Student(112,"Kalinda",6000);
s1.display();
s2.display();
}
}
```

Output

```
111 Umubyeyi 5000 frw
222 Kalinda  6000 frw
```

It is a better approach to use meaningful names for variables. So we use same name for instance variables and parameters in real time, and always use this keyword.

1. this: **to invoke current class method**

The method of the current class can be invoked by using the keyword. If **this** keyword is not used, the compiler automatically adds this keyword while invoking the method.

Example: Program where ***this*** keyword is not required

class A *{* *void m()* *{* *System.out.println("Amakuru Yanyu");* *}* *void n()* *{* *System.out.println("Muraho Neza ");* *this.m();* *}* *}* *class TestThis4* *{* *public static void main(String args[])* *{* *A a=new A();* *a.n();* *}* *}*	Output Amakuru Yanyu Muraho Neza

3. *this()* : to invoke current class constructor

The ***this()*** constructor call can be used to invoke the current class constructor. It is used to reuse the constructor. In other words, it is used for constructor chaining.

Example 12: Calling default constructor from parameterized constructor

class A *{* *A()* *{* *System.out.println("Amakuru yanyu");* *}* *A(int x)* *{* *this();* *System.out.println(x);* *}* *}* *class TestThis5{* *public static void main(String args[])* *{* *A a=new A(10);* *}* *}*	**Output** Amakuru yanyu 10

Application Activity 11.3

1. Suppose that **animal** is the name of a class and that gorillas is a variable of type animal.
   a. What is the meaning of the statement "gorillas = new animal();"
   b. What does the computer do when it executes this statement?
2. Write a Java program to explain the use of this keyword?
3. Predict the output of the below Java program:

```
class Account{
 String name;
 int balance;
 Account(String name, int balance){
  this.name = name;
  this.balance = balance;
 }
 void printInfo(){
  System.out.println("Name: " + name + " Balance: " + balance);
 }}
public class Account1 {
     public static void main(String[] args) {
    Account ac = new Account("Murwanashyaka", 5000);
  ac. printInfo();
  }
 }
```

11.4. Arrays in Java

Learning Activity 11.4

```
{
public static void main(String[] args) {
double[] myList = {19, 29, 34, 35};
for (int i = 0; i < myList.length; i++) {
    System.out.println(myList[i] + " ");
  }
double total = 0;
for (int i= 0; i < myList.length; i++) {
    total += myList[i];
  }
  System.out.println("Total is " + total);
double max = myList[0];
for (int i = 1; i < myList.length; i++) {
 if(myList[i] > max) max = myList[i];
  }
  System.out.println("Max is " + max);
 }
}
```

Run the above program and answer followings questions:

1. What is the output of the above program
2. Discuss the meaning of the following statements:
   i. double[] myList = {19, 29, 34, 35};
   ii. for (int i = 0; i < myList.length; i++)
   iii. System.out.print(myList[i] + " ");

11.4.1 Understanding Array in Java

An array is a container that holds data (values) of the same data type.

Array is a fundamental construct in Java that allows you to store and access large number of values conveniently.

a. Array syntax in Java

dataType[] arrayName;

- **dataType** can be a primitive data such as : int, char, Double, byte .
- **arrayName** is an identifier.

Example:
int[] age;
age = new int[5];
age array can hold 5 values of type int.
Note: You can replace two statements above with a single statement.
int[] age = new int[5];

b. Initialize arrays in Java

In Java, arrays can be initialized during declaration or it can be initialized (or change values) later in the program.

Example:
int[] age = {12, 4, 5, 2, 5};

This statement creates an array and initializes it during declaration. The length of the array is determined by the number of values provided which is separated by commas. In our example, the length of age array is 5.

Example : array initialization

```
class ArrayExample {
public static void main(String[] args) {
    int[] age = {12, 4, 5, 2, 5};
for (int i = 0; i < 5; ++i) {
System.out.println("Element at index " + i +": "
+ age[i]);
    }
  }
}
```

Output

```
Element at index 0: 12
Element at index 1: 4
Element at index 2: 5
Element at index 3: 2
Element at index 4: 5
```

Example: program that computes sum and average of arrays elements in Java

class SumAverage { *public static voidmain(String[] args) {* *int[] numbers = {2, -9, 0, 5, 12, -25, 22, 9, 8, 12};* *int sum = 0;* *Double average;* *for (intnumber: numbers) {* *sum += number;* *}int arrayLength = numbers.length; // Change sum and arrayLength to double as average is in double* *average = ((double)sum / (double) arrayLength);* *System.out.println("Sum = " + sum);* *System.out.println("Average = " + average);* *}* *}*	**Output** Sum = 36 Average = 3.6

11.4.2 Array of object in Java

An object of class represents a single record in memory, if we want more than one record of class type, we have to create an array of class or object. As we know, an array is a collection of similar type, therefore an array can be a collection of class type.

a. Syntax for creating an array of object

```
class ClassName
{
        datatype var1;
        datatype var2;
        - - - - - - - - - -
        datatype varN;
        method1();
        method2();
        - - - - - - - - - -
        methodN();
}
```

Syntax of array of object

Class-Name [] object = new Class-Name [size];

Example: creating an array of object

```
    import java.util.Scanner; class Employee
{ intId;
String Name;
intAge; I
ongSalary;
void GetData()        // Defining GetData()
  {
Scanner sc = new
Scanner(System.in);
 System.out.print("\n\tEnter
Employee Id : "); I
d = Integer.parseInt(sc.nextLine());
System.out.print("\n\tEnter
Employee Name : ");
Name = sc.nextLine();
System.out.print("\n\tEnter
 Employee Age : ");
Age = Integer.parseInt(sc.nextLine());
System.out.print("\n\tEnter
Employee Salary : ");
Salary = Integer.parseInt(sc.nextLine());
    }
void PutData()
// Defining PutData()
  {
System.out.print("\n\t" + Id + "\t" +Name + "\t"
+Age + "\t" +Salary);
 }
public classArrayofobject
{ public staticvoid main(String args[])
 {
Employee[] Emp = new Employee[3];  inti; for(i=0;i
<3;i++)
   Emp[i] = new Employee();
  // Allocating memory to each object
for(i=0;i<3;i++)
 {
 System.out.print("\nEnter details of
"+ (i+1) +" Employee\n");
 Emp[i].GetData();
 }
 System.out.print("\nDetails of
Employees\n");
for(i=0;i<3;i++)
 Emp[i].PutData();
 }}
```

Output

```
Enter details of 1 Employee
 Enter Employee Id : 123
 Enter Employee Name : Rul-
inda
 Enter Employee Age : 23


  Enter Employee Salary :
150000
Enter details of 2 Employee
 Enter Employee Id : 124
  Enter Employee Name :
Mwiza
 Enter Employee Age : 21
  Enter Employee Salary :
125000
Enter details of 3 Employee
 Enter Employee Id : 125
  Enter Employee Name :
Kanyana
 Enter Employee Age : 24
  Enter Employee Salary :
124000
Details of Employees
  123 Rulinda 23 150000
124 Mwiza 21 125000 125
Kanyana 24 124000
```

Application activity 11.4.

1. What is the output of this program?

```
class evaluate
 {
   public static void main(String args[])
     {
           int arr[] = new int[] {0 , 1, 2, 3, 4, 5, 6, 7, 8, 9};
           int n = 6;
      n = arr[arr[n] / 2];
           System.out.println(arr[n] / 2);
     }
 }
```

2. What is the output of this program?

