# Codenera no. :- 01

# Operating system :- Intermidatior betn user and Computer.

# Operating System :-

An operating system is a program that acts as an interface betn the user and Computer hardware and Controls the execution of all kinds of programs.

Ex - Windows, Linux, mack os, etc.

# Software :-

Software is a set of instructions, data or program used to operate Computers and executes specific task.

There are two types of software application software and system software.

- **Application software :-**

Is a Computer software package that performs a specific function for a user or in some cases for another application.

Ex - Calculator, paint, etc.

- **System Software :-**

System software is a are the programs which is designed to run a Computer application program and hardware.

System Software Cordinate the activity and functions of hardware and Software.

Ex - (os) operating system.

| * Application Software :- | System Software. |
|---|---|
| i) Software that perform specific task to user need. | i) general purpose software that manage basic system resources and processes. |
| ii) wReturn in high level languages such as python and java. | 2) wReturn in low level assembly language or machine code. |
| iii) Does not take hardware in to account and doesn't intract directly with hardware. | iii) Must meet specific hardware needs intract closely with hardware. |
| iv) User or admin install software when needed. | iv) Insatalled at the same time as the (OS) operating system usually by the manifacture. |
| v) Users triggers and stops the programs. | v) Run anytime the computer is ON. |
| vi) Is not needed to system to function. | vi) Is necessary for the system to function. |

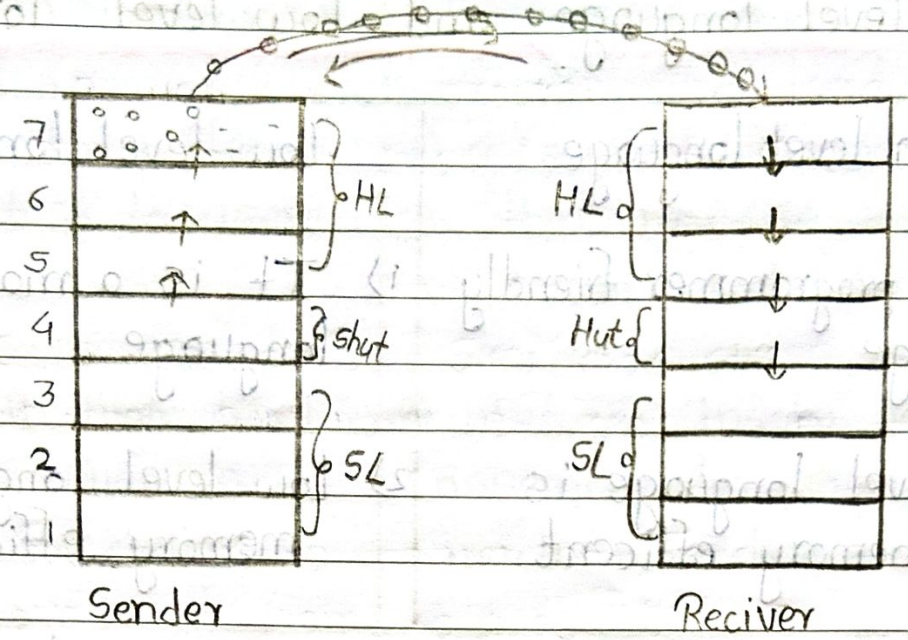
# Programs :-

A program is a set of instructions that a computer using to perform a specific function. program is created using specific programming language c, c++, python, Java, etc.

# High level language and Low level language :-

| High level language | Low level language. |
|---|---|
| 1) It is programmer friendly language. | 1) It is a machine friendly language. |
| 2) High level language is less memory efficent. | 2) Low level language is high memory efficient. |
| 3) It is easy to understand. | 3) It is tough to understand. |
| 4) It is easy to debug. | 4) It is Complex to debug. |
| 5) It is Simple to maintain. | 5) It is Complex to maintain. |
| 6) It is portable. | 6) It is non portable. |
| 7) It can run on any platform. | 7) It is machine dependent language. |
| 8) It needs Compiler or interpreator for transaction | 8) It needs assembler for translator |
| 9) It is used widly for programming. | 9) It is not Commanly used nowadays. |

## * Networking :-




## * Pointer :-

Pointer is a Variable which Copy the address of another Variable "*".

## # History of Java programming language :-

Java is a object-oriented programming language developed by James Gosling in the early 1990 s. The team initiated this project to develop a language for digital device such as set-top boxes, television, etc. Originally C++ was Considered to be used in the project but the idea was rejected for several reasons (for instance C++ required more memory); Gosling endeavoured to alter and expand C++ however befor long surrendered that for making another stage Called Green James Gosling and his team Called their project ~~and later~~ "Greentalk" and its file extension was .gt and later became to known as "OAK".

- **Why OAK :-**

The name OAK was used by Gosting after an OAK tree that remained outside his office. Also, OAK is an image of Solidarity and picked as a national tree of numerous nations like the U.S.A, France, Germany, Romania etc. But they had to later rename it as "JAVA" as it was already a trademark by OAK Technologies.

Team of Java Greator :-
James Gosling, Mike sheridan, and patrick Naughton.

Company name :- Sun engineer / sun Naughton microsystems
Now a subsidian oracle in 1995.

* **JAVA :-**

Gosling and his team did a brainstorming session and after the session, they Came up with several names such as JAVA, DNA, STLK, RUBY, etc.
Java name was decided after much discuss since it was so unique. The name Java originates from a Sort of espress a bean. Java Gosling Comes with this name while have Coffee near his office.
Java was Created on the principles like Robust, portable Independent, High performance, Multithead, etc.

**Java Versions :-**

JDK Alpha + Beta — 1995
JDK 1.0 — 23rd 1996
latest Version of Java is JAVA SE 15 → 15th Sep 2022

# What is Java ?

Java is a high level programming language, platform independent, object-oriented and secure programming language.

- Java is developed by SunMicrosystems. (Which is now subsidiary of oracle) in the year 1995).
- James Gosling is a father of Java.
- Before java it name was OAK since OAK was already registed Company, so James Gosling and his team change the name from OAK to Java.
- Java is object oriented However is not Considered as pure object oriented as it provides Support for primitive data types (like, int, char).
- The Java Code first Compiled into byte Code (machine independent Code). Then the byte Code run on Java Virtual machine (JVM) regardless of the underlying architecture.
- Java syntax is similar to c/c++. But Java does not provide low level programming functionalities like pointers. Also java Code are always written in the form of Classes and object.
- Java is used in mobile application (android is java based); desktop application, Web application, cilent Server, application, enterprize applications and many more.

# What is Java platform ?

Java platform is a Collection of program

that help to programmers to develop and run Java programming application efficently.
It includes an execution engine, a compiler, and set of libriaries in it.

# Importance of Java application :-

- It is use for developing Andriod apps.
- Helps you to create enterprise software.
- Wide range of mobile java applications.
- Scientific Computing applications.
- Used for big data anatytics.
- Java programming of hardware devices.
- Used for server - side technologics like applications
- J Boss, glass fish etc.

# Features of java :-

The primary objective of java programming language creation was to make it portable.
simple and secure programming language.

1) object oriented
2) Simple
3) Secured
4) platform Independent
5) Robust
6) Portable
7) Architecture netural
8) Dynamic
9) Interpreted
10) High performance

11) Multithreaded
12) Distributed.

# Components of java programming Language :-

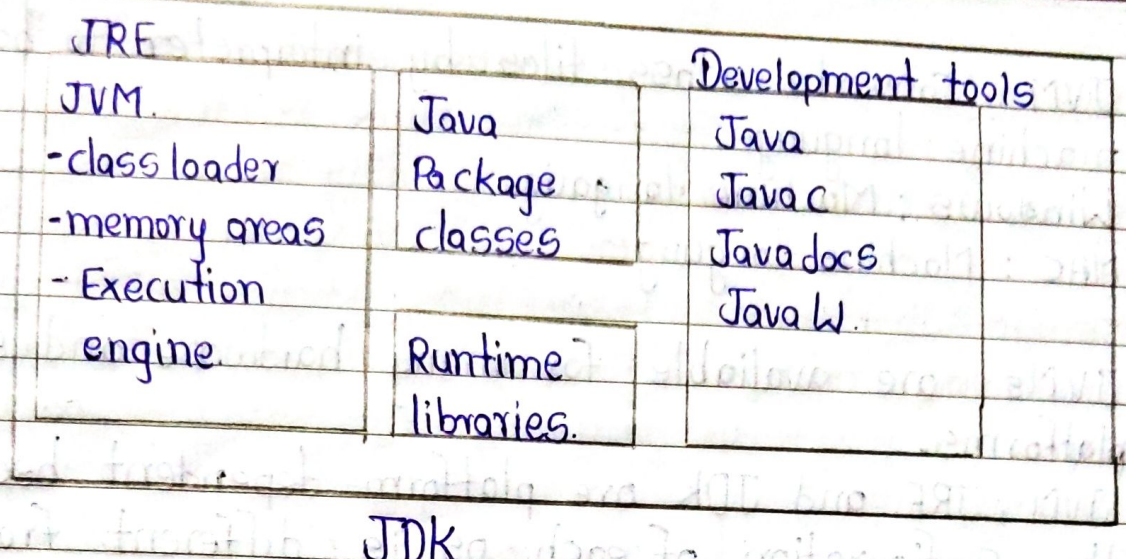
1) Java Development kit (JDK).
2) Java Runtime Environment (JRE).
3) Java Virtual Machine (JVM).

1) Java Development kit (JDK) :-

JDK is ana Gonym for Java development kit. The java development kit (JDK) is a Software development environment which is used to develop Java applications and applets. It physically exists. [It Contains JRE + development tools]

- JDK implementation of any one of the below.
  i) Standard Edition Java platform (SE) J2SE9.
  ii) Enterprise Edition Java platform (EE) J2EE
  iii) Micro edition Java platform (ME) J2ME.

The JDK Contains a private Java virtual machine (JVM) and a few other resources such as an interpeter / loader (java) a Compiler (javac) an a archive (jar) a documentation generator (javados) etc.




## 2] Java Runtime Environment (JRE) :-

JRE is an acronym for Java runtime environment. It is also written as Java RTE. The Java Runtime environment is a set of software tools which are used for developing java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It Contains a set of libraries + other files that JVM uses at runtime.

## 3] Java Virtual Machine (JVM) :-

JVM is an abstract machine. It is Called a Virtual machine. because it doesn't physically exist. It is a Specification that provides a runtime environment in which Java bytecode Can be executed. It Can also run those programs which are written in other langauges and Completed to Java byte code.

- Bytecode is intermediary Code which is used for JVM. It extension name is class.

