* How to create a Java Program ?

```
Void display ()
{
System out print \n (" The student no. II. No is
  " + SRL No)
import java.lang.*;
Class Class name
{
 Global variable
 own Method
 main method
}
import java.lang.*
Class Cw max
{
  int SRL No;
 String S. Name
 Float S. marks
 String S Address;
```

* Modifier :-

Public static void main :-

how to complie ?

Java C. file name Java

how to save this application ?.

file → save as → file name.Java

file name same as the class name

In case of any errors comeback the application verify.

In case of not errors save the Java file name.

* Write a Java program to create employe details with following data.

1. empbye name
2. employe ID
3. empbye salary
4. employe address

Employe designation :-

Program :-

```
Class priya
{
 string &Name;
 int &ID ;
 float &salary
 string & Add,EDes;
 void data ( )
{
 &Name = "priya"
 &ID = 2347;
```

```
    E salary = 27549.50f
    E Add = "Gandhi Nagar"
    E DES = "Ast. Manager"
  }
  Void display E)
  {
   system.out.println("Name : "+E Name);
   system.out.println(" ID : "+E ID);
  system.out.println(" salary : "+E salary);
 system.out.print("address :"+E Add);
 system.out.print(" Designation :"+E Des);
 }
  public static void main (string args[])
  {
    Priya p= new priya();
    P.data();
    P.display();
  }
}
```

Java Characteristics (or) Buzz words

* Java is simple. any user wants to creating of application using simple coding instructions provided by Java software.

Java language provides user friendly classes objects, methods, packages

* Java is multiplated language.

Thread is a single sequential flow of control with in application.

* Multiplating means any program (or) application using more than one single sequential flow of control with in application.

* Inheritance ;-

Inheritance means acquiring the properties from one class to another class.

Ex ;-

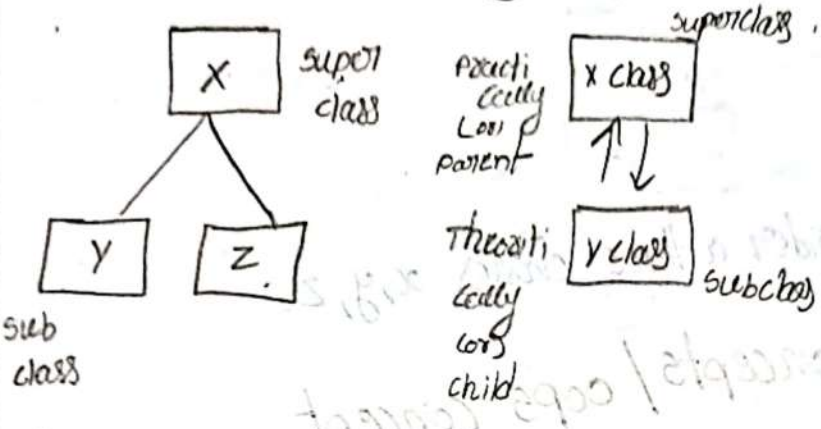
super class to sub class

Inheritance are divided into 3 types

1. Single (or) single level inheritance
2. Multilevel inheritance.
3. Multiple inheritance.

1. Single inhesitence exchanging :~

Any amount of informalion forom supor class to sub class using extends keyboard

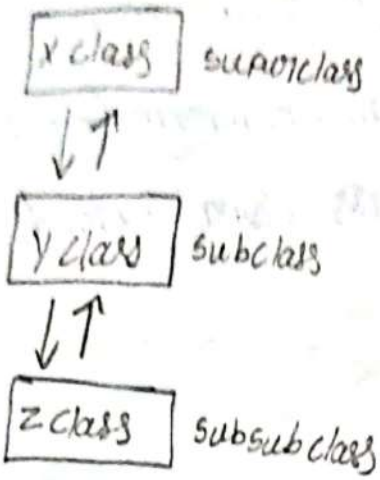


Program ;~

```
class x
{
 ≡≡  ≡≡
};  };
classy exceeds x
{
```

2. Multilevel inheritance ;~

exchanges of any amount of information from 1 class to different classes (or) more than 1 class using extends keyword.



consider a three classes x, y, z.

Basic concepts / oops concept

Java is a widely-used, high-level, object oriented programming language known for it's portability performance and scalability. Here are the basic concepts in Java:

1. Class
2. Object
3. Abstraction
4. Encapsulation
5. Polymorphism
6. Message
7. Inhexitance
8. Multi-Threading.

1. Class :-

Class is a blue print (or) Class is a collection of objects.

Class is a keyword

Syntax :-

```
class classname
{
  Global variable;
  Own method;
  Main Method;
}
```

* Write a Java program to represent class

Sol

```
Class student {
    int id;
    String name;
    public static void (String args[])
    {
      student s1 = new student();
      System.out.println(s1.id);
      System.out.println(s1.name);}
    }
```

2 Object :-

object is a instance of a class. object means anything in the real world. An object is always maintain different attributes. Attribute means property of an object is called attribute.

Ex :- car, cycle.

* Write a Java program to create an object?

Ans

```
class Lamp.
{
  boolean is on;
  void turn on()
  {
    is on = true;
    system.out.println("light on ?" + is on);
  }
  void turn off()
  {
    is on = fals;
    system.out.println("light on ?" + is on);
  }
}
class main
{
  public static void main(string[] args)
```

```
{
public static void main(string[] args)
 {
  Lamp led = new lamp();
  Lamp halogen = new Lamp();
  led.turnOn();
  halogen.turnoff();
 }
}
```

According to Java language attribute means Variable.
Each and every object are created using new operation
syntax ; Car c = new Car();
Different methods are associated to objects using (.)
operator.
Example ;-

```
import java.lang.*;
class Hello
Void display()
{
System.out.println("Hello world");
}
Main
{
Hello newHello();
H.display();
}
```

3 Abstraction ;-

Hiding on complexity information to the users

Ex ;- Abstract class, Abstract window.

4. Encapsulation ;-

encapsulation means combining of variables and methods.

Ex ;-

Methods with parameters.

5. polymorphism ;-

* poly means Many and Morphism means forms.
* According to the polymorphism means different object using same Methods.

Ex ;- aeroplane fly in the sky, birds fly in the sky. same Method - fly, object aeroplane, birds.

6 Message :-

To exchanging of any amount of information from one object to another object.

7. Inheritance ;-

* Inheritance means acquiring properties to one class to another class.

* Acquiring the properties from one class to another class by using extends keywords.

Types of inheritance ;-

1. single level
2. multi level
3. Hybrid level

8. Multi Threading ;-

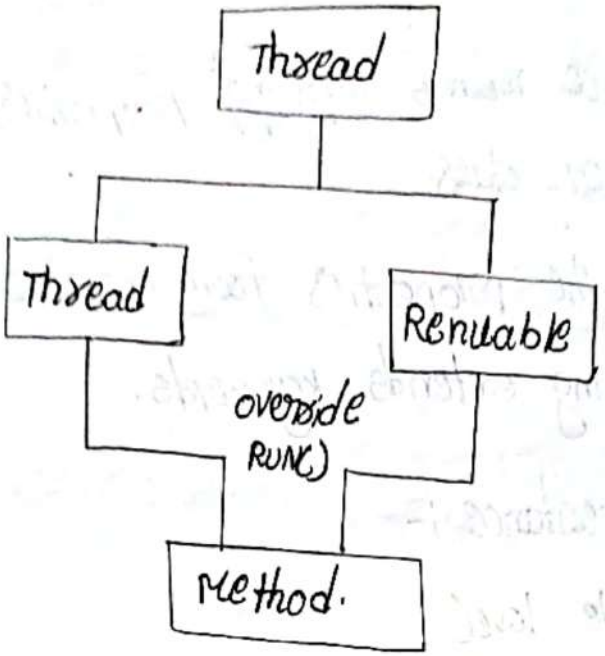
* Threads means single sequential flow of control with application.