```
class array_output
 {
   public static void main(String args[])
   {
    int array_variable [] = new int[10];
        for (int i = 0; i < 10; ++i)
    {
      array_variable[i] = i;
      System.out.print(array_variable[i] + " ");
      i++;
    }}}
```

3. Write a Java program to sort a numeric array and a string array
4. Write a Java program to find the index of an array element

11.5. Inheritance in Java

Learning activity 11.5

```
class Teacher {
  String designation = "Teacher";
  String schoolName = "Lycee de Kigali";
 Void does(){
  System.out.println("Teaching");
  }
}
public class computerTeacher extends Teacher{
  String mainSubject = "Computer Science";
public static void main(String args[]){
        computerTeacher obj = newcomputerTeacher();

System.out.println(obj.schoolName);
        System.out.println(obj.designation);
        System.out.println(obj.mainSubject);
        obj.does();
  }}
```

Write the above program in the computer and run it and after analyzing the result, answer the following questions:

1. What is the output of the above program
2. Discuss the role played by the below statement in the program.**public class** computerTeacher **extends** Teacher{ String mainSubject = "Computer Science";

10.5.1 Understanding inheritance in java

a. Definition

Inheritance in java is a mechanism in which one object acquires all the properties and behaviors of parent object. It is an important part of Object Oriented programming system. The idea behind inheritance in java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of parent class, and you can add new methods and fields also. Inheritance is used for two reasons: method overriding and code reusability.

Note: Inheritance represents the IS-A relationship, also known as parent-child relationship.

b. Terms used in Inheritance

Below are some of the terms used in inheritance:

i. **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.

ii. **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.

iii. **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.

iv. **Reusability:** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in previous class.

c. Syntax of Java Inheritance

```
class Subclass-name extends Superclass-name
{
  //methods and fields
}
```

11.5.2 The use of *"extends "*keyword in Java

The extends is a Java keyword used in inheritance and it indicates that you are making a new class that derives from an existing class.

Note: In the terminology of Java, a class which is inherited is called parent or super class and the new class is called child or subclass.

Example: Inheritance in Java

Programmer is the subclass and Employee is the superclass. Relationship between two classes is Programmer IS-A Employee. It means that Programmer is a type of Employee.

```
lass Employee{
 float salary=40000;
}
class Programmer extends Employee{
 int bonus=10000;
 public static void main(String args[]){
   Programmer p=new Programmer();
  System.out.println("Programmer salary is:"+p.salary);
     System.out.println("Bonus of Programmer is:"+p.
bonus);
} }
```

Output

```
101 Uwera 45000.0
102 Kabera 25000.0
103 Mukiza 55000.0
```

Note: In the above example, Programmer object can access the field of own class as well as of Employee class i.e. code reusability.

11.5.3 Types of inheritance in Java

On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.

Note: Multiple inheritance is not supported in java through class.

**a. Single Inheritance **

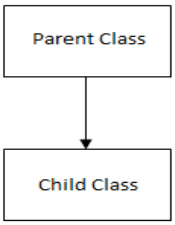


Figure 1: single inheritance illustration

Single inheritance enables a derived class to inherit properties and behavior from a single parent class. It allows a derived class to inherit the properties and behavior of a base class, thus enabling code reusability as well as adding new features to the existing code.

Example: Single Inheritance

class Animal{ *void eat(){System.out.println("eating...");}* *}* *class Dog extends Animal{* *void bark(){System.out.println("barking...");}* *}* *public static void main(String args[]){* *Dog d=new Dog();* *d.bark();* *d.eat();* *}}*	**Output** barking… eating…

b. Multilevel Inheritance

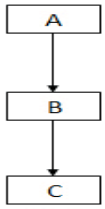


Figure 2: Multilevel Inheritance Illustration

Multiple Inheritance refers to the concept of one class extending (or inherits) more than one base class. The inheritance we learnt earlier had the concept of one base

class or parent. The problem with "multiple inheritance" is that the derived class will have to manage the dependency on two base classes.

Example : Multilevel Inheritance

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class BabyDog extends Dog{
void weep(){System.out.println("weeping...");}
}
public static void main(String args[]){
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat();
}
}
```

Output

weeping...
barking...
eating...

c. Hierarchical inheritance in java

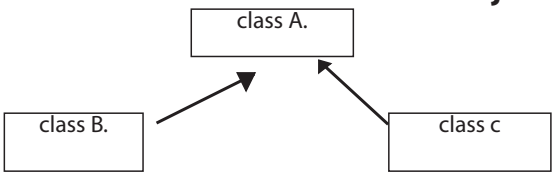


Figure 3: Hierarchichal Inheritance

Consider the above figure, there are three classes A, B and C . Class C and B inherits class A.

Example : Hierarchical inheritance

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class Cat extends Animal{
void meow(){System.out.println("meowing...");}
}
class TestInheritance3{
public static void main(String args[]){
Cat c=new Cat();
c.meow();
c.eat();
//c.bark();//C.T.Error
}}
```

Output

meowing...
eating...

Note: Java doesn't support Multiple Inheritance through classes. Multiple Inheritance is a feature of object oriented concept, where a class can inherit properties of more than one parent class. The problem occurs when there exist methods with same signature in both the super classes and subclass. On calling the method, the compiler cannot determine which class method to be called and even on calling which class method gets the priority.

Application Activity 11.5

3. Explain the types of inheritance in Java?
4. Discuss the purpose of use of inheritance in java?
5. Give the general form of declaring inheritance in Java?
6. Discuss IS-A relationship in inheritance?
7. Can a class extend itself? Explain your answer?
8. Write a java program to the demonstrate the concept of multi-level inheritance in Java
9. Predict the output of the below program

```
class Shape {
public void display() {
System.out.println("Inside display");
}
}
class Rectangle extends Shape {
public void area() {
System.out.println("Inside area");
}
}
class Cube extends Rectangle {
public void volume() {
System.out.println("Inside volume");
}
}
public class Cartest {
public static void main(String[] args) {
    Cube cube = new Cube();
   cube.display();
   cube.area();
   cube.volume();
  }
 }
```

11.6 Access Modifiers in java

Learning activity 11.6

```
class Teacher {
private String designation = "Teacher";
private String schoolName = "Lycee de Kigali";
public String getDesignation() {
        returndesignation;
  }
protected voidsetDesignation(String designation) {
        this.designation = designation;
  }
protected String getschoolName() {
        returnschoolName;
  }
protected voidsetschoolName(String schoolName) {
        this.schoolName = schoolName;
  }
 voiddoes(){
        System.out.println("Teaching");
  }
}
public classJavaExample extends Teacher{
  String mainSubject = "Computer Science";
public static voidmain(String args[]){
        JavaExample obj = new JavaExample();
        System.out.println(obj.getschoolName());
        System.out.println(obj.getDesignation());
        System.out.println(obj.mainSubject);
        obj.does();
  }
}
```

Run the above program and answer followings questions:

1. What is the output of the program
2. Discus the below keywords used in the program:
   i. Public
   ii. Extends
   iii. Protected
   iv. Private

Understand Access modifiers in Java

Access modifiers are keywords used to specify the accessibility of a class (or type) and its members. These modifiers can be used from code inside or outside the current application. The purpose of using access modifiers is to implement encapsulation. The following are benefits of using access modifiers:

i. Prevention of access to the internal data set by users to invalid state.
ii. Provision for changes to internal implementation of the types without affecting the components using it.
iii. Reduction in complexity of the system by reducing the interdependencies between software components

There are 4 types of java access modifiers: private, default, protected, public

a. Default access modifier

When no access modifier is specified for a class, method or data member, then it is said to be having the default access modifier by default. The data members, class or methods which are not declared using any access modifiers i.e. having default access modifier are accessible only within the same package.

Example: Default access modifier

We are going to create two packages pack and mypack and access the A class from outside its package. Since A class is not public, so it cannot be accessed from outside the package.

```
//save by A.java
package pack;
class A{
  void msg(){System.out.println("Hello");}
}
//save by B.java
package mypack;
import pack.*;
class B{
  public static void main(String args[]){
   A obj = new A();//Compile Time Error
   obj.msg();//Compile Time Error
  }
}
```

Output

Running Error

Note: The scope of class A and its method msg() is default so it cannot be accessed from outside the package.

b. Private access modifier

The private access modifier is specified using the keyword private.

- The methods or data members declared as private are accessible only within the class in which they are declared.
- Any other class of same package will not be able to access these members.
- Classes or interface cannot be declared as private

Example: private access modifier

In this example, we will create two classes A and B within same package p1. We will declare a method in class A as private and try to access this method from class B and see the result.

```
Package p1;
classA
{
  Private void display()
   {
     System.out.println(«Rwanda Nziza»);
   }
}
classB
{
  Public static void main(String args[])
    {
         A obj = newA();//trying to access private
method of another class
       obj.display();
    }}
```

Output

Running Error

Note: Role of Private Constructor

The role of a private constructor is that if you make any class constructor private, you cannot create the instance of that class from outside the class.

Example: private used on a constructor

```
class A{
private A(){}//private constructor
void msg(){System.out.println("Hello java");}
}
public class Simple{
 public static void main(String args[]){
  A obj=new A();//Compile Time Error
 } }
```

Output

Running Error

c. Protected access modifier

The protected access modifier is accessible within package and outside the package but through inheritance only. The protected access modifier can be applied on the data member, method and constructor. It can›t be applied on the class.

Example: protected access modifier

We have created two packages pack and mypack. The A class of pack package is public, so can be accessed from outside the package. But msg () method of this package is declared as protected, so it can be accessed from outside the class only through inheritance.