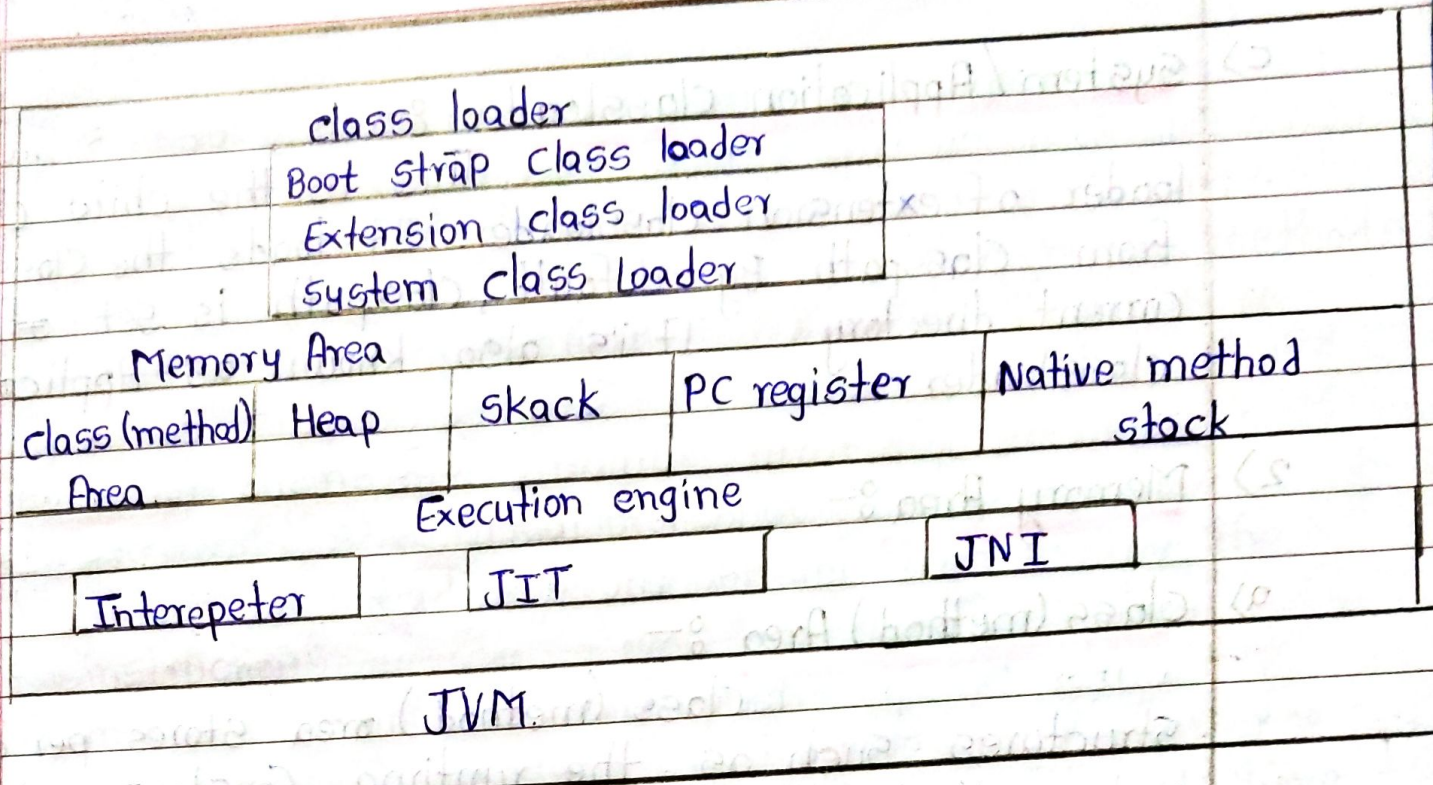
- JVM Converts class files by interpreter to base machine language.
  Windows : Machine langauge.
  MAC : Machine language.

- JVMs are avaliable for many hardware and software platforms.
  JVM, JRE and JDK are platform dependent because the Configuration of each OS is different from each other.
- However java platform is independent.

# There are three notions of the JVM :-
1) Specification.
2) Implementation.
3) Instance.

* JVM performs the following tasks.
  - Loads Code.
  - Verifies Code
  - Executes Code
  - Provides runtime environment.

* JVM Architecture :-




1) **Class Loader :-**

class loader is a subsystem of JVM while is used to load class. files. Whenever we run the java program, it is loaded first by the classloader. There are three built in class loaders in java.

a) **Bootstrap Class Loader :-**

This is the first classloader which is the super class of extension classloader. It loads the rt.jar file which contains all class. Files of java standard edition like java.long package classes, java.net package classes, java.util, package classes, java.io package classes etc.

b) **Extension Classloader :-**

This is the child classloader of bootstrap and parent classloader of system classloader. It loads the jar files located inside any directory.

c) System / Application Classloader :-

This is the child class-loader of extension classloader. It loads the Classfiles from Classpath. By default, Classpath is set ~~of~~ to current directory. It is also known as Application classloader.

2) Memory Area :-

a) Class (method) Area :-

Class (method) area stores per class structures such as the runtime Constant pool field and method data, the Code for methods.

b) Heap :-

It is the runtime data area in which object are allocated.

c) Stack :-

Java Stack stores frames. It holds local Variables and partial results and plays a part in method invocation and return.

Each thread has a provide JVM Stack, Created at the Same time as thread. A new frame is Created each time a method is invoked. A frame is destroyed when its method invocation Completes.

d) program Counter Register :-

PC (program Counter) register contains the address of the java Virtual machine instruction

Currently being executed.

e) Native method stack :-

It Contains all the native methods used in the application.

3) Execution engine :-

a) A virtual processor.

b) Interpreter : Read bytecode stream then execute the instructions.

c) Just In - Time (JIT) Compiler.

It is used to improve the performance JIT Compiles parts of the byte Code that have similar functionality at the same time, and hence reduces the amount of time needed for Compilation, Here the term "Compiler" refers to a translator from the instruction set of a java Virtual machine (JVM) to the instruction set of a specifier CPU.

4) Java Native Interface :-

JNI is a framework which provides an interface to Communicate with another application written in another language like c, c++, Assembly etc. Java uses JNI framework to send output to the Console or interact with OS libraries.

# Different types of java platform :-

There are 4 platforms of the java programming language

i) Java platform, Standard edition (Java SE).

ii) Java platform enterprise edition (Java EE).
iii) Java platform, microedition (Java ME).
iv) Java Fx (Java FE).

i) Standard Edition (Java SE) :-

Java SE, a part of java software platform family, uses Java programming language, to develop stand alone applications. It provides the core functionality of java. Java SE was earlier known as Java 2 platform, Standard edition (J2SE)

ii) Enterprise edition (Java EE) :-

Java EE a port of java software platform family, is based on java SE specification. It contains Java APIs that can be used to write server side applications. Java EE is earler known as Java 2 platform enterprise edition (J2EE).

iii) Java platform, micro edition (Java ME) :-

Java ME, a pare of java software platform family uses the java programming language and provides APIs to develop mobile applications. Java ME was earlier known as Java 2 platform, micro edition (J2ME)

iv) Java fx :-

It is a port of java software platform family that is used to create rich internet application. It uses a lightweight user interface API. The

applications built using Java Fx Can run on devices like desktop Computers, mobile phones, etc.

# Variables in Java :-

- A Variables is a name given to a memory location. It is the basic unit of storage in a program.
- The Value Stored in a Variable Can be changed during program execution.
- A variable is only a name given to a memory location all the operations done on the Variable effects that memory location.
- In Java C and C++ all variables must be declared before use.
- A Variable name Can consist of Capital letter A-Z, lower Case letter a-z, digits 0-9 and two Special characters such as underscore (_) and Doller ($).

## * Rules for Creating Variable :-

- The first character must be a letter.
- Blank spaces Cannot be used in Variable name.
- Java keywords Cannot be used as Variable names.
- Variable names are Case-sensitive.
- We Can use only two Special character i.e. ($) & (_).

# Keywords :-

- It is predefine word ~~wich~~ ~~wich~~ which has Some special meaning.
- They keywords "Const" and "goto" are reserved even though they are not Currently used.

→ True, false and null might seem like keywords but they are activity literals. You cannot use them as identifiers in your programs.

★ Public ↓ Access modifier

Static ↓ non-access modifier

Void ↓ It could not return any thing

main ↓ method

String[] ↓ arguments

args ↓ Variable

Class which is in package long

for o/p

for new line

System.Out.print In ()← method – It is used to terminate the line.

# Data types in java :–

Data types specify the different sizes and that can be stored in the variable. There are two types of datatypes in java:-

1) Non-primitive datatypes.
2) Primitive datatypes.

1) Non-primitive datatype :–

The length and size is not fixed is known as non-primitive data type.
Ex- Array, String, classes, Interfaces.

2) Primitive datatypes :–

The size and value are fixed is known as primitive datatypes.
Ex- Boolean, char, byte, short, int, long, float as double.

| | Data type | Default Value | Defaulth Size |
|---|---|---|---|
| 1) | Boolean | False | 1 bit |
| 2) | Char | '\U0000' | 2 byte |
| 3) | Byte | 0 | 1 byte |
| 4) | Short | 0 | 2 byte |
| 5) | Int | 0 | 4 byte |
| 6) | long | 0L | 8 byte |
| 7) | Float | 0.0f | 4 byte |
| 8) | decimal | 0.0d | 8 byte |

1) **Boolean Data type :-**

The boolean data types is used to store only two possible Values true and false. This data type is used for simple flags that track true / false Condition.

2) **Byte data type :-**

- The byte data type Value range between (-128 to 127) (inclusive).
- It's default Value is '0'.
- The byte data type is used to save memory in large arrays where the memory savings is most required. It save space because a byte is 4 times smaller than integers.

3) **Short data type :-**

- The short data type can also be used to save memory just like byte data type. A short data type is 2 times smaller than an integer.

- It's Value range is (-32,768 to 32,767).
- It's default Value is '0'.

4) Int data type :-

- The int datatype is generally used as a default datatype for integral Values. unless if there is no problem about memory.
- It's Value -range lies betn ($-2^{31}$ to $2^{31-1}$) inclusive.
- It's defalut Value is '0'.

5) Long data type :-

- The long data type is used when you need a range of values more than provides by int.
- It's Value range is ($-2^{64}$ to $2^{63-1}$) inclusive.
- It's default Value is '0'.

6) Float data type :-

- It's Value range is unlimited.
- It is recommented to use a float if you need to save memory in large arrays of floating point number.
- It's default Value is 0.0f.

7) Double data type :-

- It's Value range is unlimited.
- The double data type is generally used for decimal Values like float.
- It's default Value is 0.0d.

8) Char data type :-

- It's Value - range lies between '\u0000' to '\UFFFF'.
- The char data type is used to store characters.

9) What is Java ?

→ Java is a object - oriented, class based high level programming language. It is platform independent secure. It is widely use for the developing applications with the help of standard edition enterprice edition, micro edition and Java fx. James Gosling is a faster of java he works for sun microsystem he works to Create java in 1990. The first of java is launched in 1995. The latest Version is java 18 in march 22nd 2022.

By the help of byte Code we Can also run those programs which are written in other language. by the help of JVM. The Components of java programming language are JDK, JRE and JVM.

# Scanner Class :-

It is a predefine class which is located in java. until package. To import the scanner class we write in side the Code " import java.until.*

↑ it is used for all Classes in until package.

import java. util. Scanner ; - only for scanner class.

Imp

```
Scanner  Sc = new Scanner (System.in);
   ↓      ↓     ↓
 class  object  keyword.
```

# Program :-

```
import java.until.scanner;
class name
{
 public static void main (String [] args)
{
 Scanner Sc = new Scanner (system.in);
 int a, b;
 a = Sc.nextInt ();
 b = Sc.nextInt ();
/* double c;
   c = Sc.next double ();
*/
 int sum = a+b;
 System.out.println (" Sum of two numbers" + sum)
}
}
```

# JDK (Java development kit) :-

Software development environment which used to develops java application [JRE + development foots]

# JRE (Java Runtime environment) :-

set of software tools which are used developing java application.

- Used to runtime environment.

# JVM (Java Virtual machine) :-

It doesn't physically exist specificly that provides runtime environment in which byte Code Can execute.