* Any application is used more than one single sequential flow of control is called multi threading.

* Java supported in built multi concept.

user thread are created using 2 different types.

1. Extends thread
2. Implement Runnable.



ELEMENTS OR TOKENS IN JAVA PROGRAM

1. Keywords :~ Elements ;- Tokens in Java are 5 types they are

Key these are reserved words in Java that have a specific meaning to the compiler. For example, key words class, public, and static are used to declare a class, and the keywords if, else, and while are used to control the flow of a program.

2. IDENTIFIERS ;~

These are names given to variables, methods, classes and packages, Identifiers must start with latter, an underscore (_), or a dollar sign ($) and can be followed by any Numbers of letters, digits,

underscore, or dollor signs.

3. CONSTANTS :~

Constants are also like normal variables. But the only difference is, their values cannot be modified by the program once they are defined. Constants refer to fixed values. They are also called as literals.

4.B SPECIAL OPERATORS :

BRACKETS [] : opening and closing brackets are used as element reference.
These indicate single and Multidimensional subscripts

PARANTHESIS () : These special symbols are used to indicate function calls and function parameters.

BRACES {} : These opening and ending curly braces marks the start and end of a block of code containing more than one executable statement.

COMMA (,) : It is used to seperate more than one statement like for separating parameters in function calls.

SEMICOLON: It is an operator that essentially invokes something called an initializtion list.

ASTERLCK(*): It is used to create pointer variable.

ASSIGNMENT OPERATOR: It is used to assign values.

5. OPERATORS :-

operators type	operators	operators Description
Arithmetic operator	+, -, *, /, %	Addition substraction, Multiplication, Division, Modulo Division.
Relational operator	==, >, >=, <, <=, !=	Equal to, greater than, greater than or equal to, less than, less than or equal to, Not equal to.
unary operator	++, --, &, size of()	Increment, Decrement Address, size of operator.
Shif operator	<<, >>, >>>	left shift, right shift, unsigned shift.

Assignment operator	+=, -=, *=, /=, %=	Addition assignment subtraction assignment, Multiplication Assignment, Division Assigment, Modulus Assignment.
Ternary operator	condition ? expression 1 : expression 2	condition : true or false exp1 : true, exp2 : false
Logical operator	&&, ||, !	AND, OR, NOT
Bitwise operator	&, |, ^, ~, <<, >>	Bitwise (AND, OR, XOR, Compliment shift left, shift right).

* Write a java program on operator ?

Sol

```
Public class operator examples
{
  public static void main (string [] args)
  {
    //Arthmetic operator example : Addition
    int num 1 = 10 ;
    int num 2 = 5 ;
    int sum = num 1 + num 2 ;
    System. out. print ln ("sum of" + num 1 + "and
    + num 2 + " is :" + sum);
    //Relational operator example greater than
    boolean is greater = n
```

JAVA STATEMENTS :~

Statement is an executable instruction that tells the compiler what to perform. It forms a complete command to be executed and can include one or more expressions

Control statements :~

1. if else
2. switch
3. while
4. do while
5. for
6. break
7. continue
8. return
9. labelled break, continue.

1) if else :-

syntax :-

```
If (condition statement) {
  statement to be executed if condition becomes
}
else
{
  statement to be executed if the above condition
becomes false
}
```

2. switch :-

syntax :

```
switch (bytes/short/int)
{
  case expression:
     statements
  case expression:
     statements
  default:
     statements
}
```

3. while loop :-

syntax :

```
while (condition - statement -> true)
```

{
statement to be executed when the condition becomes true and execube them repeatedly until condition becomes false
}

Example :

```
int x=2;
while (x>5)
{
system.out.Print(n(" value of x:"+x);
x++;
}
```

4. do - while :-

syntax :-

```
do
{
statement block
}
while (condition);
```

5 for loop :

→ for (initialization; condition ; increment / decrement)

{
statement to be executed until the condition becomes false
}

Example :

```
for (int x=0; x<10; x++)
{
 system.out.println("value of x:"+x);
}
```

6. Break :-

→ Break is used in the loops and to skip the current executed and continues with the next iteration.

Example :

```
for (int i=0; i<50; i++)
{
 if (i%13 ==0)
 {
  continue;
 }
 system.out.println("value of i:"+i);
}
```

7. Continue :-

→ Continue makes the loop to skip the current execution and continue with the next iteration.

Example :

```
for(int i=0; i<50; i++)
 if(i%13==0){
 continue;
 }
 System.out.println("value of i",+i);
 }
```

8 Return

Return statement can be used to cause execution to branch back to the caller of the method.

9. Labelled break; continue :

Labelled break and continue statement will break or continue from the loop that is mentioned used in nested loops.