```
//save by A.java
package pack;
public class A{
protected void msg(){System.out.
println("Hello");}
}
//save by B.java
package mypack;
import pack.*;
 class B extends A{
  public static void main(String args[]){
   B obj = new B();
   obj.msg();
  }
}
```

Output

Running Error

d. Public access modifier

The public access modifier is specified using the keyword public.

- The public access modifier has the widest scope among all other access modifiers.
- Classes, methods or data members which are declared as public are accessible from everywhere in the program. There is no restriction on the scope of a public data member.

Example: public access modifier

```
//save by A.java
package pack;
public class A{
public void msg(){System.out.println("Hello");}
}
//save by B.java
 package mypack;
import pack.*;
class B{
 public static void main(String args[]){
A obj = new A();
 obj.msg();
 }
}
```

Output

Running Error

Application Activity 11.6

1. What is the difference between access specifiers and access modifiers in java?
2. What access modifiers can be used for class?
3. Explain what access modifiers can be used for methods?
4. Explain what access modifiers can be used for variables?
5. Write a Java program that show how "Protected" visibility is used in Java.

11.7 Overloading and overriding Method in Java

Learning activity 11.7

```
class Human{
public voideat()
  {
    System.out.println("Human is eating");
  }
}
class Boy extends Human{
    public voideat(){
    System.out.println("Boy is eating");
  }
```

```
public staticvoid main( String args[]) {
    Boy obj = new Boy();
        obj.eat();
  }
}
```

Write the above program in the computer and run it and after analyzing the result, answer the following questions:

a. What is the output of the above program
b. Discuss how Java implement the below methods

```
public voideat()
  {
    System.out.println("Human is eating");
  }
```

And

```
public voideat(){
 System.out.println("Boy is eating");
```

11.7.1 Method Overloading

a. Understanding method overloading

Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different.

Three ways to overload a method

b. Number of parameters

Example:

add(int, int)
add(int, int, int)

This is a valid case of overloading

c. Data type of parameters

Example:

add(int, int)
add(int, float)

d. Sequence of Data type of parameters

Example:

add(int, float)
add(float, int)

Note:

Invalid case of method overloading
int add(int, int)
float add(int, int)

If two methods have same name, same parameters and have different return type, then this is not a valid method overloading example. This will throw compilation error.

11.7.2 Method Overriding in Java

a. Definition Method Overriding in Java

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in java.

In other words, if subclass provides the specific implementation of the method that has been provided by one of its parent class, it is known as method overriding.

b. Rules for Method Overriding

- The argument list should be exactly the same as that of the overridden method.
- The return type should be the same or a subtype of the return type declared in the original overridden method in the superclass.
- The access level cannot be more restrictive than the overridden method's access level. For example: If the superclass method is declared public then the overriding method in the sub class cannot be either private or protected.
- Instance methods can be overridden only if they are inherited by the subclass.
- A method declared final cannot be overridden.
- A method declared static cannot be overridden but can be re-declared.
- If a method cannot be inherited, then it cannot be overridden.
- A subclass within the same package as the instance's superclass can override any superclass method that is not declared private or final.
- A subclass in a different package can only override the non-final methods declared public or protected.
- Constructors cannot be overridden.

c. Usage of Java Method Overriding

Method overriding is used to provide specific implementation of a method that is already provided by its super class. If a class inherits a method from its superclass, then there is a chance to override the method provided that it is not marked final. The benefit of overriding

is the ability to define a behavior that's specific to the subclass type, which means a subclass can implement a parent class method based on its requirement.

In object-oriented terms, overriding means to override the functionality of an existing method.

Example: Method Overriding

```
class Animal {
public void move() {
            System.out.println("Animals  can
move");
  }
}
class Dog extends Animal {
public void move() {
   System.out.println("Dogs can walk and
run");
  }
}
public class Animaltest {
public static void main(String args[]) {
     Animal a = new Animal();   // Animal
reference and object
      Animal b = new Dog();    // Animal
reference but Dog object
   a.move();  // runs the method in Animal
class
     b.move();   // runs the method in Dog
class
  }}
```

Output

```
Animals can move
Dogs can walk and run
```

Explanation

In the above example, you can see that even though **b** is a type of Animal it runs the move method in the Dog class. The reason for this is: In compile time, the check is made on the reference type. However, in the runtime, Java Virtual Machine figures out the object type and would run the method that belongs to that particular object. Therefore, in the above example, the program will compile properly since Animal class has the method move. Then, at the runtime, it runs the method specific for that object.

11.7.3 The use the "super" keyword in Java

When invoking a superclass version of an overridden method the **super** keyword is used.

Example: use of super keyword

```
class Animal {
public void move() {
        System.out.println("Animals can move");
  }
}
class Dog extends Animal {
   public void move() {super.move(); // invokes the super class method
   System.out.println("Dogs can walk and run");
  }
}
public class Animaltest {
public static void main(String args[]) {Animal b = new Dog(); // Animal reference but Dog object
     b.move(); // runs the method in Dog class
  }
```

Output

```
Animals can move
Dogs can walk and run
```

11.7.4 The use of "Final" Keyword in Java

Final keyword in Java has three different uses: final variable, prevent inheritance and prevent methods from being overridden.

a. Final variable

Final variables are nothing but constants. We cannot change the value of a final variable once it is initialized.

b. Using final to define constants

If you want to make a local variable, class variable (static field), or instance variable (non-static filed) constant, declare it final. A final variable may only be assigned to once and its value will not change and can help avoid programming errors.

Once a final variable has been assigned, it always contains the same value. If a final variable holds a reference to an object, then the state of the object may be changed by operations on the object, but the variable will always refer to the same object.

Example: Use final to define constant

```
class Demo{
 finalint MAX_VALUE=99;
void myMethod(){
MAX_VALUE=101;
  } }
public static voidmain(String args[]){
 Demo obj=new  Demo();
  obj.myMethod();
 } }
```

Output

```
Running Error
```

There is a compilation error in the above program because of the change the value of a final variable "MAX_VALUE".

Note: *It is considered as a good practice to have constant names in UPPER CASE (CAPS)*

c. Blank final variable

A final variable that is not initialized at the time of declaration is known as blank final variable. Let initialize the blank final variable in constructor of the class otherwise it will throw a compilation error (Error: variable MAX_VALUE might not have been initialized).

Example: blank final variable is used in a class.

```
class Demo{
final intMAX_VALUE;  //Blank final variable
  Demo(){ //It must be initialized in constructor
MAX_VALUE=100;
  }
void myMethod(){
    System.out.println(MAX_VALUE);
    }}
```

```
public classFinaltest1 {
public static voidmain(String[] args)
{
      Demo obj=new  Demo();
   obj.myMethod();
  }
}
```

Output

```
100
```

d. Final method

When a class is extended by other classes, its methods can be overridden for reuse. There may be circumstances when you want to prevent a particular method from being overridden, in that case, declare that method final. Methods declared as final cannot be overridden which means even though a sub class can call the final method of parent class without any issues but it cannot override it.

Example

```
class XYZ{
final voiddemo(){
   System.out.println("XYZ Class Method");
  }
}
class ABC extends XYZ{}
public class Finaltest2 {
public static voidmain(String[] args) {
ABC obj= new ABC();
obj.demo();
  }
 }
```

Output

Class XYZ Class Method

This above program runs fine because the program is not overriding the final method.

Example

```
class XYZ{
final voiddemo(){
    System.out.println("XYZ Class Method");
  }
}
class ABC extends XYZ{
void demo(){
    System.out.println("ABC Class Method");
  }
}
```

```
public classFinaltest3 {
public static voidmain(String[] args) {
     ABC obj= new ABC();
    obj.demo();
}}
```

Output

Running Error

The above program give a compilation error because the program is overriding the final method

e. Final class

If you find a class's definition is complete and you don't want it to be sub-classed, declare it final. A final class cannot be inherited, therefore, it will be a compile-time error if the name of a final class appears in the extends clause of another class declaration; this implies that a final class cannot have any subclasses.

It is a compile-time error if a class is declared both final and abstract, because the implementation of such a class could never be completed.

Because a final class never has any subclasses, the methods of a final class are never overridden

Example: final class