# Byte code :-

Computer object code that Can intrepretor Converts into binary machine Code.

Q) What is java.

→ Java is high level object-oriented platform independent class based robust and secure programming language. designed for lesser implementation dependencies development. It is widely use for developing java applications in laptops, data centre, game consoles, scientic super-Computer, cell phones etc.

- Java is object-oriented but not pure object oriented because it provide support for primitive data types like (int, char, etc).
- The java Code first Compiled into byte Code (machine independence code) Then byte Code runs on java Virtual machine.
- Java syntax is similar to c, c++ but java does not provide low level programming functionalities like pointer.
- Java Codes always written in the from of classes + objects.
- James gosling is known as the father of java and java was developed by sun microsystem in the year 1995. Java first Version JDK alpha + Beta realsed on 1995 by Sunmicrosystem. In java there are 3 Components JDK, JRE JVM.

There are Four types of edition (standard edition) It is used to developed standard application (enterprised edition) It is used to write server side application (micro edition) It is used to developed mobile application (Java Fx) It is used to developed rich internale application.

# Inputing of String and character :-

## ★ String Input :-

'A' - character.

"Amon" - "A10" - group of characters.

Imp next () - after blank space doesn't exist.

nextline () - make all story it doesn't matter.

## ★ Character Input :-

CharAt (0) → String class method.

ex → ch = Sc. next (). charpt (0);

# Operators :-

1) + → It is for addition.
2) - → It is for substraction.
3) * → It is for multiplication.
4) / → It is for division.
5) % → (moduls). It return reminder in Java.

# Uniary operators :-

++, → Increment ⇒ +1
--, → Decrement ⇒ -1

## * Java Comparison operators :-

Comparison operators are used to Compare two Values :-

| | operators | Description. |
|---|---|---|
| 1) | == | Equal to |
| 2) | != | Not equal to |
| 3) | > | Greater than |
| 4) | < | Less than |
| 5) | >= | Greater than or equal to |
| 6) | <= | Less than or equal to. |

# Control Statement :-

Control Statement is the Statement which control the flow of program.
Java Compiler executes the code from top to bottom. The statements in the code are executed according to the order in which they appear. However, Java provides statements that can be used to control the flow of java code. Such statements are called Control flow statements.

# Types of Control statements :-

1) if else.
2) else -if
3) nested if / ladder.
4) Switch Case.

## 1) if else :-

Syntax :-

```
if (condition)
{
 statement;
}
else
{
 Statement;
}
```

## 2) if else if :-

Syntax :-

```
if (condition)
{
  statement;
  Statement;
}
else if (Condition)
{
 statement;
 statement;
}
 ⋮
 n
else
{
 statement;
 statement;
}
```

3) Nested if :-

Syntax :-
```
if (condition)
{
  if (condition)
  {
    if (condition)
    {
      statement;
    }
  }
}
```

4) Switch Case :-

Syntax :-
```
switch (expression)
{
  Case Value 1:
    // Code to be execute;
  break;
  Case Value 2:
  // Code to be executed;
  break;
 default:
  Code to be executed if all Cases are
      not matched;
}
```

# Codenera no. :- 02

## # Loops :-

The java loops is used to iterate a parts of program several times loop in java come in to use when we need to repeatly execute block of statement.

Types of loop are while loop, do while loop, for loop.

It check condition first.

### 1) While :-

Syntax :-

```
Start Pt. → initialization.
            While (Condition)   → boolean
                                ↳ endpt.
            {
              statement;
              iteration or updation → (i++ or i--).
            }
```

First check condition then execute code.

ex.

```
class C1
{
  public static Void main (String [] args)
  {
    int i = 1;   // initialization
    while (i <= 10)
    {            ↳ condition
      System.out.println (i);
        i++;   → iteration
    }
  }
```

Q) Reverse.

```
Class C1
{
public static Void main (string [] args)
{
 int a = 1234, i, rev = 0, rem;
 i = a;
 while (i! = 0)
 {
  rem = i % 10;
  rev = (rev * 10) + rem;
  i = i/10;
 }
System.out.println ("Reverse = " + rev);
}
}
```

Q) Palindrom.

```
Class C1
{
public static Void main (string [] args).
{
 int a = 121, i, rev = 0, rem;
 i = a;
 while (i! = 0)
 {
 rem = i % 10;
 rev = (rev * 10) + rem;
 i = i/10;
 }
 if (a = = rev)
 {
```

```
System.out.println (" No. is a palindrome ");
}
else
{
System.out.println (" No. is not palindrome ");
}
}
}
```

Q) Searching

```
Class C1
{
public Static Void main (string [] args)
{
int a = 12345 ; i, rem, c=0;
int n = 2 // Searching no.
i = a;
while (i! =0)
{
rem = i % 10;
if (rem == n)
{
c++;
}
i = i/10;
}
if (c >0)
{ system. out. println (" Number found = " + c);
}
else
System.out.println (" No. not found ");
}
```

Q) Factorial.

```
import java. util. Scanner;
class fact.
{
 public static Void main (string [] args).
{
 Scanner  Sc = new Scanner (System. in);
 int n, fact = 1, i;
 System. out. println (" Enter any no. to find factorial");
 n = Sc. next Int ();
  i = 1;
 while (i<=n)
 {
   fact = fact * i;
    i++;
 }
  System. out. println (fact);
 }
}
```

Q) Armstrong number.

Number of digits that's power's Sum is equal to that number.

first no is 153  Ex - $1^3 + 5^3 + 3^3$

$= 1 + 125 + 27$

$= 153$

Program

```
Class Armstrong
{
 public static Void main (string [] args)
 {
```

```
int n = 153, i, rem, sum = 0;
i = n;
while (i != 0)
{
 rem = i % 10;
 Sum = Sum + rem * rem * rem.
 i = i/10;
 }
 if (Sum == n)
  System.out.println("No. is a Armstrong");
 else
   System.out.println("No is not Armstrong");
 }
}
```

Q) Fibonacci Series :-

Sum of last two digits = next no.

ex - 0+1 = 1; 1+1 = 2; 1+2 = 3; 2+3 = 5

Fb = 0 1 1 2 3 5

```
import java.util.scanner;
 Class Fb
 {
 public Static Void main (string [] args)
  {
   Scanner sc = new Scanner (System.in);
   int n1 = 0, n2 = 1, n3, i;
   System.out.println(n1 + ---- + n2);
     i = 2;
   int n = sc.nextInt();
   while (i < n)
   {
```

```
n3 = n1 + n2;
System.out.println(" " + n3);
n1 = n2;
n2 = n3;
}
}
}
```

# Typecasting in java :- (Compare on the logic of size).

Type casting is process that Converts data types into another data type.

Convert the Value of one datatype to another datatype.

There are two types of Casting widing type Casting, Narrowing typecasting.

1) Widing typecasting :-

Converting a lower datatype in to higher datatype. is known as widing typecasting. It is also known as implicite Conversion or Casting down.

- It done automatically it's Safe because there is no chance to loose the data.

The target type must be large than the source type.

Ex →
```
byte a = 10
short b = a;
System.out.println(b);
```

2) Narrowing typecasting :-

Converting a higher datatype into a

lower one is Called narrowing type Casting. It is also known as explicte Conversion or Casting up.

- It is done manually by program.
- If we don't do easting than Compiler report to a Compiler time error.

Ex → int a = 10;
System.out.println ("int to double :" +(double)a);

* When we use operator in side the datatype it will be upcast ourself.

byte c = a ⊞ b
↓
It will be upcast, in bigger one
Can it will go into the int.

# Do while loop :-

- It is the loop which is used to Continue flow of the program.
- If Condition is true or false it doesn't matter the loop will execute at a single time.
- If Condition is true then loop will be Continue otherwise it will terminate.

Q) Palindrome by using do while.

```
class palindrome.
{
public Static Void main (String [] args)
 {
   int i, rem, rev = 0;
```

```
int i=n;
do
{
  rem = i % 10;
  rev = (rev * 10) + rem;
while (i! =0)
{
  if (rev = = n)
    System.out.println ("No. is a palindrome");
  else
    system.out.println ("No. is a not palindrome");
}
}
```

Syntax :-

```
initialization
do
{
  // Statement;
    iteration -> if needed
}
while (condition);
```

# While and do-while loop :-

| While | Do while |
|---|---|
| 1) while loop is entry to Control loop. | 1) do while loop is exist Control loop. |

| | While | dowhile. |
|---|---|---|
| 2) | A while loop is a pre test loop. | 2) do while loop is post test loop. |
| 3) | It tests the condition before executing the loop body. | 3) It test the condition at the end of loop body. |
| 4) | There is no semicolon at end of the loop | 4) There is semicolon at end of the loop. |
| 5) | Ex - while(n<10)<br>{<br>print(" ");<br>} | 5) Ex - do<br>{<br>Print(" ");<br>}<br>while (n < 10); |

Q) Sum of add each number.

```
class add
{
public static Void main (String [] args)
{
int i = 1, 2 3 4 5;
int sum = 0, rem, count = 0;
while (i! = 0)
{
rem = i % 10;
if (i % 2! = 0)
```

```
{
    sum = sum + rem;
    Count ++;
  }
  i = i/10;
}
System.out.println(" sum of odd no. = " + sum );
System.out.println(" Count= " + count)
}
}
```

# For loop :-

Syntax :-

```
for (initialization; Condition; iteration)
↑
Keyword.
```

Ex →

```
for (i=1; i<=10; i++)
    System.out.println (i);
```

- In the for loop, the first iteration check the initialization and Condition. it Could not check the interation.
- on the second time it will check the iteration.

# Nested loop :-

Syntax :-

```
        initialization.
        while (Condition)
Parent loop → {
        initialization
```

```
child loop → while (condition)
        {
            statement
            iteration
        }
    iteration.
}
```

# Nested do while loop :-

```
initialization
do
{
 initialization
 do
 {
   // statement;
   // iteration;
 } while (condition);
 // iteration;
} while (condition);
```

# prime numbers :-

- which is divisible by 1 and itself is Called as prime numbers.

- To check it is divisible between 1 + n which to give  Ex - 1, 2, 3, 4, 5

divisible by both it is not prime no.

# program.

```
class prime
{
public static Void main (string [] args)
{
 int n; i, c=0;
 i = 2;
 while (i<n)
 {
  if (n % i ==0)
  {
   c++;
  }
  i++;
 }
 if (c==0)
  sop ("No is a prime no.");
 else
  System.out.println ("No is not a prime no.");
 }
}
```

# Nested Syntax :-

1) While :-

```
// initialization
while (condition)
{
  // initialization
  while (condition)
  {
```

```
        // statement ;
        // iteration ;
    }
        // iteration ;
}
```

2) Dowhile :-

```
initialization
do
{
initialization
do
{
statement ;
iteration ;
} while (condition);
iteration ;
} while (condition);
```

3) For :-

```
for (initialization ; condition; iteration)
{
    for (initialization ; condition ; iteration)
    {
    statement ;
    }
}
```

# Prime series nested while

```
class prime
{
```

```
public static void main (String [] args)
{
  int i, n=200, c=0, j;
  i=1;
  n=200;
  while (i<=n)
  {
    j=2; c=0;
    while (j<i)
    {
      if (i % j ==0)
      {
        c++;
      }
      j++;
    }
    if (c==0 || c! = 1)
    System.out.println (i);
    i++;
  }
}
}
```

# prime number series in do while.

```
class prime
{
public static void main (string [] args)
{
  int i, n=200, j, c=0
  i=1;
  do
```

```
{
  if (i % j == 0)
  {
    c++;
  }
  j++;
} while (j < i);
 if (c==0 && c!=1)
{
  System.out.println(i);
}
  i++;
} while (i<=n);
}
}
```

# Prime no. series in for.

```
class prime
{
 public static Void main (String [] args)
{
  int i,j,n;
  n=200;
  for (i=1; i<=n; i++)
 {
  int c=0;
  for (j=2; j<i; j++)
 {
   if (i % j ==0)
  {
    c++;
  }
}
```

```
            if (c==0)
            {
              System.out.println(i+ "  ");
            }
        }
    }
```

# pattern :-

a) Size

b) Shape.