JAVA DATA TYPE :-

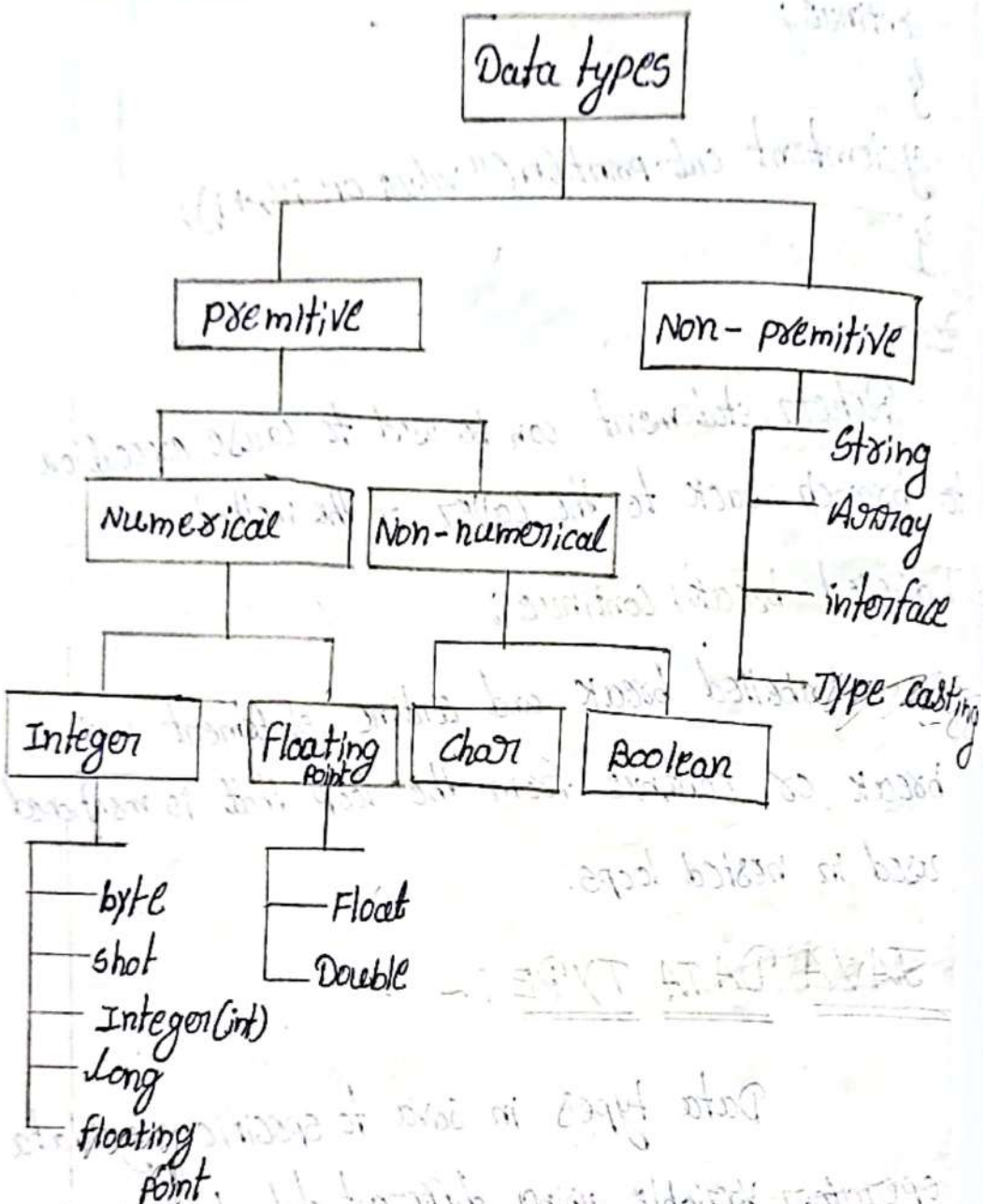
Data types in java to specific any data operation variable using different data types are in two types.

1. premitive data types
2. Non-premitive data types.

JAVA DATA TYPES

1. Premitive

* Flowchart :-



Data types in java to specific any data operator Variable using different data types are in two types:

1. premitive data type
2. Non-premitive data type

1 premitive data type :-

premitive data types are divided into two types.

1. Numerical data type
2. Non-numerical data type

1. Numerical data types are again divided into two types. They are :

1. integer type
2. floating point type.

2. Non-numerical data types are again divided into two types. They are :

1. Char
2. Boolean

* integer is again divided into 4 types.

1. Byte : It is maintains 1 byte size are 8 bits.
2. Shot : It maintains 2 byte size are 11 bits.
3. integer (int) : It maintains 32 bits long ~~# 64bits~~
4. long : It maintains 64 bits.

2 - Mon
3 - Tue
4 - Wedne
5 - Thur
6 - Fridar
7 - Saturry
8 - Sudax
9 - Mon
10 - Tue
11 - wed
12 - Thurs
13 - Fri
14 - Sat
15 - Sun

5. floating point are 2 types

1. float - It is maintained 4 bytes 32 bits

2. Double float - It is maintained 64 bits.

* integer :-

* A byte type variable takes 1 byte of storage and can store from -128 to 127.

* short data type is 16 bit. It has -32,768 max 32,74

* int data type 32 - bits.

* ilong data type 64 - bits.

* Floating points :-

Floating points are 2 types

Float :

A float gives you approximately 15 - 16 decimal digits

* Needs @ 4 bytes of storage

Ex : float f = 36.8929929;

Double :

A double gives you approximately 15 - 16 decimal digits.

needs a 8 bytes of storage

Ex :- double d = 4.58877;

* Character :-

UTF - unicode text format

It supports 84,536 characters, 16 bit uni-code character. A char is a single character.

Ex : char grade = 'A';

* Boolean :-

A boolean type is declared with boolean keyword and only takes the values true or false.

Ex :- Boolean is salad Healthy = true;

Boolean is salad Tasty = false;

2. Non-premitive :-

Non-premitive data types are divided into 4 types they are :

1. String
2. Arrays
3. interface
4. Type casting

* String :-

A string is collection of characters. Java strings is classified into two types

1. Mutable (can be altered 2. Immutable (cannot be altered).

* Interface :-

An Interface in java is used to specify the contract or capability that implementing classes.

Ex ;-

```
public interfesence Area {
public void compute (float x);
final float pi = 3.14 f;}
```

* Type casting :-

* Type casting is the process to converting one data type variable.

* Type casting is again divided is again divided into two types :

1. Implicit
2. Explicit

*In general type casting supported terminal.

* Any user converted higher data variable to lower data variables. these lose of data is called narrow conversion.

syntax ;-

variable name = Data type (variable name)

Ex : int i;
float a;
i = (int)a;

STRING BUFFER METHODS

* string buffer can be immutable. Immutable means it can be altered any given strings.

* Approx string Buffer maintains 3 constructors and methods.

1. sb = new string Buffer ();
2. sb_1 = new string Buffer int index); increasing buffer capacity.
3. sb_2 = new string Buffer (string str);

The string buffer methods are of;

1. length()	5. insert()
2. capacity()	6. delete()
3. reverse()	7. charAt()
4. append()	8. equal()

1. length ();

It is used to finding the given string length.

2. Capacity ();

It is used to finding the capacity of given string capacity.

3. reverse ();

It is used to convert given string

4. append ();

It is used to adding of newstring to old str

4. append();

It is used to adding of newstring to old str

5. Insert();

It is used to adding insert new string.

6. delete();

It is used to delete new string.

7. Char At();

It is used to find out individual character from the given string.

8. equals():

1. write a java program to implement a set of give string.

Ans

```
import java.lang.*;
Class string buffer
{
 string buffer sb1 = new string buffer("Madam");
 string buffer sb2 = new string buffer("welcome");
 string buffer sb3 = new string buffer("To java lab");
{
 system.out.println("first string length is"+s1.length());
 system.out.println("second string length is"+s2.length());
```

```
system.out.println("third string length is"+s3.length());
system.out.println("first string reverse is"+s1.reverse());
system.out.println("Append of third string"+s3.append
                    ("is a programming language");
}
}
```

2. write a java program to implement lower, upper cases and charAt method of a given string.

```
import.java.lang.*;
class string
{
public static void main(string args[])
{
  string s1=new string("Hello");
  string s2=new string("world");
  system.out.println("lower case":+s2.to lower case);
  system.out.println("upper case:+s1.to upper case);
  sysem.out.println("charat 3:"+s2.charat(3));
}
}
```

3. write a java program to implement type casting of a given data type variable.

Sol:

```
import java.lang*;
class type casting
{
  int D=13;
  float f;
```

```
public static void main (String args[])
{
  D = (int) f;
  f = D;
  System.out.println ("F:" + f);
 }
}
```

UNIT-2

* object is a anything in the real world
  Ex :- cycle, car

* Java language soported each & every thing objects.
objects are created with help of new operator.