```
final class XYZ{
}
class ABC extends XYZ{
  void demo(){
    System.out.println("My Method");
  }
public classFinaltest4 {
public static void main(String[] args) {
   ABC obj= new ABC();
   obj.demo();
  }
  }
```

Output

None

Note: *The type ABC cannot subclass the final class XYZ*

Application activity 11.7

1. Write a Java program that demonstrates the concept of a method overloading in Java?
2. Write a Java program that demonstrates the concept of a method overriding in Java?
3. Discuss the difference between a method overloading and a method overriding in Java?
4. Predict the output of the below Java program

```
public class CrunchifyObjectOverriding {
        public static void main(String args[]) {
                Company a = new Company();
                Company b = new coffee();
                a.address();// runs the method in Company class
                b.address();// Runs the method in eBay class
        }
}
class Company {
        public void address() {
                System.out.println("This is Address of Rwanda Tea Company...");
        }
}
class coffee extends Company {
        public void address() {
                super.address();
                System.out.println("This is Rwanda Coffee Company 's Address...");
        }
}
```

11.8 Encapsulation in Java

Learning Activity 11.8

```
class Encapsulate
{
private String Name;
private int addres;
 private int Age;
public intgetAge()
  {
returnAge;
  }
public String getName()
  {
returnName;
  }
public String getName()
  {
returnName;
  }
public intgetaddres()
  {
returnaddres;
  }
public voidsetAge( int newAge)
  {
Age = newAge;
  }
public void setName(String newName)
  {
Name = newName;
  }
public voidsetAddres( int newAddres)
  {
addres = newAddres;
 }}
public classTestEncapsulation
{
public static voidmain (String[] args)
  {
   Encapsulate obj = new Encapsulate();
    obj.setName("Masabo");
    obj.setAge(19);
    obj.setAddres(51);
     System.out.println("Name: " + obj.
getName());
      System.out.println(" Age: " + obj.
getAge());
    System.out.println(" Registration : "
+ obj.getaddres());
  }
}
```

Answer the following questions questions:

1. What is the output of the above program?
2. Discuss the role played by the bellow code:
   i. public String getName()
   ii. public void setName(String newName)

11.8.1 Understanding encapsulation in Java

a. Definition

Encapsulation simply means binding object state (fields) and behavior (methods) together. While creating class, you are doing encapsulation.

The whole idea behind encapsulation is to hide the implementation details from users. If a

data member is private it means it can only be accessed within the same class. No outside class can access private data member (variable) of other class.

Steps to implement encapsulation in Java:

- Make the instance variables private so that they cannot be accessed directly from outside the class. You can only set and get values of these variables through the methods of the class.
- Have getter and setter methods in the class to set and get the values of the fields.

Example: Encapsulation in Java

```
class Employee{
private int ssn;
private String empName;
private int empAge;
  //Getter and Setter methods
  public int getEmpSSN(){
returnssn;
  }
public String getEmpName(){
returnempName;
  }
public int getEmpAge(){
returnempAge;
  }
public void setEmpAge(int newValue){
empAge = newValue;
 }public void setEmpName(String newValue){
empName = newValue;
  }
public void setEmpSSN(int newValue){
ssn = newValue;
  }}
```

```
public class Employeetest{
 ;
public static void main(String args[]){
      Employee obj = new Employee()
      obj.setEmpName("Rukundo");
      obj.setEmpAge(32);
      obj.setEmpSSN(233);
       System.out.println("Employee Name: " +
obj.getEmpName());
    System.out.println("Employee SSN: " + obj.
getEmpSSN());
    System.out.println("Employee Age: " + obj.
getEmpAge());
  }
}
```

Output

```
Employee Name: Rukundo
Employee SSN: 223
Employee Age: 32
```

Explanation:

In above example all the three data members (or data fields) are private which cannot be accessed directly. These fields can be accessed via public methods only. Fields empName, ssn and empAge are made hidden data fields using encapsulation.

The below are Advantages of encapsulation

- It improves maintainability and flexibility and re-usability
- The fields can be made read-only or write-only

- User would not be knowing what is going on behind the scene. They would only be knowing that to update a field call set method and to read a field call get method but what these set and get methods are doing is purely hidden from them.

11.8.2 Understand use of getter and setter method in Java

In Java, getter and setter method are two conventional methods that are used for retrieving and updating value of a variable.

Example: simple class with a private variable and a couple of getter and setter methods

```
public class Add {
  private int number;
  public int getNumber() {
    return this.number;
  }
  public void setNumber(int num) {
    this.number = num;
  }
}
```

In above example, Add class declares a private variable, number. Since number is private, code from outside this class cannot access the variable directly like this:

```
Add obj = new Add ();
obj.number = 10;    // compile error, since number is private
int num = obj.number; // same as above
```

Instead, the outside code have to invoke the getter, getNumber() and the setter, setNumber() in order to read or update the variable.

Example:

```
Add obj = new Add ();
obj.setNumber(10); // OK
int num = obj.getNumber();  // fine
```

Note: a setter is a method that updates value of a variable. And a getter is a method that reads value of a variable. Getter and setter methods are also known as accessor and mutator in Java.

By using getter and setter, the programmer can control how his important variables are accessed and updated in a correct manner, such as changing value of a variable within a specified range.

Consider the following code of a setter method:

```
public void Add (int num) {
  if (num < 10 || num > 100) {
    throw new IllegalArgumentException();
  }
  this.number = num;
}
```

That ensures the value of number is always set between 10 and 100. Suppose the variable number can be updated directly, the caller can set any arbitrary value to it:

obj.number = 3;

And that violates the constraint for values ranging from 10 to 100 for that variable. Of course we don't expect that happens. Thus hiding the variable number as private and using a setter comes to rescue. On the other hand, a getter method is the only way for the outside world reads the variable's value:

```
public int getNumber() {
  return this.number;
}
```

Explanation:

So far, setter and getter methods protect a variable's value from unexpected changes by outside world the caller code. When a variable is hidden by private modifier and can be accessed only through getter and setter, it is encapsulated. Encapsulation is one of the fundamental principles in object-oriented programming (OOP), thus implementing getter and setter is one of ways to enforce encapsulation in program's code.

Application activity 11.8

1. What do you understand by Encapsulation in Java?
2. What is the primary benefit of encapsulation in Object Oriented?
3. Discuss s the difference between encapsulation and abstraction?
4. What are the features of encapsulation
5. What are the advantages of using encapsulation in Java and Object Oriented Programming?

11.9 Abstract class in Java

Learning activity 11.9

```
abstract class Shape
{
   String color;
abstract doublearea();
   public abstract StringtoString();
public Shape(String color) {
  System.out.println("Shape constructor called");
this.color = color;
   }Public String getColor() {
Return color;
   }
}
class Circle extends Shape
{
double radius;
public Circle(String color,double radius) {
super(color);
      System.out.println("Circle constructor called");
this.radius = radius;
   }
double area() {
return Math.PI * Math.pow(radius, 2);
   }
public StringtoString() {
return "Circle color is " + super.color + "and area is : " + area();
   }
}
class Rectangle extends Shape{
double length;
double width;
public Rectangle(String color,double length,double width) {
super(color);
      System.out.println("Rectangle constructor called");
this.length = length;
this.width = width;
   }
double area() {
return length*width;
   }
```

```
public StringtoString() {
 return"Rectangle color is " + super.color + "and area is : " + area();
   }
}
public class Test
{
public static voidmain(String[] args)
   {
      Shape s1 = new Circle("Red", 2.2);
      Shape s2 = new Rectangle("Yellow", 2, 4);

      System.out.println(s1.toString());
      System.out.println(s2.toString());
   }
}
```

Answer the following questions:

1. What is the output of the above program
2. What is the role play the Keyword Abstract
3. Discuss how the class shape is implemented

11.9.1 Understanding Abstraction in Java

Abstraction is a process of hiding the implementation details and showing only functionality to the user. In other words, it shows only important things to the user and hides the internal details.

Example: sending sms, you just type the text and send the message. You don't know the internal processing about the message delivery.

Note: *Abstraction focuses on what the object does instead of how it does it.*

a. Abstract class in Java

A class that is declared as abstract is known as abstract class. A class that is declared with abstract keyword. It needs to be extended and its method implemented.An Abstract class cannot be instantiated.

abstract class syntax

abstract class name {}

b. abstract method

A method that is declared as abstract and does not have implementation is known as abstract method.

abstract method syntax

abstract void methodname ();

Example : abstract class that has abstract method

In below program, there is a class which is the abstract class and it contains only one abstract method with the method name run. There is a class Honda that is derived from Bike class. The implementation of the abstract class Bike is provided by the Honda class.