3) Contant

* Tips to draw pattern :-

1) First check the size then shape + Content.

2) Outer loop for the row.

3) Inner loop for the Coloumn.

4) For the inner loop do not use println for printing.

5) For the outer loop you must use println for printing in 1 side the next line.

Q)

```
* * * *
* * * *
* * * *
* * * *
```

```
class P
{
public static Void main (string []args)
{
  int i, j;
  for (i= 1; i< =4; i++)  → row
  {
```

```
for (j=1; j<=4; j++) → Coloumn
{
    system.out.print("*");
}
system.out.println();
}
}
}
```

Q)

```
*
* *
* * *
* * * *
```

```
class P
{
  public static Void main (string [] args)
  {
    int i, j;
    for (i=1; i<=4; i++)
    {
      for (j=1; j<=i; j++)
      {
        System.out.print(" * ");
      }
      System.out.println();
    }
  }
}
```

Q. * * * *
* * *
* *
*

```
Class P
{
 public Static Void main (string [] args)
  {
    for (i = 1; i<=4; i++)
   {
    for (j = i; j<=4; j++)
   {
     System.out.print(" * ");
    }
     System.out.println();
    }
  }
}
```

Q. *
* *
* * *
* * * *

```
Class P
{
 public static Void main (String [] args)
  {
    int i, j;
    for (i=1; i<=4; i++)
    {
```

```
for (j=1; j<=4; j++)
{
    if (i==j)
    System.out.print(" * ");
    else
      System.out.print("   ");
    }
    System.out.println();
    }
  }
}
```

Q.

```
  *
* *
* * *
* * * *
```

```
class P
{
public static Void main (String [Jargs)
{
    int i, j;
    for (i=1; i<=4; i++)
    {
     for (j=i, j<=4; j++)
    {
     System.out.print("   ");
    }
    for (j=1; j<=i; j++)
    {
      System.out.print(" * ");
    }
      System.out.println();
```

one space.

```
        }
    }
}
```

# Ternary Operator :-

Syntax :-

Variable = { (condition)? "true" : "false" )}

ex.

WAP enter a no and check it is even or odd.

C -

P -

```
int n = inputing;
SOP (n % 2 == 0) "Even" : "Odd");
```

# Array :-

It is a Collection of Variables of same datatypes

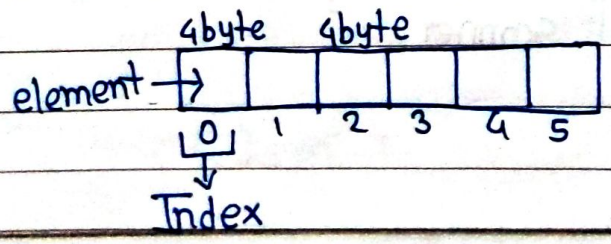
By inputing.

Syntax :-

datatype [ ] Variable = new datatype [size];

ex -

int [ ] a = new int [n];



a.length - It is use for the print the length of array

Q) WAP to print the array and find the length of array using inputting.

→
```
import Java. until. Scanner;
Class length
{
public static Void main (String arrys [] )
{
Scanner sc = new Scanner (system. in);
int a [] = new int [10];
int i
System.out. println ("enter the elements = ");
for (i=0; i<10; i++ )
{
a [i] = sc. next Int ();
}
for (i=0; i<10; i++)
{
system.out.println (a [i]+ "  ");
}
system.out. println (" length of arrays= " +a. length);
}
}
```

Q) WAP take 10 input in array and find the sum.

```
import_ Java. util. Scanner;
Class Sum
{
```

```
public static Void main (string [] args)
{
  Scanner Sc = new Scanner (system. in);
  int a[] = new int [10];
  System.out.println(" Enter the elements ");
  int i, Sum = 0;
  for (i=0; i<10; i++)
  {
    a [i] = Sc. nexInt ();
  }
  for (i=0; i<10; i++)
  {
    Sum = Sum + a [i];
  }
  System. out. println(" sum=" + sum);
  }
}
```

# Merging of two array :-

```
import Java. util. Scanner;
class merge
{
  public Static Void main (string [] args)
  {
    Scanner Sc = new Scanner (system. in);
    int a [] = new int [5];
    int b [] = new int [5];
    int c [] = new int [a.length+b.length];
    for (i=0; i<=n; i++)
    {
      c [i] = a[i];
```

```
    }
    for (i=0; i = b.length ; i++)
    {
      c [i+a.length ] = b [i];
    }
     for (i=0; i = c.length ; i++ )
    {
      System.out.print( c[i] + " ");
    }
  }
}
```

Q) a [] = {10, 20, 30, 40, 50};
b [] = {1, 2, 3, 4, 5};
O/P = {1, 10, 2, 20, 3, 30, 4, 40, 5, 50};

```
Class Array
{
 public static void main (String [] args)
 {
   int a [] = {10, 20, 30, 40, 50}
   int b [] = {1, 2, 3, 4, 5};
   int h = 0, k = b.length -1, i;
   for (i=0; i<a.length; i++)
   {
      c[h] = a [i];
      h++;
      c[h] = b[k];
      h++;
      k--;
   }
   for (i=0; i<c.length ; i++)
```

```
        {
          System.out.print (c[h]);
        }
      }
    }
```

# Frequency :-

```
Class Frequency
{
public static Void main (String []args)
{
 int a[] = {1, 1, 2, 5, 1, 4, 5};
 int b[] = [a.length]
 int i, j, s, lock = -1, c = 0;
 for (i=0; i<a.length; i++)
 {
   c = 1;
 for (j = i+1; j<a.length ; j++)
 {
  if (a[i] = = a[j])
  {
    c++;
   b[j] = lock;
 }
}
  if (b[i] = lock;
}
}
 if(b[i]!= lock)
 b[i] = c;
}
```

```
for (i=0; i<a.length ; i++)
{
  if (b[i]! =lock)
  System.out.print(a[i]+"    " + b[i]);
  }
}
}
```

Q) WAP to find the index of an element in an array.

```
Class index
{
 int a[]= {1,2,3,4,5,8};
 int i, m=4;
 for (i=0; i<a.length; i++)
 {
  if (a[i] ==m)
  {
   system.out.print(i+"    ");
  }
}
}
}
```

Q) WAP to move all zeros to the end of array.

```
Class zeros
{
pubic static void main (String [] args ).
 {
  int a[] = {1,0,1,0,2,1,0};
  int i, j, tem =0;
  for (i= 0; i<a-length ; i++)
```

```
{
 for (j=i+1 ; j<a.length ; j++)
 {
  if ( a[i] < a[j])
  {
     temp = a[i];
     a[i] = a[j];
     a[j] = temp;
  }
 }
}
     System.out.print (a[i] + "    ");
    }
  }
}
```

O/P - {2, 1, 1, 1, 0, 0, 0};

# Insert element in given array.

```
import java.util.Scanner;
class insert.
{
Public Static Void main (String args[])
{
 Scanner sc = new Scanner (system.in);
 system.out.print(" enter the size of array = ");
 int a[]= new int [5+1];
 System.out.print (" enter the elements = ");
 int i
 for (i=0; i<5; i++)
 {
  a[i] = Sc.nexInt ();
 }
```

```
System.out.println (" elements in array - ");
for (i=0; i<s ; i++)
{
 System.out. print (a[i] + "  ");
}
System.out.print (" Enter the Value of position = ");
int P = sc.nextInt ();
System.out.print (" enter the element which want
                    to insert ");
int el = sc. next Int ();
for (i = a.length - 1); i>(p-1); i--)
{
  a [i] = a[i-1];
}
 a [P-1] = el;
for (i=0; i<a.length ; i++)
{
 System.out. print (a[i] + "  ");
}
  }
}
```

Q) Deletion.

```
import java. util. Scanner;
class deletion.
{
 public static Void main (String args [])
{
 Scanner sc= new Scanner (system.in);
 System.out. print (" Enter the size of array ");
 int s= sc.next Int ();
```

```
int a[] = new int [5];
System.out.print("elements in array = ");
int i;
for (i=0; i<5; i++)
{
  a[i] = sc.nextInt();
}
System.out.print(" Elements in array = ");
for (i=0; i<5; i++)
{
  system.out.print (a[i]+" ");
}
System.out.print(" Enter the position = ");
int p = sc.nextInt();
for (i = (p-1); i<a.length; i++)
{
  a[i] = a[i+1];
}
System.out.print("After the deletion element from array =
for (i=0; i<a.length -1; i++)
{
  system.out.print (a[i] + " ");
}
}
}
```

Q) Rotation [anty clockwise].

```
import java.util.Scanner;
Class anty clockwise
{
```

```
Public static Void main ( string [] args)
{
  Scanner sc = new Scanner (system. in );
  System.out.print ("enter the Value of size = ");
  int s = sc.nextInt ();
  int a[] = new int [s];
  System.out.print ("enter the elements = ");
  int i, j, temp = 0;
  for (i = 0; i<a.length ; i++)
  {
    a [i] = sc.nextInt ();
  }
  for (i=0; i<a.length ; i++)
  {
    System.out.print (a [i] + " ");
  }
  System.out.print (" How many time do you want to
                rotate array = ");
  int n = sc.nextInt ();
  for (i=0; i<n ; i++)
  {
    temp = a [0];
    for (j=0 ; j a.length^-1 ; j++ )
    {
      a[j] = a [j+1];
    }
    a [a.length - 1] = temp ;
  }
  System.out.print (" After the rotating array = ");
  for (i=0; i< a.length ; i++)
  {
```

```
system.out.print (a[i] + "  ");
    }
  }
}
```

Q) Rotation (clockwise).

```
import java.util. Scanner;
class clockwise.
{
 public static void main (String [] args)
{
Scanner sc = new Scanner (system. in);
System.out.print(" enter the size of array = ");
 int s = sc.nextInt ();
 int a[] = newint [s];
System. out.print ("enter the elements = ");
 int i, j, temp = 0;
 for (i = 0; i <a.length; i++)
 {
  a [i] = sc.next Int ();
 }
 system.out.print(" How many time to you want to
                    rotate array");
 int n = sc. next Int ();
 for (i=0; i<n; i++)
 {
  temp = a [a.length - 1];
 for (j=a.length -1; j>0; j--)
 {
  a [j] = a [j - 1];
 }
```

```
        a[a] = temp;
      }
    System.out.print("After rotating array = ");
    for (i=0; i<a.length; i++)
    {
      System.out.print(a[i]+" ");
    }
  }
}
```

Q) Finding missing number.

```
      Class missing
      {
    public static Void main (string [] args)
    {
    int a[] = {1,2,3,4,5,7,8,9,10};
    int i, sum=0;
    int ex_el = (a.length + 1);
    int total_sum = (ex_el * (ex_el+1)) /2;
    for (i=0; i< a.length ; i++)
    {
      sum = sum + a[i];
    }
    System.out.print("missing number = "(total_sum - sum));
  }
}
```

Q) Finding maximum number without sorting.

```
      Class max
      {
    public static Void main (string [] args)
```

```
{
  int a[] = {1,2,8,9, 5,7,10};
  int i;
  int max = a[0];
  for (i=0; i<a.length; i++)
  {
    if (a[i] > max)
    {
      max = a[i];
    }
  }
  System.out.print("Maximum number = " + max);
}
}
```

Q) Find minimum number.

```
class min
{
  int a[] = {1,2,3,6,5,4,8,9};
  int min = a[0];
  int i;
  for (i=0; i<a.length; i++)
  {
    if (a[i] < min)
    {
      min = a[i];
    }
  }
  System.out.print("minimum number = " + min);
}
}
```

Q) Shifting 1 to the end without Sorting.

```
class shifting
{
public static Void main (String [] args)
{
 int a[] = {1, 0, 0, 1, 1, 0, 1, 1, 0, 1};
 int i, c=0;
 for (i=0; i<a.length; i++)
 {
  if (a [i] ==0)
 {
   c++;
 }
}
 for (i=0; i<c; i++)
 {
  a [i]=0;
 }
  for (i=c; i<a.length; i++)
  {
   a [i] = 1;
  }
  for (i=0; i<a.length; i++)
  {
  system.out.print(a [i] + " ");
 }
}
}
```

Output. M.