Ex :- cycle = new cycle();
(operator: new; object: cycle; constructor: cycle())

* Class : Class is a blue print of object
  * Class is collection variable & methods
  * Class is always act as a xerox copy of the object.

structure of the class

```
Package package name;
 class class name
{
 Global variables;
 Methods;
 main method.
{
} // close the main method
} // close the class
```

Write a Java program with the student details.

Student name
Student id
Student address
Student marks.

Program :-

```
import java.lang.*;
class Janu
{
String sNam;
int s id;
String sAddress;
Float marks;
}
void set data()
{
Sname = Janu;
sid = 2519;
s Address = 1+N·Purm;
s marks = 580;
}
void display()
{
System.out.print In("the student name
```

```
is " + sname );
}
Public static void main (String arg [])
{
    S. Janu = new Janu ;
    S. set data ();
    S. display ();
    S. Student = new student
    S. Set data ();
    S display student details ();
}
}
```

Class Decleration ;~

Global Variables ;~

* Global variables are defined under the class. Global Variables are used any where of the class methods. Constructors, main method of the java.

Ex ;~

Global variables are different types.

1. string student name ;
2. int Student Roll Name ;
3. float student Address ;
4.

local variables :-

local variables defined with in the body of the method or function / constructor.

Ex :-

```
Return type method name (parameter list)
{
 int n;
```

① write a java program to create student details in the following data. ~~student~~

student name
student Avg marks.
student ID
student Address.

② write a java program to create a employe details.

Employe name
Employe ID
Employe salary
Employe Add
Employe Dej

③ write a java program to create ~~a employ~~ string & string buffer methods.

① Ans

```
import.java.lang.*;
class student
{
 string sName;
 float s Avg marks;
 int s ID;
 string s address;
 void set data() student()
 {
  S name = Bhuvana
  S Avg marks = 9.5 f
  S ID = 232U1A3740
  S address = Kurchaala padu
 }
  void display()
  {
  system.out.println(" name of the student is "+s.Name);
  system.out.println("student Avg marks is "+ s mark);
  system.out.println("student ID is "+s.ID);
  system.out.println("student address is "+s address);
  }
  public static void main (string argg[])
  {
  student s=New student;
        s.set data();
```

```
        s.display();
    }
}
```

(2)

```
import java.lang.*;
class E name
{
  String E Name;
  int E ID;
  float E salary;
  String E Add;
 String E Des;
 Void data()
 {
   E Name = "Bhuvana"
   F ID   = 28201:A3740
  F Salary = 30000.F
  E Add =  Kuzchorlapadu
 E Des  = software
 }
  Void display()
 {
   System.out.println("Name :" + E Name);
  System.out.println("ID:" + E ID);
System.out.println("salary:" + E salary);
```

```
system.out.println("address:"+E.Add);
system.out.println("Designation:"+E.des);
}
public static void main (string arg[])
{
  Employe E = new employe();
        E = set data.();
        E = display();
    }
}
```

3)
A

```
import java.lang.*;
Class string methods
{
 public static void main (string[] args)
 {
  string str = "Hello world"
  int length = str.length();
  system.out.println("length of the string is" +
                                    length);
  system.out.println("upper case is" +str upper
                                    case());
  system.out.println("lower cas is"+str to lower
                                    case2());
  }
}
```

Text editor - program
Save desktop command
Terminal cd documents
Javac filename.java
Java filename.

* **Method :-**

Method is a collection of variables and instructions.

Method syntax :-

< Modifier >< Access specifier >< returntype >< Method Name >< parameter list >

* Modifier means given additional meaning to the class or method
* specifier means given additional meaning to the class or variable or method.
* Method name consider a depends on the consider a any name.
* Method name consider any name depends on the user / developer.
* parameter consider as a parameter depends on the user

Ex :- public static void main (String [] args)

public → keyword or modifier
static → Access modifier
void → return type
main → collection of parameter arguments
String [] args → args converted into string type

Public is a keyword

Static is a Access specifier

Method signature is a collection of return type, method name and parameter list / arguments.

Ex :-

Creating of a Method.

```
void setdata(int a, int b)
{
  Student name = 'x'
  Student ID = 1234
  student address = "Gist"
}
```

* Method overloading :-

* Method overloading means changing the functionality of one method to another method.
* It depends on different conditions that are.
*
1. 2 Method ~~or~~ name are same Method name.
2. Number of parameters
3. order of parameter
4. Types of parameter

Ex :-

void display(int a

void display(int a)
void display(int a, int b)
void display(int a, int b)

void display(int a, int
void display(int b, int a)
void display(str, a, str, b

Program :-

```
import. Java. lang. *
void class Add
{
 void add (int a+ int b + int c)
 }
 void Add (int b+ int c)
 {
 ======
 }
import. jav. lang. *;
class Multiply
void Multiply (int a * int b)
 {
   }
   void Multiply (int b * int a)
```

constructor ;-

1. constructor is a hardware machanism.
2. Hardware machanism is taken care of each and every thing of the constructor, invoking, relocation, objects (memory allocation, deallocation, objects).
3. constructor is created same as the class name.
4. constructors are not required no return type.
5. constructors are required modifiers (sum times)
6. constructors are supported memory allocations
7. constructors are created using new operator.

Ex ;- ~~Class~~ Sai s = new Sai();

operator ↑ constructor

8. constructors are modified with help of using methods.
9. constructors are supported delete mechanism. (garbage mechanism).

Ex ;- Write a Java program to create student details using constructor.

Program :-

```
import java.lang.*;
class class student
{
  String s name;
  int s ID ;
  int s Avg marks;
  String s Address;

  student ()
  {
   s name = "Kumari";
   s ID = 1901;
   s Avg marks = 65;
   s Address = "GIST";
  }
  void display ();
  {
   System.out.println ("The student name
                        +s name);
   System.out.println (" The student ID is"
                        + s ID);
   System.Out.println ("The student Avg mar
                        is" + s Avg mar
```

(Global variables)

system.out.println(" the student address" + s
address);

}

public static void main (string args[])
student s = new student()
s.display();
s.set
}
}

② Write a java program to create employe details using constructor E name, E ID, E salary, E Address, E Designation.

Sol

import java.lang.*;
Class Emboye
{
string E name;
int E ID ;
string E salary;
string E Address ;
string E Designation;
Employe();

```
{
  E Name = "Bhuvana";
  E ID   = 232U1A0501
  E Salary = 40000;
  E Address = "Kurichedlapadu"
  E des  = "software"
}
Void display()
{
system.out.println("The Empbye Name is" + E Name);
system.out.println("The employe ID is" + E ID);
system.out.println("The Empbye salary is" + E Sal);
system.out.println("The Empbye Address is" + E Add);
system.out.println("The employe Des is" + E Des);
}
Public static Void main(String args[])
{
  Empolye E = new employe();
  E.display();
}
}
```

This keyword

* This Key word is used to differentiating
* This Key word is always refferring of current object.
* The key word Applied on different variables using (.) operator.