```
abstract class Bike{
  abstract void run();
}
class Honda extends Bike{
void run()
{
public classAbstractiontest {
System.out.println("running safely..");
} }
```

```
public static void main(String args[]){
 Bike obj = new Honda();
 obj.run();
} }
```

Output

running safely..

c. Understanding the real scenario of abstract class

In below program, Shape is the abstract class, its implementation is provided by the Rectangle and Circle classes. The end user don't know about the implementation class and object of the implementation class is provided by the factory method.

Example: abstraction

```
abstract class Shape{
abstract void draw();
}

class Rectangle extends Shape{
void draw()
{
System.out.println("drawing rectangle");
}
}
class Circle1 extends Shape{
void draw(){
System.out.println("drawing circle");
}
```

```
}
  class TestAbstraction1{
public static void main(String args[]){
Shape s=new Circle1();
 s.draw();
}
}
```

Output

drawing circle

11.9.2 Abstract class having constructor, data member, methods

An abstract class can have data member, abstract method, method body, constructor and even *main()* method.

Example: abstract class that have method body

```
abstract class Bike{
  Bike(){System.out.println("bike is created");}
  abstract void run();
                void changeGear(){System.out.println("gear changed");}
 }
 class Honda extends Bike{
                void run(){System.out.println("running safely..");}
 }
 class Abstraction2test{
 public static void main(String args[]){
  Bike obj = new Honda();
  obj.run();
  obj.changeGear();
 } }
```

Output

```
bike is created
running safely..
gear changed
```

Application activity 11.9

1. Abstract class must have only abstract methods. True or false?
2. Is it compulsory for a class which is declared as abstract to have at least one abstract method?
3. Can we use "abstract" keyword with constructor, Instance Initialization Block and Static Initialization Block?
4. Why final and abstract cannot be used at a time?
5. Predict the output of the below Java program

```
abstract class A
{
 abstract void callme();
 public void normal()
 {
  System.out.println("this is concrete method");
```

```
 }}
class B extends A
{
 void callme()
 {
  System.out.println("this is callme.");
 }}
public class Abstractapplication {
  public static void main(String[] args) {
      B b = new B();
  b.callme();
  b.normal();
  }}
```

END OF UNIT ASSESSMENT

Question 1

Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance variables a part number(type String),a part description(type String),a quantity of the item being purchased (type int) and a price per item (double). Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable. In addition, provide a method named getInvoice Amount that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0. Write a test application named InvoiceTest that demonstrates class Invoice's capabilities.

Question 2

a. Create a super class called Car. The Car class has the following fields and methods.
- Int speed;
- Double regularPrice;
- String color;
- Double getSalePrice();

b. Create a sub class of Car class and name it as Truck. The Truck class has the following fields and methods.
- Int weight;
- Double getSalePrice(); // If weight> 2000, 10% discount.Otherwise,20 % discount.

c. Create a subclass of Car class and name it as Ford. The Ford class has the following fields and methods
- Int year;
- Int manufacturerDiscount;
- Double getSalePrice(); // From the sale price computed from Car class subtract the manufacturerDiscount.

d. Create a subclass of Car class and name it as Sedan. The Sedan class has the following fields and methods.
- Int length;
- Double getSalePrice(); // If length>20feet, 5%discount, Otherwise, 10%discount.

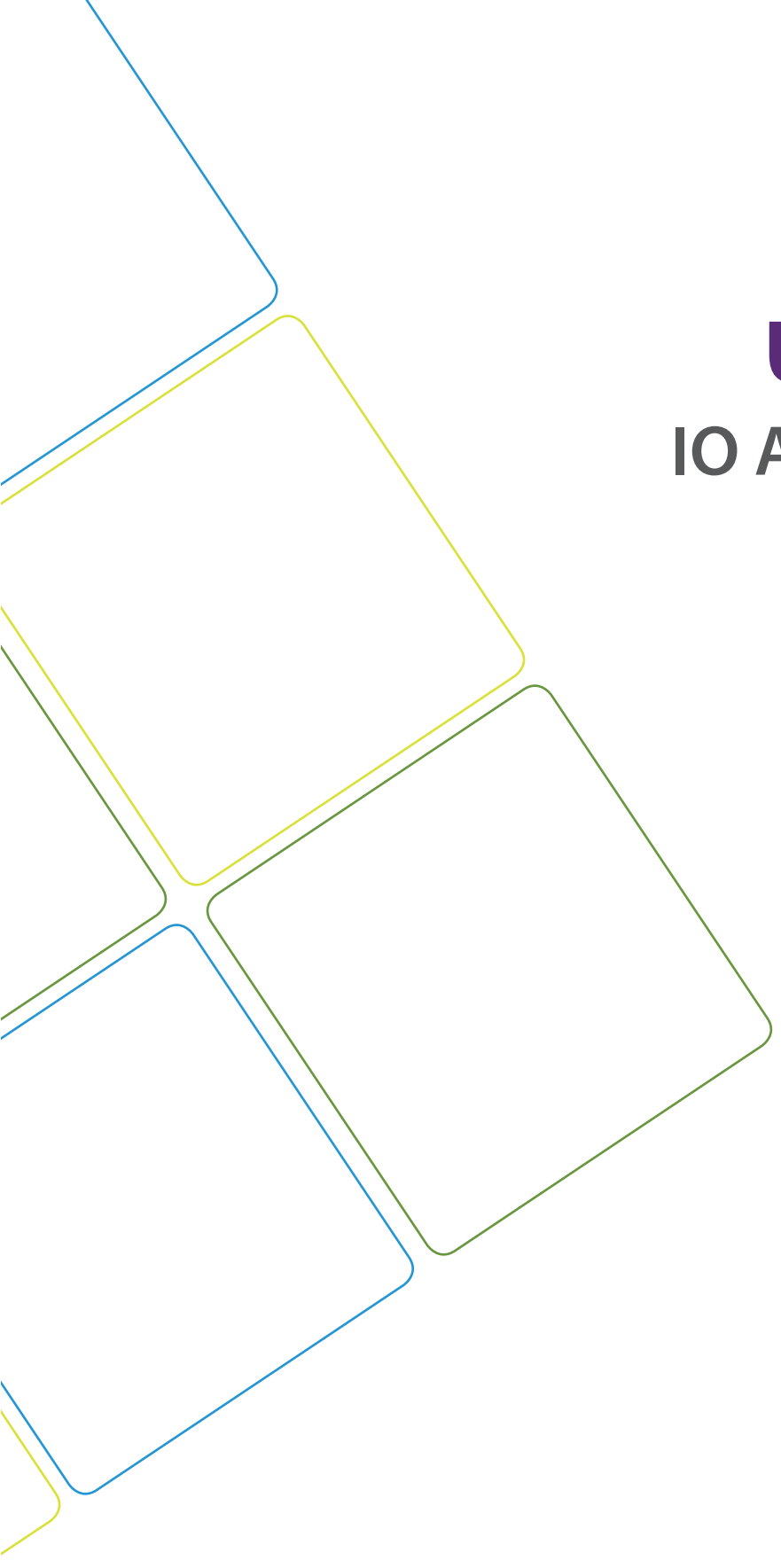
e. Create MyOwnAutoShop class which contains the main() method.

f. Perform the following within the main () method.

i. Create an instance of Sedan class and initialize all the fields with appropriate values. Use super(...) method in the constructor for initializing the fields of the superclass.

ii. Create two instances of the Ford class and initialize all the fields with appropriate values. Use super(...) method in the constructor for initializing the fields of the super class.

iii. Create an instance of Car class and initialize all the fields with appropriate values. Display the sale prices of all instance.



UNIT 12
IO AND JAVA

UNIT 12: IO and Java

Key Unit Competency: to be able to use stream, reader and writer in java

Introductory activity

Teacher Ntwari prepared examination at the District level and he saved the examination in My Documents folder under File name" district Examination". One of the questions is to calculate the following quadratic equation $x^2-7x=0$

1. Explain the path that should be used to access the file contents.
2. Solve the above quadratic equations using java programming language.
3. How many variables needed to solve the above quadratic equation if you need to run it with java programming language?

12.1. Introduction to IO Streams

Learning activity 12.1

1. Discuss the steps to be followed if you are requested to write a program that will calculate and display the sum of 10 numbers that you need to provide during the execution of the program.

The term stream refers to a sequence of values from any input source of data (keyboard, file, port, and so on), or to any output destination for data (screen, file, port, and so on).

They are,

1. **Byte Stream :** It provides a convenient means for handling input and output of byte.
2. **Character Stream :** It provides a convenient means for handling input and output of characters.

12.1.1.Definition of InputStream

The InputStream is an abstract class that describes stream input. In Java programming language the InputStream is used to read data from a source.

This class defines several methods among which include **read()** that reads byte of data

12.1.2.Definition of OutputStream

Outputstream is an abstract class that describes stream output. In Java programming language, the OutputStream is used for writing data to a destination.

This class defines several methods among which include write() that reads byte of data

12.1.3 Role of I/O Stream

IO Streams provide convenient ways of handling data input and output in a java program. While writing java programs, three IO objects are provided implicitly for performing data Input and Output. They are;

- System.in (input stream)
- System.out (output stream for normal results)
- System.err (output stream for error messages)

Generally *System.in* is connected to the keyboard and inputs character data*, and *System.out* and *System.err* are connected to the monitor and output character data.

Application activity 12.1

1. What is a stream and what are the types of Streams and classes of the Streams?
2. Differentiate the following terms:
   a. System.in and System. out
   b. output stream and input stream

12.2. InputStream, OutputStream and FileStream

Learning activity 12.2

From the previous lessons in this unit, we have learned the standard I0 stream in Java. Write and Discuss in groups the source code program allows user to calculate area of any given triangle if base and height that are entered through the keyboard and print the area

The goal of InputStream and OutputStream is to abstract different ways to input and output: whether the stream is a file, a web page, or the screen shouldn't matter. All that matters is that information is received from the stream (or send information into that stream.)

12.2.1 OutputStream

Java application uses an output stream to write data to a destination, it may be a file, an array, peripheral device or socket.

Code of Printing "WELCOME WORLD" without advancing to a new line

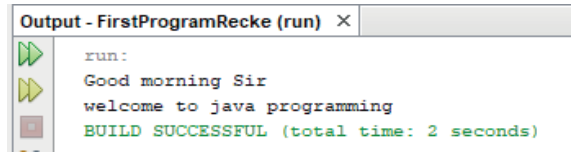
System.out.print(" WELCOME WORLD");

Code of Printing "WELCOME WORLD " and advance to a new line.

System.out.println(" WELCOME WORLD");

Example