0 0 0 0 1 1 1 1 1

Q) Shifting 1 to the end with Sorting.

```
class shift
{
  public static Void main (string []args)
{
  int a[] = {1,0,2,1,1,0,5,8,6};
  int i, j, temp;
  for (i=0; i<a.length ; i++)
  {
    for (j=i+1 ; j<a.length ; j++)
    {
      if (a[i] == 1.)
      {
        temp = a[i];
        a[i] = a[j];
        a[j] = temp;
      }
  }
  for (i=0; i<a.length; i++)
  {
    System.out.print(a[i]+ "  ");
  }
}
}
```

output :- 0 2 0 5 8 6 1 1 1

Q) Dubplate :-

```
class bubplate
{
public static Void main (string []args)
{
```

```
int a[] = {1, 2, 3, 6, 2, 5, 8, 6, 8, 3};
int n = 9;
int i, j, sum;
for (i=0; i<a.length; i++)
{
  for (j = i+1; j<a.length; i++)
  {
    if (a[i] + a[j] == n)
  {
    System.out.print (a[i]+" + "+ a[j] +" = " +n);
  }
}
}
}
}
```

Q) Triplate

```
class Triplate
{
public static void main (string [] args)
{
int [a] = {1, 2, 4, 5, 9, 8, 7, 11};
int n = 9;
int i, j, k, sum;
for (i=0; i<a.length; i++).
{
  for (j= i+1; i<a.length; j++)
  {
  for (k= j+1; k<a.length; k++)
  {
```

```
if (a[i] + a[j] + a[k] == n)
{
System.out.print (a[i]+"+" + a[j]+"+" + a[k]+"= "+n);
}
}
}
}
}
}
```

Q) Find Palindrome numbers in array.

```
Class palindrome
{
public static Void main (String [] args)
{
int a[] = {121, 151, 555, 334, 263};
int n, i, rem, rev=0
for (i=0; i<a.length; i++)
{
n=a[i];
while (n! =0)
{
rem = n % 10;
rev = (rev * 10) + rem;
n=n/10;
}
if (a[i] == rev)
System.out.print (a[i]+" ");
rev = 0;
}
}
}
```

Q) Find Armstrong numbers in array.

```
class Armstrong
{
 public Static Void main (string [] args)
{
 int a[] = {153, 121, 327, 372, 155};
 int i, n, rem, Sum=0;
 for (i=0; i<a.length; i++)
 {
    n = a[i];
    while (n! = 0)
    {
     rem = n % 10;
     Sum = Sum + rem* rem* rem*;
     n = n /10;
    }
     if (Sum == a[i])
    {
     system.out.print (sum);
     }
    Sum = 0;
    }
 }
}
```

Q) Find prime numbers in array.

```
class prime
{
public static Void main (string [] args)
{
```

```
int a[] = {56, 1, 2, 3, 4, 5};
int i, j, c=0;
for (i=0; i<a.length; i++)
{
  for (j=2; j<a[i]; j++)
  {
    if (a[i] % j == 0)
    {
      c++;
    }
  }
  if ((c==0) && (a[i] != 1))
  {
    System.out.print(a[i]+" ");
  }
  c=0;
}
}
}
```

Imp # Identity Matrix :-

```
import java.util.Scanner;
class identity
{
 public static Void main(String [] args)
 {
  Scanner sc = new Scanner(System.in);
  int a[][] = new int [3][3];
  System.out.print("Eneter the elements");
  int i, j, c=0;
```

```
        for (i=0; i<a.length; i++)
        {
         for (j=0; j<a.length; j++)
        {
         a[i][j] = sc.nextInt();
        }
      }
      system.out.print("element of matrix");
      for (i=0; i<a.length; i++)
      {
       for (j=0; j<a.length; j++)
       {
        System.out.print(a[i][j]+"  ");
       }
      }
      for (i=0; j<a.length; i++)
      {
       if((i==j)) && (a[i][j] != 1))
       {
        c++;
      }
      else  if(i!=j) && (a[i][j] != 0))
      {
       c++;
      }
    }
  }
  if (c==0)
  {
  system.out.print("Identity matrix");
  }
```

```
        else
        {
         System.out.print(" Not identity matrix ");
        }
    }
}
```

Imp# What is Jagged array ?

→ Those array whose columns size are not defined is Called jagged array.

Ex - int [][] a = new int [3][];
a [0] = new int [] {10, 20, 30};
a [1] = new int [] {4, 5};
a [2] = new int [] {5, 6, 7, 8, 9};

ex -

```
Class Jagged
{
 public static Void main (String [] args)
 {
  int [][] a = new int [3][];
  a [0] = new int [] {10, 20, 30};
  a [1] = new int [] {4, 5};
  a [2] = new int [] {6, 7, 8, 9};
  int i, j;
  for (i=0; i<a.length ; i++)
  {
   for (j=0; j < a [i].length ; i++)
   {
    system .out. print (a[i][j]+ "  ");
   }
```

```
        System.out. println ();
        }
    }
}
```

Q) WAP to find the intesection and union of array.

a = {1, 2, 3}; b = {2, 4, 1};

```
Class C1
{
 public static Void main (String [] args)
{
 int a[] = {1,2,3};
 int b[] = {2,4,1};
 int c[] = new int [a.length + b.length];
 int i, j;
 for (i=0; i<a.length; i++)
 {
   c[i] = a[i];
 }
  for (i=0; i<b.length; i++)
 {
   c [a.length + i] = b[i];
 }
System.out.print (" Intersection of array ");
for (i=0; i< c.length; i++)
{
 for (j= i+1; j< c.length; j++)
 {
  if (c[i] == c[j])
  {
```

```
        System.out.print (c[i]+" ");
        }
    }
}
    System.out.print(" Union of array ");
    for (i=0; i<a.length; i++)
    {
      System.out.print ( c[i]+" ");
      }
    }
}
```

Q) Delete duplicate element.

```
Class delete
{
 public static void main ( String c]args)
 {
  int a[]= {1,2,5,4,1};
  int i,j, lock =-1;
  for (i=0; i<a.length; i++)
  {
   for (b=i+1; j<a.length; j++)
   {
    if (a[i] == a[j])
    {
     a[i] = lock;
  }
}
```

# Codenera no. :- 03

## # OOPS :-

Object - Oriented programming Structure.

**Define :-**

It is a paradiagrm that follow java.

## * There are six pillers of oops :-

1) Class
2) Object
3) Inheritance
4) Abstraction
5) Polymorphism
6) Encapsulation.

* If any one is missing in last four Concepts in the any language it is known as object based programming language.

Ex - Java Script.

## 1) Class :-

-> class is Collection of objects.
-> class is not the real world entity.
-> class doesn't aquire memory in JVM.
-> class is blue print of the object.
-> class is the user defined datatype.

**Syntax :-**

```
class class_Name
{
    attribute
```

method
Constructor
block
nested class
}

2) object :-

-> object is the real world entity.
-> object aquire memory in JVM (heap).
-> object has : i) Identity (dog)
ii) State / attribute (colour, breed)
iii) behaviour (run, eat, bark)

Q) How to declare object :-

Syntax : class_Name obj = new class_Name ();

object name keyboard.

// Animal dog = new Animal ();

# Method :-

It is set of Codes that perform some task or operation. There are four types of method :-

1) No return type without argument.
2) Return type with no argument.
3) With return type with argument.
4) No return type with argument.

* What we write inside the parenthesis ( ) it is argument.

Q) How we declear a method:

Syntax :-

```
Access_modifire return type. method_name ()
{
}
```

Ex -

```
public Void main ()
{
}
```

Q) How Can We Call method.

→ Using . (dot) operator with object name we Can call the method.

# Instance Variable :- OR Global Variable :-

Those Variable are declear inside the class.

# Local Variable :-

Those Variable who defined in method or blocks is Called local Variable.

Ex :-

```
Class Animal
{
int a; // Instance / global Variable.
public Void run () // method. (No return type no argument)
{
System.out.print ("Bhag raha hai");
}
```

```
    public static void main (string [] args ) // Main method.
    {
    Animal obj = new Animal ();   // object Creation.
    obj. run ();   // Method Call
    }
}
```

* If we declared a Variable inside the class it is accessible inside the class.

* If we declared Variable inside the any method it is accessible for that method block only.

# Access Modifier :-

It is the set of rules that define how our Variable, methods, Class are accessible.

1] Public :
2] Private :
3] Protected :
4] Default :

# Methods :-

1] No return type without argument :-

Syntax :-

```
Access_modifier return type method_name ()
{
}
```

Ex -

```
Public Void Animal ()
{
}
```

2) Return type with no Argument :-

Syntax :-

```
Access_modifier return type method name ()
{
}
```

ex :-

```
public int Animal ()
{
return 0;
}
```

- In that method the method return data type is always datatype.
  Ex - Int, string, boolean, etc.
- In a method return should be always one's. If it should be in a Condition matter it will return always one's return Value only.

3) No return type With animal :-

Syntax :-

```
Access_modifier return type method name ()
{
}
```

ex :-

```
public Void Animal (int a)
{                          ↳ Arguments

}
```

4) With return type with argument :-

Syntax :-

```
Access_modifier return type method_name ()
{

}
```

ex :-

```
public int Animal (int a)
{
return a;
}
```

Q) WAP by using with return type and with argument method find the given no. is even or odd. if it is even print true else false.

→

```
Class C1
{
public boolean even (int a, int b)
{
if (a % 2 == 0)
return true;
else
```

```
        return false;
    }
    public static Void main (string [] args)
    {
        C1 ob = new C1();
        System.out.print(ob.even(4));
    }
}
// output → True.
```

# Constructor :-

It is a special type of method which calls automatically when object of the class is Created.

# Rules for Constructor :-

→ Class name and Constructor name must be Same.

→ It must be public.

→ It cannot be return type Void and primitive or nonprimitive.

→ Constructor Calls first.

→ One object one Constructor.

Q) when Constructor Called:

→ Each time an object is Creating, using new keyword that Constructor will Call.