Ex :- write a java program to create student details using this keyword.

```
import.java.lang.*;
class student
{
  string sname;
  int SRL No;
  int smarks;
  Void set data (string sn, int SRL, int sm)
  {
    This sname = sn;
    This SRL NO = SRL;
    This smarks = sm;
  }
  Void display
  {
```

```
system.out.print("the student name is" + SN);
system.out.print("the student roll no is:" + SR);
system.out.print("the student marks is" + Sm);
}
public static void main(string args[])
{
    student = new student();
    S.set data = ("Bhuvana", 232U1A3700, 95);
    S.display()
    }
}
```

② write a java program to implements red, blue, green colours using this keyword.

```
import java.lang.*;
class class
{
    String red;
    String blue;
    String green;
    void set data(string R, string B, string G);
```

```
{
    this.Red = R;
    this.Blue = B;
    this.green = G;
}
system void display()
    {
    system.out.println("The red colour is" + R);
    system.out.print(" The blue colour is" + B);
    system.out.print(" The green colour is" + G);
    }
Public. static void main (string args[])
{
    colour c= new colour(Red, green, blue);
    s.set data();
    s.display();
}
}
```

* Constructor overloading :-

* Constructor are should maintain same name.

* Constructor overloading depends of

1. number of parameters
2. number of types of parameters
3. order of parameters

* § constructor more than should maintain const signature, any java classes supported more than

syntax :- Constructor

C.S = Modifier + constructor name + parameters

EX = public sai (int a, int b)

Ex :-

Write a java program to method overloading

```
import java.lang.*;
class class student
{
  String sname;
  String s RLNo;
  String s marks;

  student (string sn, int sRL, int sm)
```

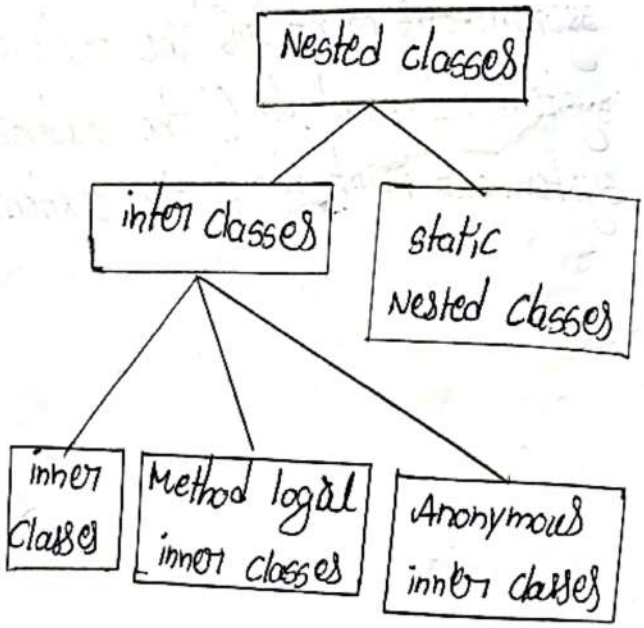
```
{
    This.s name = sn;
    This.s Roll no = SRL;
    This.s marks = sm;
}
student (string sn, int SRL)
{
    This.s name = sn;
    This.s Roll no = SRL;
}
system.out.println("The student name is"+SN);
system.out.println("The student roll no is"+SRL);
system.out.println("The student marks is"+sm);
}
public static void main(string args[])
{
    ____
    ____
    ____
}
}
```

Nested classes :-

The java programming language allows you to a class with another class such a class is calle a nested class

syntax :-

```
class outerclass{
...
class Nested class{
...
}
}
```



1 inner classes :-

A class declared with in the class is called inn class. Inner classes can be form in two ways.

1. Regular inner classes
2. static inner classes

```
class outer class {
....
}
static class static Nested class {
... }
class inner class {
....

}
}
```

* If a class declared directly with in another class such as type of class is called normal or regular inner class.

* In this inner classes we are not allowed to declare any static declaration. Hence normal inner classes are not allowed main() method.

Object creation for inner class :-

1. instance area

```
inner class obj = new inner class();
```

2. static area of outer (or) from out side of outer class:

```
outer o = new outer();
outer.inner obj = o.new inner();
                  (or)
outer.inner obj = new outer().new inner();
```

Static inner class :

```
class outer class {
    .....
static class static Nested Class {
 }
}
```

Example :-

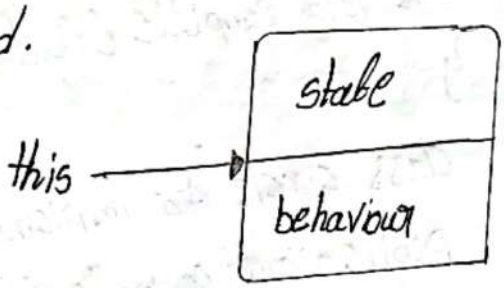
```
class outer
{
 class inner
 {
  public void m1()
  {
   system.out.println("inner class method");
  }
 }
public static void main (string args())
{
 outer o = new outer();
 outer.inner i = o.new inner();
  i.m1();
 }
}
```

This Keyword :-

The this keyword refers to the current object in a method or constructor. The most common use for this keyword is to eliminate the confusion between class attributes and parameters with the same name.

uses of this keyword :-

1. This keyword can be used to refer current class instance variable.
2. " " " " used to invoke current class method (implicity).
3. " " " " used to invoke the current constructor
4. " " " " passed as an argument in the method call.
5. " " " " passed as an argument in the constructor call.
6. " " " " used to return the current class instance from the method.



Final keyword.

- Final is a cross access specifier.
- Access specifier - given additional meaning to any class, variable and method.
- Final can be associated with class and variables.
- Final can't be associated with method.
- Sometimes final can be associated with method.

Final variable ⟹ To create constant variable

Final Method ⟹ Prevent method overriding

Final classes ⟹ prevent inheritance.

Write a java program to find area of the circle and semicircle using the multiple inheritance.

```
import java.lang.*;
Public interface Area {
final float pi = 3.14;
public void compute (float x);
}

class circle Ar implements Area{
Public void compute (float x){
System.out.println (" The area of circle is:"+
```

```
(pi*r*r));
}
}
class semicircle Ar implements Area
{
public void compute(float r)
{
System.out.println("the Area of semicircles is":+(1/2
*pi*r*r));
}// end of the semicircle class
class Multinheritancetest
{
circle Ar cA = new circle Ar();
semicircle Ar sA = new semicircle Ar();
Area A;
cA = A;
cA.computer(5.23f);
sA = A;
sA.computer(5.23f);
}
}
```

Expected output :~

The area of circle is : 85.88

The area of semicircle : 42.94.

pass by Value :~

- when you pass a primitive data type (eg, int, float, to a method, it is passed by value.
- This means a copy of the actual value is passed to the method parameter. Any changes made to the parameter inside the method do not affect the original value outside the method.

Program :~

```
public class pass by value Exomple {
public static void main (string args) {
int num = 10;
system.out.println ("Before call methode :" +num);
modify primitive (num);
system.out.println ("After calling method:" +num);
}
public static void modify primitive (int number) {
number = 20;
```

```
        System.out.println("Inside method:" + number);
    }
}
```

Output :~

Before calling method : 10
inside method : 20
After calling method : 10

Pass by Reference :~

- when you pass an object (which includes arrays and instances of classes) to a method, it is passed by reference.
- This means the method recieves a reference to the original object. changes made to the object's state inside the method will affect the original object.

program :~

```
Public class pass by reference example {
Public static void main (string[] args) {
string Builder sb = new string Builder ("Hello");
```

```
system.out.println("Before calling method:"+sb);
modifyObject(sb);
system.out.println("After calling method:"+sb);
}
public static void modifyObject(StringBuilder str
str.append("(world");// changes are made to the ori
object 'sb'
system.out.println("Inside method:"+str);
}
}
```

output :-

Before calling method. Hello
inside method: Hello world
After calling method : Hello world.