```
class FirstProgram{
 public static void main(String[] args)
{
   System.out.println("Good morning
Sir" );
       System.out.println("welcome  to
java programming " );
 }}
```



12.2.2. InputStream

Java application uses an input stream to read data from a source, it may be data, a file, and an array. There are different input stream classes that can be applied to input data from user.

- scanner()
- BufferedReader()
- DataInputSteam()

a. Using Scanner class.

Java has a number of predefined classes which can be used. Predefined classes are organized in the form of packages. A Package can be defined as a grouping of related types (classes, interfaces, enumerations and annotations) providing access protection and namespace management.

In Java, data can be input with the help of the **Scanner** class. This **Scanner** class is found in **java.util** package. So, to use the Scanner class, java.util package needs to be included first in the program. Apackage is included in a program with the help of **import** keyword.

The **java.util.Scanner** class or the entire **java.util** package are then imported.

The advantage of using Scanner class is that it provides convenient methods for captur-

ing primitives (nextInt(), nextFloat(), …) from input devices whereas its **disadvantage** is that reading methods are not synchronized .

To import a class or a package, one of the following lines should be added to the very beginning of the code.

import java.util.Scanner; *// This will import just the Scanner class*

import java.util.*; *// This will import the entire java.util package*

After importing, the following statementwill be valid when written in the program.

Scanner k = new Scanner (System.in);

Scanner k, here K is declared as an object of **Scanner** class. **System.in** within the round brackets tells Java that this will be System Input i.e. input will be given to the system.

The main purpose of the **Scanner** class is to parse primitive types and strings using regular expressions, however it can also be used to read input from the user in the command line. Here is an example:

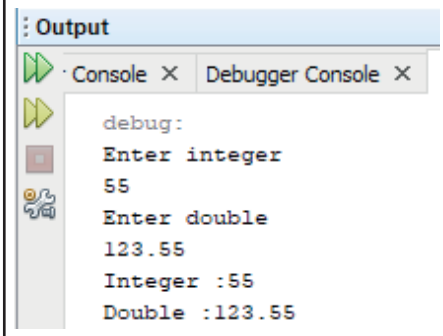
```
Scanner scanner = new Scanner( System.in );  //Create a Scanner using the InputStream available.
System.out.print( "Type some data for the program: " ); //Don't forget to prompt the user
String input = scanner.nextLine();              // Use the Scanner to read a line of text from the user.
System.out.println( "input = " + input );       // Now, you can do anything with the input string that you need to.
                                                // Like, output it to the user.
```

List of datatypes.

Method	Inputs
nextInt()	Integer
nextFloat()	Float
nextDouble()	Double
nextLong()	Long
nextShort()	Short
next()	Single word
nextLine()	Line of Strings
nextBoolean()	Boolean

12.1 table data type

Example 1: a programme that asks to enter numbers from the keyboard and displays those numbers on the screen

import java.util.Scanner; *import java.util.*;* *class I2{* *public static void main(String[] args){* *Scanner s1 = new Scanner(System.in);* *System.out.println("Enter integer");* *int n = s1.nextInt();* *System.out.println("Enter double");* *double db = s1.nextDouble();* *System.out.println("Integer :" + n);* *System.out.println("Double :" + db);* *}}*	OUTPUT 

n= s1.nextInt(); s1.nextInt() will take integer input from the user and it assigns it to n. . So, this statement will be like **n = 14;** when user entered 14.

Similarly, values of other data types can be also input . Same as nextInt() is used to input an integer value, other methodss in **table 12.1** will be used to input those values.

b. Using BufferedReader Class

Java BufferedReader class is used to read the text from a character-based input stream. It can be used to read data line by line by readLine() method. It makes the performance fast. It inherits Reader class.

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

System.out.print("Enter your name: ");

String name = reader.readLine();

System.out.println("Your name is: " + name);

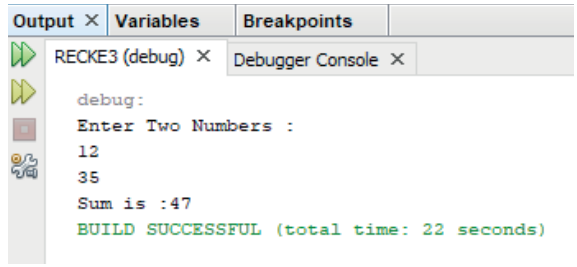
In the above example, the **readLine()** method reads a line of text from the command line.

Advantages: The input is buffered for efficient reading and its disadvantageis that the wrapping code is hard to remember.

Example of program using BufferedReader

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
class Test {
  public static void main(String args[]) throws Exception {
    int a, b, s;
        InputStreamReader is = new InputStreamReader(System.in);
          BufferedReader br = new BufferedReader(is);
        System.out.println("Enter Two Numbers :");
    a = Integer.parseInt(br.readLine());
    b = Integer.parseInt(br.readLine());
s = a + b;
    System.out.println("Sum is :" + s);
  }}
```

OUTPUT



Notice that a "keyboard object" for reading from the console keyboard can therefore be instantiated as follows:

InputStreamReader inStream = new InputStreamReader(System.in);

BufferedReader keyboard = new BufferedReader(inStream);

The intermediate variable `inStream`could be eliminated and it can be done in one step like this:

```
BufferedReader keyboard =
  new BufferedReader(new InputStreamReader(System.in));
```

c. DataInputStream

InputStream uses **readLine()** method of DataInputStream to read data from the keyboard. Java InputStreams are used for reading byte based data, one byte at a time.