# Constructor types :-

1) Default Constructor or Non-parametric

Constructor.

2) Parametric Constructor.

ex :-

```
Class Constructor
{
  Constructor ()
  {
    System.out.print (" Hello " Java ");
  }
  public static Void main (string [] args)
  {
    Constructor ob = new Constructor ();
  }
}
```

* Note :-
  If we write two Constructor in a class it will Call only one at a time. That's why we used this keyword.

# This keyword :-
  This keyword is reference Variable to the Current class object.

* There are three uses of this keyword :-
  1) It refers to Current class instances Variable.
  2) It calls Current class method internally.

  ex :-

```
Class Constructor.
{
  public Void display 1 ()
{
  System.out.print (" Hello M1 ");
}
  public Void display 2 ()
{
  system.out.print (" Hello M2 ");
}
  public Void disp()
{
  system.out.print (" Hello. M");
  this. disp1 ();
  this. disp 2 ();
}
public static Void main (string [] args)
{
  Constructor ob = Constructor ();
  ob. disp ();
}
}
```

3) It Calls Current class default Constructor in tornally in parametric Constructor.

```
class Constructor.
{
 Constructor ()
{
  System.out.print (" default Constructor ");
}
```

```
    Constructor (string name)
    {
      this ();
      System.out.print (name);
    }
    public static Void main (string [] args)
    {
      Constructor ob = new Constructor ("Mango");
    }
}
```

# Inheritance :-

- It make the relation between two class we use (extends).
- Inheritance increases the revesability of Codes.
- It is the relationship between parent Class and child class.

* Types of inheritance :-

1) single level inheritance.
2) Multilevel inheritance.
3) Hybrid / Herictial inheritance.

* Note :-

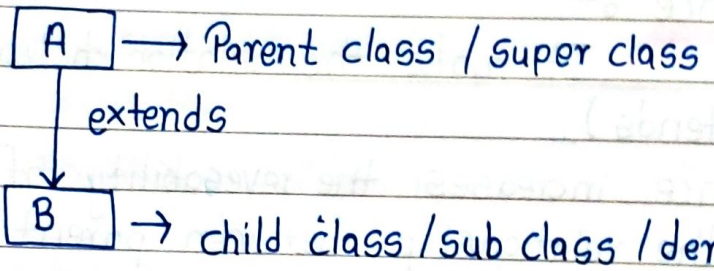
1) To make a relation between two class we used (extend) keyword.
2) which class is extends that class is child class.
3) It means which class is aquire the property it is Child class.
4) That class who extends that is parent class it means

which class is gives our property to another class it is parent class.

5) In one file we can write multiple class but public class should be one in which class has public in that class main method is present only. (Public static Void main (string [] args).

6) We always Create object of a Super child class because Super child has all upper classes parents it will be access all classes methods.

1) Single level Inheritance :-




Ex :-

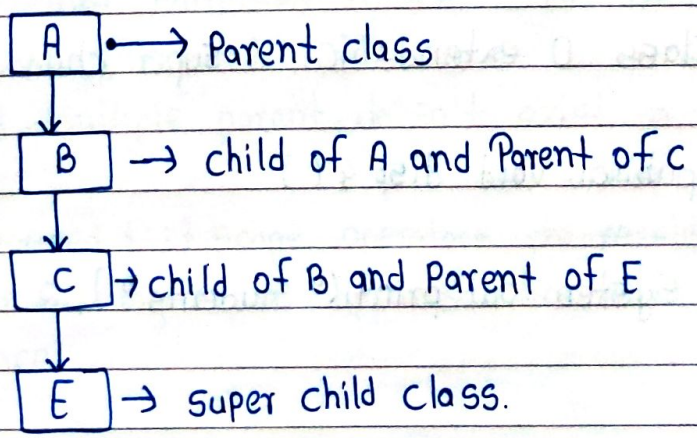
```
Class A
{
  String s = " Mango ";
  public Void disp ()
  {
    System.out.println (" hello ");
  }
}

Class B extends A
{
 public void disp1 ()
 {
```

```
        System.out. println (s);
      }
    }
    public class inheritance
    {
      public static Void main (string [] args).
      {
        B ob = new B ();
        ob. disp ();
        ob.disp1 ();
      }
    }
```

2] Multilevel inheritence :-

- It is a type of ladder.
- It has the single parent class and multiple child Class.
- It is the chain of a classes.




ex :-

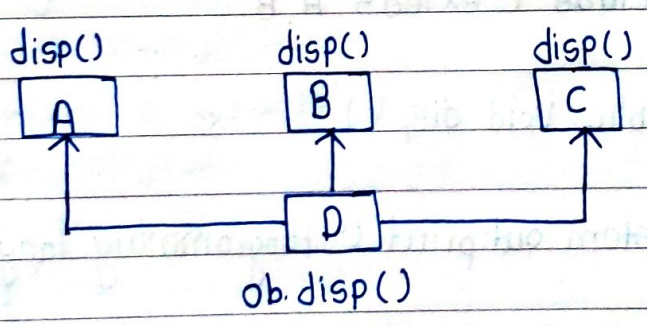
```
Class A  // Parent class.
{
```

```
    public Void disp1 ()
    {
      System.out.print (" Codenera ");
    }
  }
    Class B extends A
    {
      public Void disp 2 ()
      {
        System. out. print (" Abishekh sir ");
      }
  }
    Class C extends B
    {
      public Void disp3 ()
      {
        System.out.print (" Aman sir ");
      }
  }
    Class D extends C // Super child class.
    {
      public Void disp 4 ()
      {
        System.out. print (" Students ");
      }
  }
    public class Multilevel
    {
    public Static Void main (String []args)
    {
```

```
D ob = new D();      // object of super child class
ob.disp1();          // method of class A.
ob.disp2();          // method of class B
ob.disp3();          // method of class C
ob.disp4();          // method of class D
    }
}
```

3] Multiple Inheritance / does not support in Java because of ambiglsity error. :-

- When multiple parent class extends single child class it produce ambigvity error.




// Compiler will confused which class method should be ~~aa~~ call at that situation.

- one child and multiple parent doesn't exist in java.

[ In c++ we used {::} scope operators to resolve the problem that's why c++ supports the multiple inheritance ]

```
Class A
{
public Void disp()
 {
```

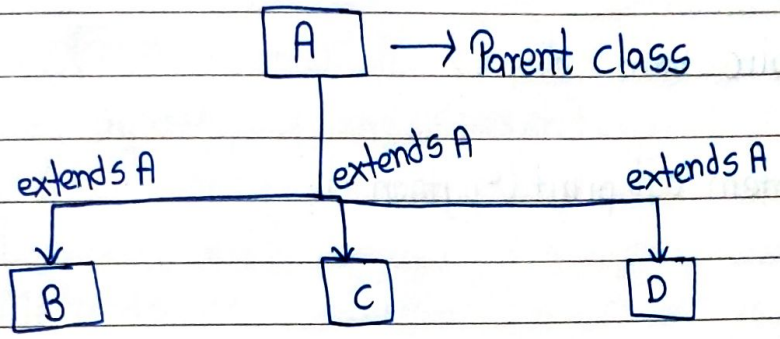
```
        System.out.print("Hello");
        }
    }
    Class B
    {
    public Void disp()
    {
        System.out.print("Java");
    }
}
```

# Embigvity Error :-

```
        Class C exteds, A, B.
        {
        public Void disp()
        {
        System.out.print("Programming language");
        }
    }
    public Multiple Inheritance.
    {
    public static Void main (String [] args)
    {
        C ob = new C();
        ob . disp();
```

4) Hericlrial Inheritance :-

-> One parent class and multiple child class who aquire the properties of parent class to creating the objects of child class.




ex -

```
Class A
{
 public Void disp ()
 {
  system.out.print ("programming language");
 }
}
 Class B extends A
 {
  public Void disp ()
  {
   system. out. print ("Java");
  }
}
 Class C extends A
 {
  public Void disp()
  {
   System. out. print ("C++");
  }
}
 Class D extends A
 {
```

```
    public void disp()
    {
      System.out.print("python");
    }
}
public class Herichical Inheritace
{
  public static Void main (string [] args)
  {
      B ob = new B();
      ob.disp();
      C oc = new C();
      oc.disp();
      D od = new D();
      od.disp();
  }
}
```

* **Super keyword :-**

It refers the parent class inheritance Variable.

Ex - //Variable.

```
Class P
{
  String n = "Mango";
}
class child 1 extends P
{
  String n = "Groupes";
  public Void disp()
```

```
    {
      system.out.print(super.n);
    }
  }
//Methods
   Class A
  {
    public Void disp()
  {
    System.out.print ("Hello Java");
    }
}
   Class B extends A
   {
     public Void disp1 ()
   {
      super.disp();
   System.out.print ("Hello");
   }
}
   public class D
    {
     public static Void main (String []args)
      {
        B ob = new B ();
        ob. disp();
    }
}
   // Constructor :- It call parent class Constructor.
        class A
```

```
{
  A()
  {
   system.out. print ("hello Java");
  }
}
  Class B extends A
  {
    B (int a)
    {
      Super ();
      System.out.print (a);
    }
}
  public class D
{
  public static Void main (String [] args)
   {
    B ob = new B(10);
   }
}
```

# Access Modifier :-

It is a set of rules where we Can accessible our Variables, methods, + class. There are Four types of access modifier :-

1) Public
2) Private
3) Protected
4) Default

# polymorphism :-

- The word polymorphism derived from two greek words poly which means many and morphs which means forms.
- polymorphism is performed by changing the implementation of the method. Can be derived in two parts:
  - i) Static ~~imple~~ polymorphism.
  - ii) Dynamic polymorphisms.

## i) Static Polymorphism :-

Static polymorphism is occurs at the time of Compilation is Called static polymorphism. In static polymorphism, the java Compiler bind the method Calls with method Code at Compile time is Called - time polymorphism.

- It is also achived by method overloading :-

## * Method overloading :-

It is a feature that allows a Class to have more than one method having the same name, if there arguments list are different.

Three ways to overload a method :-

- i) Number of parameter.
- ii) Data type of parameters.
- iii) Sequence of data type of parameters.

- Method overloading is a example of static polymorphism or Compile time polymorphism. Static overloaded method only access through the Class name not from the object - name. class having method overloaded can be inherited.

* Rules for method - overloading :-

1) Class must be same.
2) Method name must be same.
3) Parameters / arguments must be different either the no. of argument or types of arguments. (means, int, double, string, etc).

Ex :-

```
public class poly
{
  public void disp()
  {
    System.out.print("hello Java!");
  }
  public void disp(int a);
  {
    system.out.print("Value of a = " +a);
  }
  public void disp (double a)
  {
    System.out.print("Value of double a = " +a);
  }
  public void disp (string name)
  {
    system.out.print("Value of name =" + name);
  }
  public static void main (string [] args)
  {
    poly ob = new poly();
    ob.disp();
    ob.disp(2.2);
```

```
        ob.disp(5);
        ob.disp("Java");
    }
}
```

ii) Dynamic polymorphism :-

- polymorphism at run-time is Called dynamic polymorphism. It means that when method is Called the method Called is bount to the method body during runtime, or execution time of the of the program.

- The java Comipler is not aware of the method to be invoked during Compilation, now JVM invokes the relevant method during run time. It is achived by runtime polymorphism thorough method overriding.

* Method Overriding :-

- It is a feature that allows a subclass or childclass to provide a specific implementation of method that already provided by one of the super or parent class.