Imp

* Access specifier :-
* Access specifier is given additional meaning to the class method & variables.
* Access specifier are classified in two types.
  1. abstract 3. static
  2. final

1. **Abstract** :- It is a specifier means it gives additional meaning to class, method
* Abstract is a key word
* Abstract can be associative with class
* Abstract can be associative with method.
* Abstract can't be associative with variable.
* Abstract classes are having.
  1. Abstract method.
  2. Non-Abstract variables
* Abstract classes provided the syntax

syntax :-

Abstract (access specifier) class x → class name. (x → keyword.)

```
{
Abstract.methods;
Non-Abstract variables;
Non-Abstract methods;
```

// some times using non abstract methods.

}

1. Abstract Method :-

1. Abstract method are not having any instruction

2. every java abstract method end with a semicolon

3. Abstract method syntax.

syntax :-

Abstract + no return type (void) + Method name

Keyword + parameter list

Ex: Abstract void sai(int a)

① Write a java program to implement abstract keyword. ~~final is a access~~

Final :-

* Final is a Access specifier

*Access specifier is given conditional meaning to class, method variable.

* ~~static~~ can be

Write a java program to implement Abstract keywood.

```
import.java.lang.*;
Public interface Area
{
 final float pi = 3.14f
 Public void compute (float r);
 }
 class circle Ar implements Area.
 {
 public void compute (float r)
 {
  system.out.println (" The area of the
                circle is:" + (pi*r*r));
 }
}
class semi circle Ar implements Area
{
 public void compute (float r)
 {
 system.out.println (" The Area of semi circle
                is:" + (1/2 * pi * r * r));
 }
 class multiinheritance test
 {
```

```
public static void main (String [] args)
{
 Circle Ar [A = new Circle Ar [];
 semi Circle Ar SA = new semiCircle[];
 Area A;
 CA =A;
 CA.compute (5.23f);
 SA = A;
 SA.compute (5.23f);
 }
}
```

not accessible

3 static :- F A S
M V C

*It is a access specifier.
* Access specifier means given additional mean
class method variable.
* It is a key word.
* It can be Associated with method, variable
* It can not be Associated with class
* static methods is always required static variables.
* some times static methods are refering non static variables.

Example:

1. Java main method

```
public static void main (string [] args)
```

2. Import java.lang.*;

```
Class Anshu
{
 static void x()
 {
 static int count=0;
 system.out.println("incremented count value:
                                    + count);
 }
public static void main (string[] args)
{
   Anshu A = new Anshu ();
       A.x ();
  }
}
```

① Write a java program to implement Multiple inheritance / interface / Access specifiers.

(or)

Write a java program to find the area of circle and semi circle

Interface: Interface is a collection of abstract methods and final variables.

It acts as both keyword and technique interface technique provides implements key words multiple inheritance is implemented using interface technique.

Interface constructed same as the class construction.

```
interface structure
interface interface name
{
 Abstract method ;
 final variables ;
}
```

Example :

```
interface Area
{
 public float compute (float x);
 final pi = 3.14 f;
}
class circle Area implements Area
{
 public float complete (float x)
 { return (pi * x * x);
```

```
    }
  }
class semicircle Area implements Area
{
  public float compute (float r)
  {
    return (0.5 * pi * r * r);
  }
}
 class main
{
  public static void main (string [] args)
  {
    Area A
Circle Area CA = new circle Area ();
semi circle Area SA = new semicircle area ();
  A = CA ;
 system.out.println ("Area of the circle is :" +
                          CA.compute (5.34f));
  A = SA ;
 system.out.println (" the Area of semi circle is :" +
                          SA.compute (5.34f);
  }
 }
```

⇒ Save it using file name.java i.e file name should be same as Main Class name.

⇒ Javac Main.java

⇒ Java Main

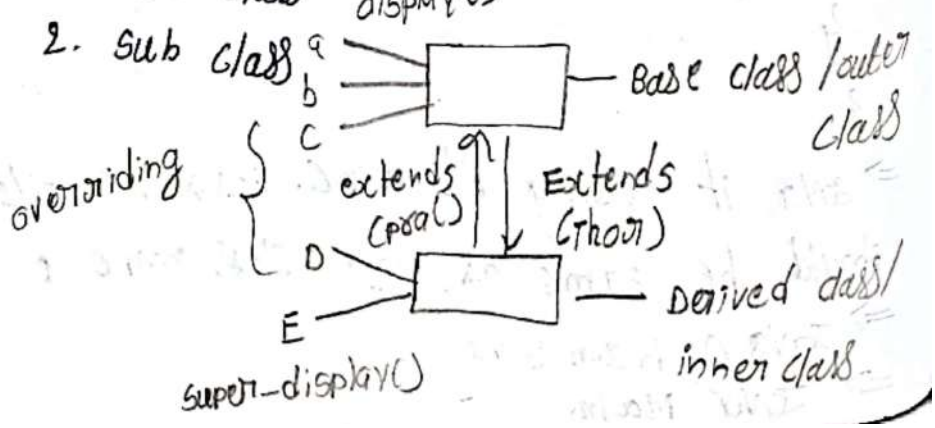
Method overriding:

* Method overiding changing the functionality one class method to another class method.
* Two methods are having same method signature.
* Method signature means collection of Met return type + Method name + parameter list
* Method overriding is implemented with the help of using super keyword.
* super keyword works at only one level
* Exchanging any method information from one class to another class.
* super key is also supported single level inheritance (acquiring properties from one class to another class)

Example:

intially consider two classes

1. Super class
2. Sub class



unit-III

Introduction of Inheritance

Inheritance means accuring the property one class to another class.

* Inheritance accuring the super class
* super class is base technically called base class (or) parent class.
* sub class derivated class (or) chaild class
* Inheritance property supported actence keyword.
* actence key word is used to exchanging information from one class to anther class.
* Inheritance are again devided into that are.
    1. single level inheritance (or) single
    2. Multilevel inheritance
    3. Multiple inheritance.
    4. Hybered inheritance.

1. single (or) single level inheritance:

* Inheritance means accuring the property from super class to sub class
* single level inheritance required minimum two class.

1. super class (or) base class

2. sub class (or) derivated class

* single level inhexitance is implemented using extends key word.

Example : write a java program to implement single level inhexitance.

```
import java.lang.*;
class parent
{
  string p name;
  int p age;
  string p address;
  parent
  {
    p name = "Rupa";
    p age = 18;
    p address = "pullathigalapadu";
  }
  void display()
  {
```

```
system.out.println("parent name":+ P name);
"    "    "    "("parent age":+ P age);
"    "    "    "("parent address":+ P add);
{
class child extends parent
{
string c name;
int c age;
child()
{
c name = "Pavani";
c age = 16
}
void display2()
{
system.out.println("The child name is"+
                    c name);
"    "    "    "("The child age is + c age);
}
}
class single inheritance test.
```

```
{
public static void main (String args[])
{
    parent p = new parent();
    Child c = new child();
    p. Display();
    C. Display 2();
}
}
```

2. Multi level inheritance :-

* inheritance means accuring the property from the super class to sub class.