The following example describes how to use datainputstream wih different data types

Example 1

```
import java.io.*;
public class KeyboardReading1
{
   public static void main(String args[]) throws IOException
{
    try (DataInputStream dis = new DataInputStream(System.in)) {
       System.out.println("Enter your name: ");
       String str1 = dis.readLine();
          System.out.println("I know your name is " + str1);
          System.out.println("Enter a whole number: ");
       String str2 = dis.readLine();
       int x = Integer.parseInt(str2);
       System.out.println("Enter a double value: ");
          String str3 = dis.readLine();
       double y = Double.parseDouble(str3);
       if(x > y)
          System.out.println("First number " +x + " is greater than second number " + y);
       else
          System.out.println("First number " +x + " is less than second number " + y);
    } }}
```

OUTPUT

```
run:
Enter your name:
UWERA
I know your name is UWERA
Enter a whole number:
12
Enter a double value:
15.5
First number 12 is less than second number 15.5
BUILD SUCCESSFUL (total time: 41 seconds)
```

Example 2

```
import java.io.*;
public class KeyboardReading1
{
 public static void main(String args[]) throws IOException
 {
       try (DataInputStream dis = new DataInputStream(System.in)) {
          System.out.println("Enter your name: ");
       String str1 = dis.readLine();
        System.out.println("I know your name is " + str1);
        System.out.println("Enter a whole number: ");
       String str2 = dis.readLine();
      int x = Integer.parseInt(str2);
       System.out.println("Enter a double value: ");
       String str3 = dis.readLine();
       double y = Double.parseDouble(str3);
       if(x > y)
          System.out.println("First number " +x + " is greater than second number " + y);
       else
        System.out.println("First number " +x + " is less than second number " + y);
    }}}
```

OUTPUT

```
FirstProgramRecke (run) × FirstProgramRecke (debug) ×
run:
Enter your name:
KARENZI
I know your name is KARENZI
Enter a whole number:
12
Enter a double value:
35
First number 12 is less than second number 35.0
BUILD SUCCESSFUL (total time: 32 seconds)
```

Explanations:

The **readLine()** method of **DataInputStream** reads a line at a time and returns a string, irrespective of what the line contains. Depending on the input value, the string is to be parsed into an **int** or **double** etc. This is extra work in the keyboard reading when compared to C/C++. In C/C++, parsing is done implicitly by %d, %f, etc.

What is "System.in"?

" **in**" is an object of " **InputStream**" class defined in System class (like "out" is an object of PrintStream class defined in System class). It is declared as static and final object. The in object is connected implicitly to the " standard input stream" of the underlying OS.

DataInputStream dis = new DataInputStream(System.in);

System.in is a byte stream which is automatically connected to the keyboard reading mechanism of operating system. It is chained to DataInputStream to facilitate the reading of user input with its readLine() method.

Like **System.in** is connected to the keyboard mechanism of OS, the **System.out** is connected to the standard output mechanism of OS.

Application Activity 12.2

1. Write a java program that allows the user to enter three integers and that displays their sum
2. Write a java program asking the user to enter a number and print its factorial.
3. Write a java program that allows the user to enter the age of KARENZI, SINGIZWA and MPORE and print the youngest (use Scanner , input stream and console)

12.3 FileStreams

Learning activity 12.3

```
import java.io.FileOutputStream;
public class FileHandlingEx {
      public static void main(String[] args){
            try(FileOutputStream fout = new FileOutputStream("Sample.txt")) {
 String s="I am writing the source code.";
byte b[]=s.getBytes();//converting string into byte array
fout.write(b);
}catch(Exception e){
 System.out.println(e);
 }
System.out.println("Write java source code please.");
 }
}
```

1. Run this program
2. What is the role of FileInputStream and FileOutputStream in the above program? 3) What is the output of the above program?
3. **What is the output of the above program?**

a. 1 FileInputStream

This **FileInputStream** is used for reading data from the files. Objects can be created using the keyword **new** and there are several types of constructors available.

The following constructor takes a file name as a string to create an input stream object to read the file:

InputStreamf=newFileInputStream("C:/java/hello");

The following constructor takes a file object to create an inputstream object to read the file. First a file object using File()methods created as follows:

Filef=newFile("C:/java/hello"); InputStreamf=newFileInputStream(f);

Once *InputStream* object i s in hand, then there is a list of helper methods which can be used to read from the stream or to do other operations on the stream.

Methods used for file InputStream

1. **public void close() throws IOException{}:** This method closes the file output-stream. Releases any system resources associated with

The file. Throws an IOException

2. **protectedvoidfinalize()throws IOException{}**: This method cleans up the connection to the file. It ensures that the close method of this file outputstream is called when there are no more references to this stream. It throws an 10 Exception.

3. **publicintread(intr)throws IOException{}**: This method reads the specified byte of data from the InputStream and returns the next byte of data and-1will be returned if it is end of file.

4. **publicintread(byte[]r)throws IOException{}:** This method reads r.length bytes from the inputstream into an array. I t returns the total number of bytes read. If end of file -1 will be returned.

5. **Public int available() throws IOException{}:** Gives the number of bytes that can be read from this file inputstream. It returns an integer.

12.3.1 FileOutputStream:

FileOutputStream is used to create a file and write data into it. The stream would create a file, if it doesn't already exist, before opening it for output.

a. File output stream constructors

The following constructor takes a file name as a string to create an outputstream object to write the file:

OutputStream f=new FileOutputStream("C:/java/hello")

The following constructor takes a file object to create an outputstream object to write the file. First, a file object using File()method is created as follows:

Filef=newFile("C:/java/hello");

OutputStream f=newFileOutputStream(f);

Once the *OutputStream* object is in hand,then there is a list of helper methods,which can be used to write to stream or to do other operations on the stream.

b. File output stream methods

1. public void close() throws IOException{}: This method closes the file outputstream. Releases any systemresources associatedwith the file.Throws anIOException.
2. **protectedvoidfinalize()throwsIOException{}:** This method cleans up the connection to the file. Ensures that the close method of this file outputstream is called

when there are no more references to this stream.Throws an IOException.

3. **publicvoidwrite(intw)throws IOException{}:** This methods writes the specified byte to the outputstream.
4. **Public void write(byte[]w):** It writes w.length bytes from the mentioned byte array to the OutputStream.

The following is the example to demonstrate FileInputStreamand FileOutputStream:

Example: Java FileInputStream to read single character

```
import java.io.FileInputStream;
public class DataStreamExample {
   public static void main(String args[]){
      try{
       FileInputStream fin=new FileInputStream(«D:\\testout.txt»);
       int i=fin.read();
       System.out.print((char)i);
        fin.close();
      }catch(Exception e)
{
System.out.println(e);
} }
}
```

Notice that before running the code, a text file named as ***"testout.txt"*** is required to be created. If in n this filethere are the following contents:

Hello Programmers.

After executing the above program, a single character from the file which is 87 (in byte form) is gotten. To see the text, the byte form converted into character.

Output:

H

Application activity 12.3

```
import java.io.FileInputStream;
public class DataStreamExample
{
   public static void main(String args[])
{
      Try{
       FileInputStream fin=new FileInputStream(«D:\\testout.txt»);
       int i=fin.read();
       System.out.print((char)i);
       fin.close();
      }catch(Exception e){System.out.println(e);}
     }
    }
```

1. What is required before running the source code?
2. What is its output?
3. Write a java program that reads input from text file that contains the following content: "Rwanda is a country of one thousand hills located in East Africa Region". This program will then display the whole content as it appears.

12.4 Readers and writer

Learning activity12.4

```
Reader reader = new FileReader("c:\\data\\myfile.txt");
  int data = reader.read();
  while(data != -1){
    char dataChar = (char) data;

data = reader.read();
  }
```

After running the above program, what is the role of reader.

The Java Reader (java.io.Reader) and Java Writer class (java.io.Writer) in Java IO work much like the InputStream and OutputStream with the exception that Reader and Writer are character based. They are intended for reading and writing text while the InputStream and Out-

putStream are byte based.

12.4.1 Reader

The Java Reader is the base class of all Readers in the Java IO. The subclasses include a BufferedReader, PushbackReader, InputStreamReader, StringReader and several others.