- Java Can achieve a runtime polymorphism by the method overriding.

Ex for understanding.

- Declaring a method in Subclass which is already present in parent Class is Called as method overriding. Overriding is done so that a child class Can give it's own implementation to a method which is allready provided by parent class. In this Case the method in present class is Called Overiden method and the method in child class

is Caller overriding method.

- Private, static and final methods cannot be overriden as they are local to the class.

## * Advantage :-

Method overriding is that the class can give it's own specific implementation to a inherited method without even modify the parent class code.

## * Rules of method overriding :-

i) The method name must same in parent and child class.

ii) The method parameters must same in parent and child class.

iii) Inheritance must be there.

iv) Final keyword is not allowed.

Ex :-

```
Class A
{
  public Void disp ()
  {
    system.out.print ("Hello parent class");
  }
}

Class B extends A
{
  public Void disp ()
  {
    super disp ();      // to see parent class o/P
    System.out.print ("Hello child class");
```

```
        }
    }
    public class poly
    {
        public static void main (* String [] args)
        {
            B ob = new B();
            ob.disp();
        }
    }
```

# Final Keyword :-

- It is in oops Concepts.
- It is used for restiction purpose.
- It make any Value to Constant.
- It is non-access modifier.
- It will be used with Variables, methods and class.

* Uses :-

i) If we used final keyword with Variable then we Can't change it, it remain Constant.

imp ii) If we used final keyword with method we Can't override it.

iii) If we used final keyword with class it Could not extends the any class.

# Abstraction :- [ib will make 0% to 100% secure].

It is property of abstraction in a oops, the only details are displayed to the user, not non-essential data displayed to the user.

- **Abstract class :-**

i) An abstract class and methods that declared with abstract keyword.

ii) It is an type of the class in OOPS that declared one or more abstract methods.

- **Abstract Method :-**

i) A method without a body or defination is known as Abstract method. It can declared in a abstract class.

ii) Abstract method cannot be final.

iii) We cannot make object of a abstract class.

iv) It has parametrized and default constructor.

v) An abstract class may or may not all abstract methods. some of them concreate method also.

* **Advantage :-**

i) Methods are allowed to group several related classes as siblings.

ii) It is also reduce complexity of the design.

Ex :-

```
Abstract Class A
{
    abstract public Void disp ();   // method declaration.
}
Class B extends A
{
    public Void disp ()  // Method define.
    {
    }
```

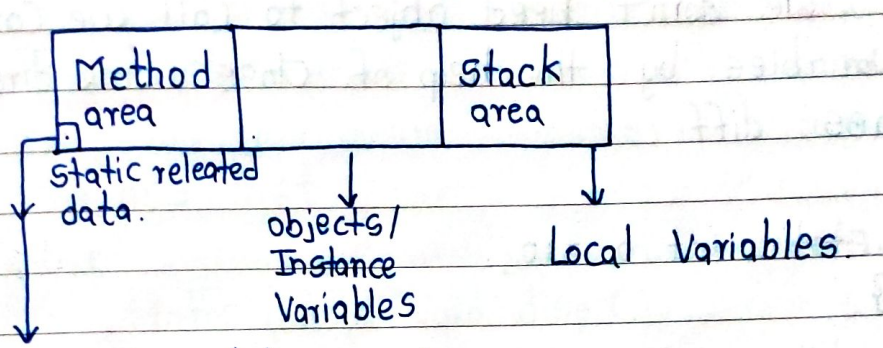
# Static Keyword :-

i) Static is non-access modifier in Java which will applicable on it block.

ii) Variables.

iii) Methods.

- It is used for memory management.




- It is store data equally.
- At the time of class loading instance Variable.
- Refresh at the time of Class loading.

## * Counter Program :-

```
public class main
{
    static int c=0;
    main ()
    {
      c++;
    system.out.print (c);
    }
    public static Void main (String [] args)
  {
    main ob = new main ();
    main ob1 = new main ();
    main ob2 = new main ();
```

```
        main ob3 = new main ();
    }
}
```

- while Creating the object that time it will Call automatically.
- Variable :-

We don't need object to Call we Can ~~u~~ Call the Variables by the help of Class name directly.

```
Class diff
{
    static int a = 10;
}
public class static
{
    public static Void main (string [] args)
    {
        system. out. print (diff. a);
    }
}
```

- **Method** :-

If method is static then we Can Call that method directly using class name. / If it is in Same class also and different Class also.

```
Public Class main
{
    Static public Void disp ()
    {
    system. out. print (" Hello Java ");
```

```
}
public static Void main (String [] args)
{
  system. out. print (Main. disp ());
}
}
```

2) For different class.

```
Class first
{
  Static public Void disp ()
  {
    System. out. print (" Hello Java ");
  }
}
public class static
{
  Static public Void disp1 ();
  {
    System. out. print (" Hello Codenera ");
  }
  public Static Void main (String [] args)
  {
    disp1 ();   // current class using directly.
    first. disp ();   // first class method using class name.
```

* Static block :-

- Always gives priority first then main method.

- 1.6 till we can run the code without using main method but currently not.
- We can create multiple static blocks in one class it will print the given sequences.

Ex :-

```
Class static_block
{
Static scanner sc = new scanner(system.in);
Static int a, b;
Static
{
  System.out.print ("Hello 1");
}
  Static
{
  System.out.print ("Enter the two numbers");
    a = sc.nextInt();
    b = sc.nextInt();
  }
  public static Void main (string [] args)
{
  System.out.print (a + "  " + b);
}
}
```

# Interface :-

It is similar to class but have abstract methods decleration.

OR

It is Collection of methods and Constants.

- It is used to achieve Complete abstraction.

- Every interface in Java is abstract by default so it is not Compulsory to write abstract keyword with an Interface method.

Class → class ⇒ extends / Interface → Interface ⇒ extend
Interface → class ⇒ Implements.

- A class implements any number of interfaces in java.

1) We Can achieve 100% abstraction.
2) Interface provide pure abstraction in java to achieve Is-A relationship.
3) Interface Contains three types of methods :-
   a) Abstract method
   b) Default method
   c) Static method.
4) All the default methods without access modifier.
5) Interface Can not have Constructor.
6) Interface is the only one mechanism that allows achieve multiple inheritance in java.
7) A java class is the only mechanism that any number of interfaces by using Keywords implements.

★ Rules of Interface :-

1) An interface may not be decleared with final keyword.
2) It Can not have instance Variables, {if we decleared Variable in instances it must be initialized}
3) It must be declare with public.

# String :-

Any thing written inside double Code is string.

- It is non-primitive datatype.
- string is also a class.

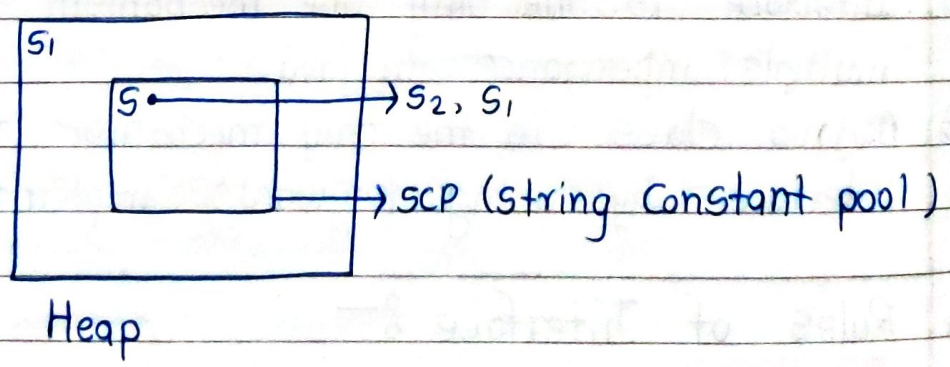
## * String has three implements :-

i) Char sequence.
ii) Serializable.
iii) Comparable.

1) String s = "Apple"; // by literal
2) String s1 = new String ("APPle"); // by new Keyword.

Ex :-

```
String s = "Apple";
String s1 = new String ("Apple");
String s2 = "Apple";
```




- It is ~~te~~ literal that goes into SCP.
- If it is new it goes both.
- So Could not make object when Variable is repeats if should return reference when it value is same.

* String is immutable :-

Because when we declear a value of string it could not change. because string is by default final class.

Ex :-

```
Class String
{
  public static Void main (String [] args)
  {
    String S1 = "Apple";
    System.out.print (S1);
    String S2 = S1.Upper Case ();
    system.out.print (S2);
    System.out.print (S1);
  }
}
```

- when you want new string you have to make object.

# Methods in String :-

1) Char At (0) :-

It takes i/p as a integer and return o/p as a character.

Ex :-

```
String S1 = "Mango";
Char ch = S1.Char At (S1.length()-1);
System.out.print("first Char =" + ch);
```

2] To upper Case (); :- change small into Capital

```
String s = " Shantanu Raut";
System.out.print (s.toUpperCase());
```

3) To Lower Case (); :- change capital into small

```
String s = " SHANTANU RAUT";
System.out.print (s.toLowerCase());
```

4) Trim () :- To remove the outer white space.

```
String sc = "    shantanu    ";
System.out.print (s.trim());
```

5) length () :- Claculate the length of string.

```
String s = " shantanu";
System.out.print (s.length());
```

6) Equals Ignore Case (). :- It return boolean Values / It ignore cases.

```
String s = " shantanu";
String s1 = " SHantanu";
System.out.print (s.equalsIgnoreCase (s2));
```

7) Equals ():- It check equality of two string it return boolean Value.

```
String s = " shantanu " ;
String s1 = " shantanu " ;
```

```
System.out.print (s.equals (s1));
```

8) ~~Compiler~~ Compare to () :- Compare betn two Character on eassy Value. it returns

-ve Value ⇒ first Small then big
+ve Value ⇒ first big then Small
0 Values ⇒ both are Same.

```
String s = "AND";
String s1 = "Boy";
```

s = | A | N | D |

s1 = | B | O | y |

```
System.out.print (s.Compare to (s1));
```

9) Index of () :-
Given Character it will return integer if it is two same character it will take first.

```
String s = "SHANTANU";
System.out.print (s.Index ('A'));
```

10) Last index of () :-
given character will return integer if it is two same character it will take first from the ending side.

```
String s = "SHANTANU";
system.out.print (s.lastindex of ('U'));
```

11) Start with () :-

It check the string where it is start it could return boolean.

```
String s = " Tushar is a very good boy ";
System.out.print (s.Startwith (" Tushar is "));
```

12) end with () :-

It check ending point it will return boolean value.

```
String s = " Tushar is very good boy ";
System.out.print (s.endwith (" boy "));
```

13) Replace () :-

It has two values they must have both is same datatype. Firstoneis old value and second one is to replace it.

```
String s = " Shantanu ";
System.out.print (s.replace ('a', 'A'));
```

14) Replace All () :-

```
String s = " I am good boy ";
System.out.print (" am ", " Am ");
```

15) Contains () :-

```
String s = " Shantanu ";
String s1 = " Shantanu ";
if (s.Contains (s1))
  {
```

== ⇒ It check referance variable
equals ⇒ It check contain.

```
System.out.print ("true");
}
else
{
System.out.print ("False");
}
```

16) Replace () :-

It has two values they must have both is same datatype. Firstone is old value + second one is for place.

```
String s = "shantanu";
System.out.print (s.replace ('a', 'A'));
```

17) Replace All () :-

```
String s = " I am good boy ";
system.out.print (" am", "Am");
```

18) Contains () :-

```
String s = "shantanu";
String s1 = "shantanu";
if (s.Contains (s.)
{
System.out.print ("true");
}
```

```
else
{
 System.out.print(" False");
}
```

19) **substring () :-**

It return string in between range which is given from the user.

s.substring (begining index, end index);

20) **interm () :-**

It chang object difference.