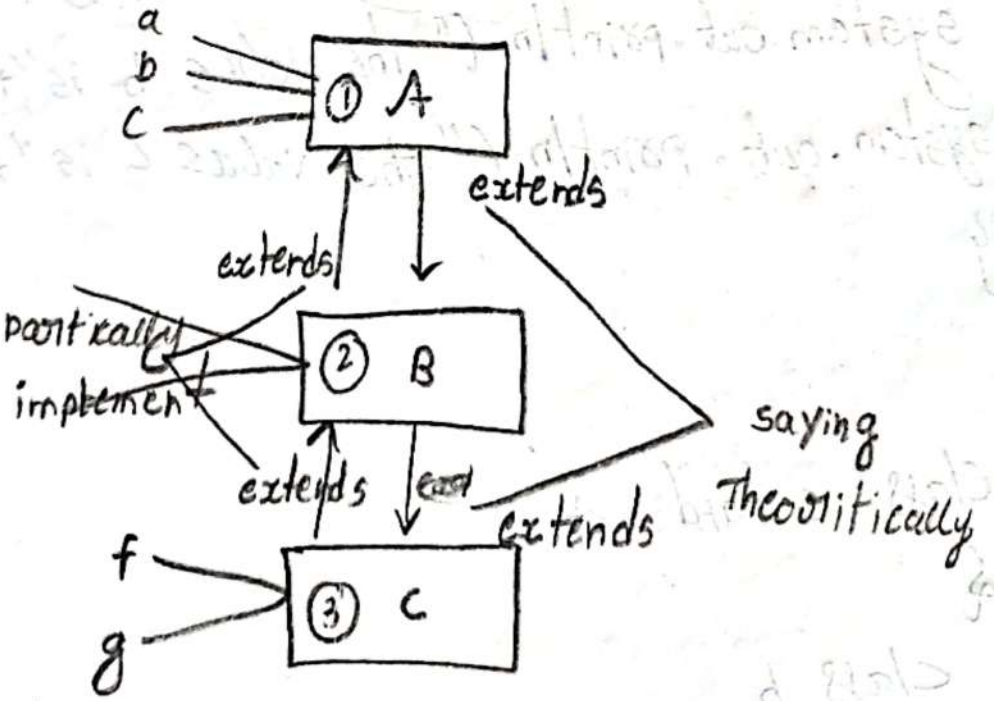
* Multilevel inheritance is required more than two class classes.

* Multi level inheritance is used different key word.

1. Extends keyword.
2. super keyword

* Extend keyword is used exchanging any information from one class to another class.
* super keyword is applied on methods.
* super keyward ~~is~~ applied to method overloading supported
* method overloading is always maintains. methods having same name.
* super key work at only one level inheritance.
* super key associated with method using dot operator.

Example program for M.L.I



```
import.java.lang.*;
class a
{
string a;
int b;
float c;
a()
{
a="x";
b = 45;
c = 15.65f;
}
void display()
{
system.oub..println("The value a is"+a);
system.out.println("The value b is"+b);
system.out.println("the value c is"+c);
}
}
class extends A
{
class b
```

```
{
  string d;
  int e;
  b()
{
  d = "y";
  e = 54;
}
void show()
{
  system.out.println("the value a is"+a);
  system.out.println("the value b is"+b);
  system.out.println("the value c is"+c);
system.out.println("the value d is"+d);
system.out.println("the value e is"+e);
}
}
class c extends B
{
  class C
  {
    string f;
    int e;
```

```
    C()
    {
    f = "z";
    c = 15;
    }
void show()
 {
  super.show();
  system.out.println("the value f is"+f);
  system.out.println("the value is"+c);
 }
}
class multilevel inhesitance.
{
  public static void main(String args[])
  {
      c c1 = new();
      c1.show();
  }
}
```

Multilevel

3. Multiple level inheritance :-

inheritance means accuring property from one class to another class.

```
import java.lang.*;
Public interface Area {
final float pi = 3.14;
Public void compute (float x);
}
class sphere Ar implement area{
Public void compute (float r){
System.out.print (" The are of circle is sphere
                    is:" + 4 * pi * r * r));
}
}
class Hemisphere Ar implement Area{
Public void compute (float r)
{
  System.out.println(" the area of Hemisphere
                    is + (3 * pi * r * r));
}
}
class Multiinheritance {
Public static void main (string[] args)
{
  sphere Ar sA = new sphere Ar ();
```

```
Hemisphere Ar HA = new Hemisphere Ar();
Area A;
SA = A;
SA.compute(5.23F);
HA = A;
HA.compute(5.23f);
}
}
```

output :-

The area of sphere is : 343.36.
The area of Hemisphere is : 257.52.

Declaration And initialization of a

* It is used to store data objects having the same data type, each and every element in an array has a unique index value. In an array, we can store elements of different data types like integer, string, data and etc.

Declaration of array:

- int arr[];

initiation initialization of array:

int [] arr = new int [20];

↳ size at array.

* Array is a group of elements of the same data type.

* Array specifies a common name.

* Array declaration has maintain 2 types in int

1. Data type arrayname []; EX: int Kumar[];
2. array name Data type []; EX: Kumar in[];

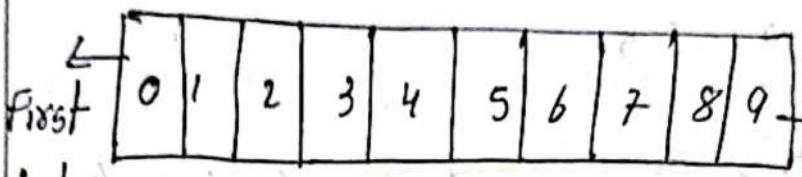
* create the memory for the array using the the object's (new operator).

* Declaration of array and creating memory location is furthorly written as

Data type array name [] = new Data type [size];

array name Data type [] = new data type [size];

put memory values into array



Array length (a).

Simple example program for array :-

```
import. java.lang.*;
class Maintest{
Public static void main (string [] args){
int age = {12, 4, 5, 2, 5};
system.out.println ("Accessing elements of array");
system.out.println ("First element:" + age[0]);
"  "  " ("second element:" + age[1]);
"  "  " ("third element:" + age[2]);
"  "  " ("fourth element:" + age[3]);
"  "  " ("fifth element:" + age[4]);
}
}
```

output :

Accessing elements of array:

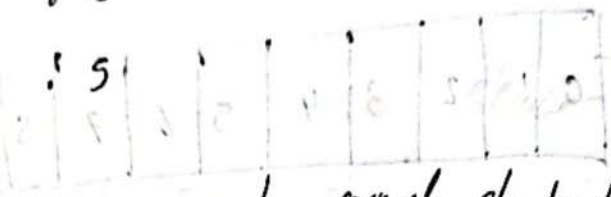
First element : 12

second " : 4

Third " : 5

fourth " : 2

fifth " : 5



* Write a java program to creat student Roll using array

* Write a java program to creat student Name using array

* Write a java program to create student Roll, Name using array.

Sol

```
import java.lang.*;
class student.
```

operations on array elem

* operation on array elements :-

operations on array elements involves some of the methods searching. sorting. insertion deletion traversing.