Combining Readers With InputStream

A Java `Reader` can be combined with an `InputStream`. To read characters from an `InputStream`, wrap it in an `InputStreamReader` and pass the `InputStream` to the constructor of the `InputStreamReader` as follows;

Reader reader = new InputStreamReader(inputStream);

12.4.2 Writer

The Java Writer class is the base class of all Writers in the Java IO API. The subclasses include BufferedWriter and PrintWriter among others.

Here is a simple Java IO Writer example:

```
Writer writer = new FileWriter("c:\\data\\file-output.txt");

writer.write("Hello World Writer");
writer.close();
```

Combining Writers With OutputStreams

A Java `Writer` can be combined with an `OutputStream` just like `Readers` and `InputStream`'s. Wrap the `OutputStream` in an **`Output Stream Writer`** and all characters written to the `Writer` are passed on to the `OutputStream`.

Here is an `Output StreamWriter` example:

```
Writer writer = new OutputStreamWriter(outputStream);
```

Combining Readers and Writers

Just like with streams, `Reader`s and `Writer`s can be combined into chains to achieve more interesting IO. It works just like combining the `Reader` with `InputStream`'s or the `Writer` with `OutputStream`'s. For instance, buffering is achieved by wrapping a `Reader` in a **`BufferedReader`**, or a `Writer` in a `BufferedWriter`.

Here are two such examples:

```
Reader reader = new BufferedReader(new FileReader(...));
Writer writer = new BufferedWriter(new FileWriter(...));
```

12.5 Character stream

Java Byte streams are used to perform input and output of8-bit in onebyte, whereas Java Characterstreams are used to perform input and output for 16-bitunicode. Thought here are many classes related to character streams but the most frequently used classes are, File Reader and File Writer. Though internally File Reader uses File Input Stream and File Writer uses File Output Stream but here major difference is that File Reader reads two bytes at a time and File Writer writes two bytes at a time.

The above example(12.4 b)which makes use of the setwoclasses to copy an inputfile (having Unicode characters)into an output file is re-written as follow:

```
package javaLesson;
import java.io.FileWriter;
public class FileWriterExample {
  public static void main(String args[]){
    try{
     FileWriter fw=new FileWriter("D:\\testout.txt");
     fw.write("Welcome to java Lesson 12.");
     fw.close();
    }catch(Exception e){System.out.println(e);}
    System.out.println("Success...");
  }
}
```

Output: Welcome to java Lesson 12.

Standard stream

All the programming languages provide support for standard I/O where user's program can take input from a key board and then produce output on the computer screen. In C or C++ programming languages, there are three standard devices STDIN, STDOUT and STDERR. Similarly, Java provides the following three standard streams:

i. **Standard Input:** This is used to feed the data to user's program and usually a keyboard is used as standard input stream and represented as **System.in**.

ii. **StandardOutput:** This is used to output the data produced by the user's program and usually a computer screen is used to standard output stream and represented as **System.out**.

iii. **Standard Error:** This is used to output the error data produced by the user's program and usually a computer screen is used to standarderror stream and represented as **System.err**.

The following is a simple program which creates **InputStreamReader** to read from standardinput stream until the user types a"q":

```
import java.io.*;
public class ReadConsole {
public static void main(String args[]) throws IOException
{
InputStreamReader cin = null;
try {
cin = new InputStreamReader(System.in);
System.out.println("Enter characters, 'q' to quit.");
char c;
do {
c = (char) cin.read();
System.out.print(c);
} while(c != 'q');
}finally {
if (cin != null) {
cin.close();
}}}
}
```

Let's keep above code in ReadConsole.java file and try to compile and execute it as below. This program continues reading and outputting same character until we press 'q':

```
$javac ReadConsole.java
$java ReadConsole
Enter characters, 'q' to quit.
1
1
e
e
q
q
```

a. BufferedReader

The Java BufferedReader class (java.io.BufferedReader) provides buffering to the Reader instances. Buffering can speed up IO quite a bit. Rather than read one character at a time from the network or disk, the BufferedReader reads a larger block at a time. This is typically much faster, especially for disk access and larger data amounts.

The Java BufferedReader is similar to the BufferedInputStream but they are not exactly the

same. The main difference between them is that BufferedReader reads characters (text), whereas the BufferedInputStream reads raw bytes.

Example

```
BufferedReader bufferedReader = new BufferedReader
        (new FileReader("c:\\data\\input-file.txt"));
```

b. InputStreamReader

The Java InputStreamReader class (java.io.InputStreamReader) is intended to wrap an InputStream, thereby turning the byte based input stream into a character based Reader. The Java InputStreamReader is often used to read characters from files (or network connections) where the bytes represents text. For instance, a text file where the characters are encoded as UTF-8. An InputStreamReader could be used to wrap a FileInputStream in order to read such a file.

InputStreamReader Example:

```
InputStream inputStream      = new FileInputStream("c:\\data\\input.txt");
Reader     inputStreamReader = new InputStreamReader(inputStream);
int data = inputStreamReader.read();
while(data != -1){
  char theChar = (char) data;
  data = inputStreamReader.read();
}
inputStreamReader.close();
```

This example first creates a FileInputStream and wraps it in an InputStreamReader. Second, the example reads all characters from the file via the InputStreamReader.

Closing an InputStreamReader

When the reading of characters is finished from the InputStreamReader, then there is a need to to close it. Closing an InputStreamReader will also close the InputStream instance from which the InputStreamReader is reading.

Closing an InputStreamReader is done by calling its close() method. Here is how closing an InputStreamReader looks:

```
inputStreamReader.close();\
```

Application Activity 12.4

Discuss on

1. Reader and Writer
2. InputStreamReader and buffered reader

END OF UNIT ASSESSMENT

1. Write a java program that allows user to input data through the keyboard and calculate the sum, average and difference
2. Write a java program that asks user to enter any number and find out if the number is prime number or not.
3. Analyze the following source code and find out its output.
4. import java.util.Scanner;

```
class AreaTriangleDemo {
  public static void main(String args[]) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter the width of the Triangle:");
    double base = scanner.nextDouble();
    System.out.println("Enter the height of the Triangle:");
    double height = scanner.nextDouble();
    double area = (base* height)/2;
    System.out.println("Area of Triangle is: " + area);
  }}
```

BIBLIOGRAPHY

1. LOFTUS, L. &. (2003). ***Foundation of program design 3 edition .*** United States of America: Pearson Education.
2. Galiwango Kityo Michael,Luzinda Roland Babumba,Sserwanja Joseph. (2013). ***MK Senior Secondary Computer Science Student's Book Grade 6.*** Kigali: MK Publishers Ltd.
3. Herman Oduor. (n.d.).
4. Oduor, H. (2013). ***Computer Science For Rwanda Student's Book 6.*** Kigali: East African Publishers Rwanda Ltd.
5. Oduor, H. (2013). ***Senior Secondary Certificate Computer Science For Rwanda Student's Book 6.*** Kigali: East African Publishers Rwanda Ltd.
6. Owoyesigyre Davis , Stephen Mburu , Geoffrey Chemwa. (2016). ***Computer Science For Rwandan Schools Senior 4.*** Kigali: Longhorn Publishers Ltd.
7. Abraham Silberschartz, H. F. (2011). ***Database System Concepts*** (Vol. 6th Edition). New York: McGraw-Hill.
8. Eng., A. W. (2012). ***Database design*** (Vol. 2nd Edition). Western Ontario.
9. Last, P. J. (2012). ***Concepts of Database Management*** (Vol. 8th Edition). Boston; Cengage Learning.
10. Scheider, R. D. (1997). ***Microsoft SQL Server:Planning and Building a High Performance Database.*** Prentice Hall PTR.
11. Sylvia Langfield, D. D. (2015). ***Cambridge International AS and A level Computer Science.*** Cambridge CB2 BBS, United Kingdom: Cambridge University Press .
12. Thalheim, B. (2000). ***Fundamentals of entity-relationship model .*** New York: Springer.
13. Williams, D. W. (2014). ***Cambridge IGCSE Computer Science.*** London: London NW1 3BH Hodder Education A Hachetter UK Compan
14. ***http://tutorials.jenkov.com***
15. ***https://www.codesdope.com/***
16. ***https://www.javatpoint.com/java-console-class*** . (2018, may 9).
17. ***https://www.safaribooksonline.com/***
18. ***https://www.lifewire.com***
19. ***www.tutorialpoint.com***