21) **toCharArray() :-**

Convert string in to character array.

22) **getbyte () :-**

It returns string into asscii value.

23) **Value of () :-**

Return boolean whach string is not static method is use class name.

```
int a=10;
int b=20;
Sysoot (a+b;    // 30
String s1 = String.Value of (a);
String s2 = String.Value of (b);
Sop (s1 + s2);   // 1020
```

24) isempty () :-
return boolean ; watch string is not.

~~24~~) character class string :-

25) isletter () :-
return boolean , check character only.

26) isuppercase () :-
check upper case or not return boolean.

27) islowercase () :-
check lowercase or not return boolean.

28) isDigit () :-
check digit return boolean.

29) is space () :-
check space return boolean.

30) is letterdigit () :-
check letter and digit.

# String Buffer :-
1) No need to Create new object for any modifying.
2) Which data is change frequently that store in side the string buffer.
3) while Creating a object of buffer string by default its Capacity is 16.

i) Capacity () → to check Capacity of object.
ii) ensure capacity () → fixed no of capacity from user, fixed a limit.

4) For Concat the stringbuffer it will use append () method.

# Inputing of stringbuffer.

```
{
    string buffer sb = new stringbuffer ();
    System.out.print(" Enter your String ");
    sb.append (sc.nextline ());

    System.out.println (sb);
}

Stringbuffer s = new string buffer ();
        s.ensure Capacity (1000);
        s.append (" Hello word");
        s.trim.to size ();
        System.out.print(s.capacity );    //100
```

| String | String buffer | String builder |
|---|---|---|
| 1) Heap and scp both create object. | 1) Create object in heap. | 1) Create object in heap. |
| 2) It create immutable object. | 2) It create mulable object. | 2) It create mutable object. |
| 3) If we change the value of string lot's of time it will take more memory. | 3) Consume less memory. | 3) occupy less memory. |
| 4) Not thread - safe | 4) All methods are synchronise and threat safe. | 4) Not synchronised method not thread safe. |
| 5) Speed is slow. | 5) Fast as compare string. | 5) Fast as compare stringbuffer. |
| 6) It place where data is not changing frequently. | 6) It place where we need to change data frequently. | 6) It place where we need to change data frequently. |

# Anagram :-

It will the Comparision between two string. All characters are present in other string or not.

```
Class Anagram
{
  public static Void main (string [] args)
  {
    String s = "SHANTANU";
    string s1 = "HATANUS";
    s = s. to lowercase ();
    s1 = s.to lowercase ();
    if (s. length () ! = s1. length ())
    {
      System.out.print(" Both string are not anagram");
    }
    else
    {
      char ch1 [] = s. to charArray ();
      char ch2 [] = s1. to charArray ();
      Array. Sort (ch1);
      Array. Sort (ch2);
      if (Arrays. equals (ch1, ch2) = = True)
      {
        System.out.print(" Anagram");
      }
      else
      {
      System.out.print(" Not Anagram");
```

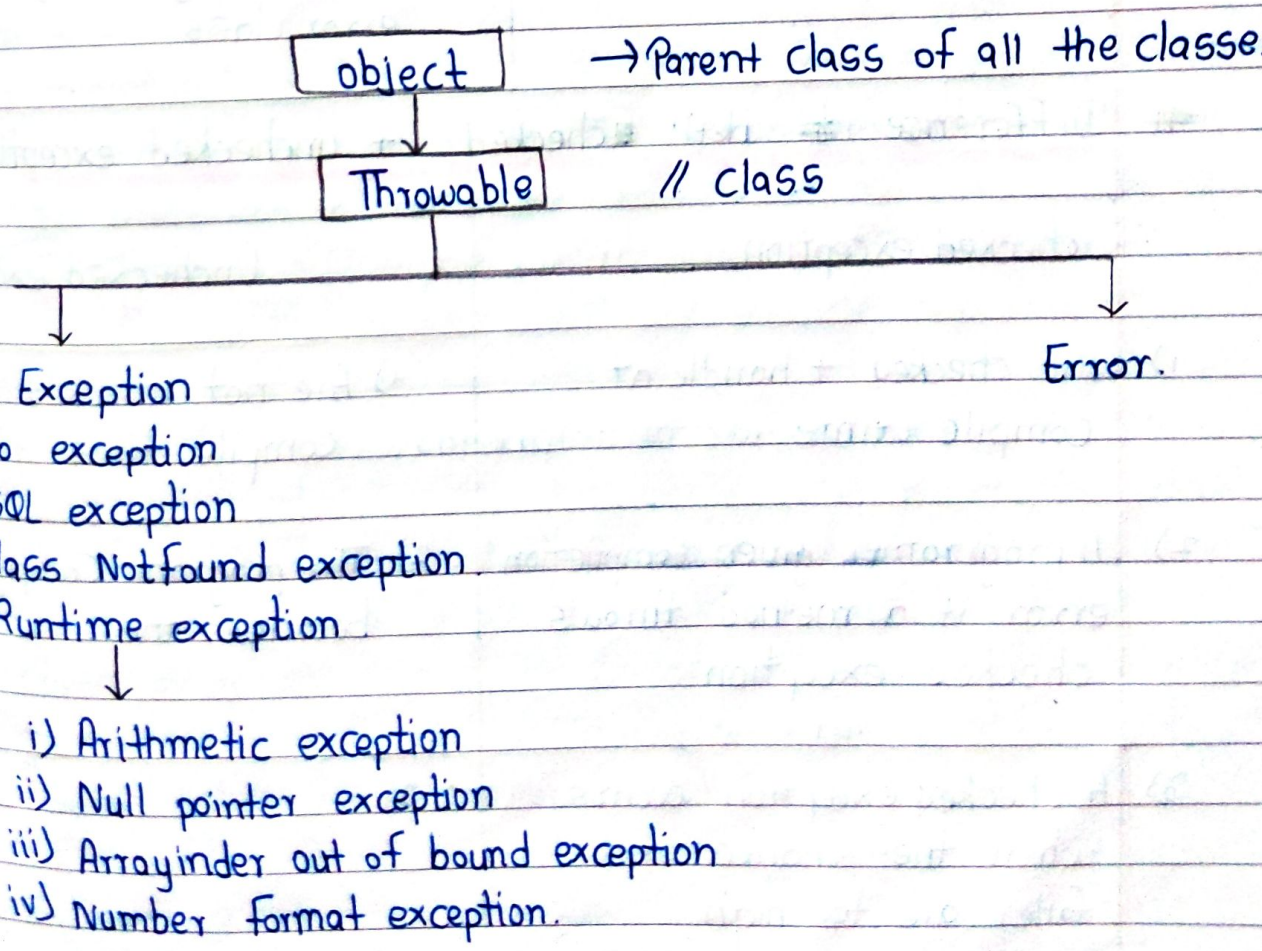
```
        }
    }
}
```

# Exception Handaling :-

An exception is an unwanted interpection which occurs during the execution of a program is at runtime that disturb the normal flow of program.

## Exception Handling :-

Exception handling is mechanism to handle runtime error such as class not found exception.



checked exception → Compile time
unchecked exception → run time.

# Difference between exception and error.

| Exception | Error. |
|---|---|
| 1) Occured by our program. | 1) Occured because of lock of system resources + not by one programs. |
| 2) Recoversible | 2) not recoversible has to deel with system |
| 3) Two types :- checked exception unchecked exception. | 3) onetype - runtime (unchecked error belongs to java. long. error class. |

# Difference ~~bt~~ betn checked + unchecked exception.

| checked exception | Unchecked exception. |
|---|---|
| 1) Are checked + handle at Compile time. | 1) Are not checked at Compile time. |
| 2) A programer given Compiletion error if a method throws checked exception. | 2) The program Compiles fine because the |
| 3) A checked exception occurs when the chance of failer are too high. | 3) |

# Exception Handling :-

1) Try block :-

- In try block the write statements where exception is occured. it maintain risky code.
- Try block associated with Catch, finally blocks.

2) Catch block :-

- Catch block maintains exception handling Code i.e. alternative way for exception.
- It executes implicity by matching occured exception object.
- we can provide multiple Catch block in single catch block.
- we maintain or handle multiple exception in single catch block by using bitwise, operator or operator.

3) Finally block :-

- It is the block to execute important statement like closing statement.
- It dosen't matter exception occured or not.
- It is associated with try block, we write one finally with one try block.

```
Class exception
{
  public static Void main(string []args)
  {
    int a=10;
    int b=0;
```

```
try
{
  System.out.print (a/b);
}
catch (Exception e)
{
  System.out.print (" please enter voild number " +e)
}
finally
{
  System.out.print (" Thank you for visit... ! ")
}
}
}
```

# Exception propogation :-

If exception is occured at the multiple time and if we handle at a single position it will resolve.

Ex :-

```
Class exception_propogation
{
  public Void disp ()
  {
    int a = 10;
    int b = 0;
    System.out.print (a/b);
  }
  public Void disp1 ()
  {
```

```
try
{
 disp();
 catch(exception e)
 {
 System.out.print(" change the number " +e)
 }
}
public.staticvoid main(string []args)
{
 Exception_propogation ob = new Exception_propogation();
   Ob.disp();
 }
}
```

# Throw :-

- In java throw is a keyword which is used for exception handling.
- It is used to throw exception object.
- It is used with checked and unchecked exception.
- Using throw keyword we can throw one exception object at a time.
- Throw keyword can be used inside a method.

ex :-

```
import.java.util.scanner;
class exceptiongenerate extends runtime exception.
{
 exceptiongenerate ()
 {
```

```
    System.out.print("you are not eligible");
  }
}
public class throw_example
{
Scanner sc= new Scanner (system.in);
public void disp ()
{
  system.out.print ("enter your age");
  int a = sc.nextInt();
  if (a<18)
  {
  throw new exception generate ();
  }
  else
  {
  System.out.print (" Congrats you are eligible ");
  }
  public static void main (string [] args)
  {
  throw.example ob = new throw_example ();
  try
  {
    ob.disp ();
  }
  catch (exception generate c)
  {
    system.out.print (e);
  }
}
}
```

# Throws :-

1) Throws keyword is used to declare exception object.

- It used with checked exception.
- It is used with declared method
- It is used with method.
- we can declare one or multiple exception object asper our requirement.

Imp worked on checked exception where it generate it throw the exception and where we call the exception at that point it will hangle.

File input strim → to open file.

```
public class throws_example
{
 public Void disp() throws File Not found exception
  {
   File Inputstrem f= new fileInputstrem ("d :/adetxt");
  }
   public static Void main (string []args)
  {
   throws_example ob= new throws_example ();
   try
   {
    ob.disp();
   }
   Catch (exception e).
   {
    System.out.print(" Sorry file not found " +e);
```

```
    }
  }
}
```

| Throw | Throws |
|---|---|
| 1) Throw is a keyword is used to throw exception object. | 1) Throw is a keyword which is used for exception handling. |
| 2) used with checked and unlocked exception. | 2) used with clocked exception. |
| 3) It is used for enduser. | 3) It is used for developer. |
| 4) using throw keyword we can declared only one exception at a time. | 4) using throw keyword we can declared multiple exception at a time. |