* searching :-

This involves finding the index of a specific element in the array. This can be done using the linear search of binary search algorithms.

* sorting :-

This involves arranging the elements of an array in a specific order, such as ascending or descending order. This can be done using sort, or quicksort algorithm.

* Insertion :-

This involves adding a new element to an array at a specific index. This can be done by shifting the existing elements of the array to make room for the new element.

Deletion :-

This involves removing an element from an array at a specific index. This can be done by shifting the remaining elements of array to fill the gap left by the removed element.

Traversing :-

This involves visiting each element of array in a specific order. This :

2. Two Dimensional array syntax :-

Data type arrayname [][] = new Data type [][]

EX : int a[][] = new int [3][3];

Example :- W. J. P. implement matrix multiplication using two dimensional array.

```
import java.lang.*;
class class name Matrix Multiplication test
~~matrix mul~~
{
 Public static Void main (string args)
 {
  int a[][] = new {{10,20,30},{40,50,60},{70,80
  90}};
  int b[][] = {{90,100,110},{10,20,30}}
 {
  int c [][] = new int [3][3];
 for (int i=0; i<3; i++)
 {
  for (int j=0 ; j<3; j++)
 {
  c[i][j]=0;
  for (int k=0; k<3 ; k++)
```

```
{
 c[i][j]=c[i][j]+a[i][k]*b[k][j];
 }
 }
}
// print a matrix
for(int i=0; i<3; i++)
{
for(int j=0; j<3; j++)
{
 system.out.print(" the matrix of a is "+a[i][j]
 }
 system.out.print("
 }
// print b matrix.
```

① Write a java program to implement addition, subtraction multiplication using switch case.

```
import.java.util.scanner;
class main
{
 public static void main (string args[]){
 char operator;
 Double number1, number2, result;
 scanner input = new scanner (system. in);
 system.out.println("I chose an operator: +, -, *);
 operator = input.next().charAt(0);
 system.out.println("Enter first number");
 number1 = input.next Double();
 system out.println("Enter second number");
 number 2 = input.next double();
  switch (op)
  {
   case "+":
    result = number1+number2;
   S.O.P(number1+" + number2+" = "+result);
   break;
case -:
    result = number1 - number 2;
    S.O.P (number +" - "+ number 2+" = "+ result);
```

```
        break;
    case '*':
        result = number1 * number2;
        S.O.P(number1 + "*" + number2 + "=" + result);
        break default.
    S.O.P("Invalid operator!");
        break;
    }
  }
}
```

unit-4

Package

* **Package :-**

Package is a collection of classes, methods and interfaces.

or

Package is a same group of classes and subclasses.

* **interface :-**

It is a collection of final variables and abstact.

* **class :-**

It is a collection of variables and method

* **method :-**

Method is a collection of variables and instructions

* Package are again divided into 2 types.

1. system defined packages or ~~Built in packages~~ of ~~predefined~~

2. user defined packages

system defined / Built in packages : It is provide by developes. system defined packages are divided into different types.

1. language package
2. AWT
3. NET
4. SQL
5. IO
6. APPLET
7. Swing / Java X Swing
8. util
9. AWT Nent
10. Math.

1. lang / language package :-

* language package is a default package.
* language package contains different informations.
  1. string 2. class template information
  3. string buffer information.
* language package information retriving from memory to user application using import statement.

import syntax:

import / Java sw name.package name.*;

(or)

import Java s/w name package name.Class name.

Ex:

1. Import Java.long.*;
2. Import Java long.Class A;

2. AWT package :-

AWT stands for abstract window toolgate. AWT meas hiding of graphical components to the user. AWT acts as a (graphical user interface). GUI.

-> It is also act as a super window or superframe.

-> Any user wants to creating of a own window or own frame. using super frame or super

3. grid layout window. or sub window using extends keyword.

-> Abstract window toolgate super different to graphical components to the user

1. Button
2. label
3. Text field
4. Text Area
5. Choice box
6. check box
7. scroll bar
8. list
9. menu bar
10. Radio Button.

-> In Additionally AWT supported layouts & listerns

-> layout means assanging any graphical compona into some order

-> Different byouts are

1. border
2. Flow layout
3. orid back
4. orid layout
5. crd layout

-> listner means any components iteracting with graphical components to another gc

-> AWT supported different.

1. Action listner
2. item leasner
3. focus listner
4. key listner
5. mouse listner
6. mouse motion leosner
7. Adiest ment listner

3 Net package :-

net package provided different internet protocal in leasorner. Different internet protocals are. It is a connection less protocal

connecting means any data information.

1. UDP (User Data protocol):

It is a connection less protocol

2. TCP (Transmission control protocal):

→ TCP stands for tonsmission control protocal.

→ It is a conection oriented protocal conection

→ TCP protocal is also established comunication between clien & server.

3. IP (Internet protocal):

→ It is a connection oriented protocal

→ Connection oriented protocal means sending any data information from client to server.

4. HTTP (Hyper text Transfer protocol):

→ HTTP is usually estaliblished

→

→ Net package is supported different internet technologies are

1. LAN (Local Area Network)
2. MAN (Metropolitan Area Network)
3. WAN (Wide Area Network)

→ Network package is also supported prime socket and sever side sockets.

5. SQL (structure query language):-

→ SQL stands for structure query language.

15/01/24

Exception Handling

Introduction :-

An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions

Program :-

```
import java.lang.*;
class Importance test
{
Public static void main (string args[])
{
   int a;
   int b;
   int c;
   a=5;
   b=0;
   c= a/b;
}
}
```

output :-

Exception in thread "main" java.lang.
Arithmetic exception : /by zero of importance
Test.main (Importance Test.java : 23)

Exception Hierarchy :

* The exception classes has two direct subclass: error and exception.
* The exception class is used to indicate errors that can be handled by the program. such as invalid input or file not found errors.
* The exception class has two further subclasses. checked exceptions and unchecked exceptions.

Program :-

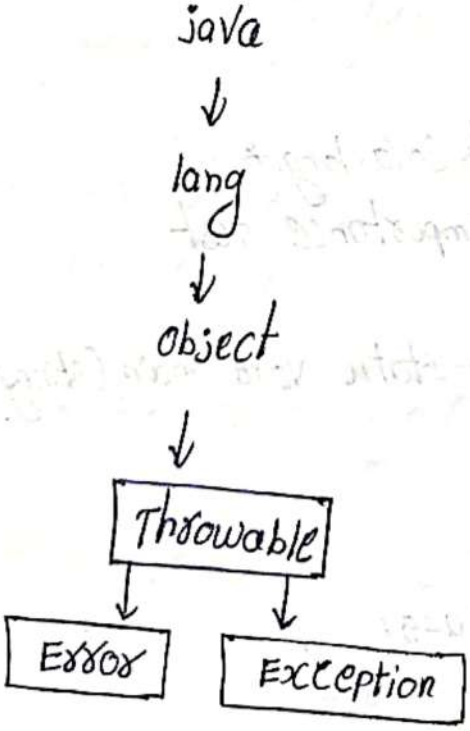
```
import java.lang.*;
Class Importance Test
{
 public static void main (string args[])
 {
  try
  {
   int a=5;
```

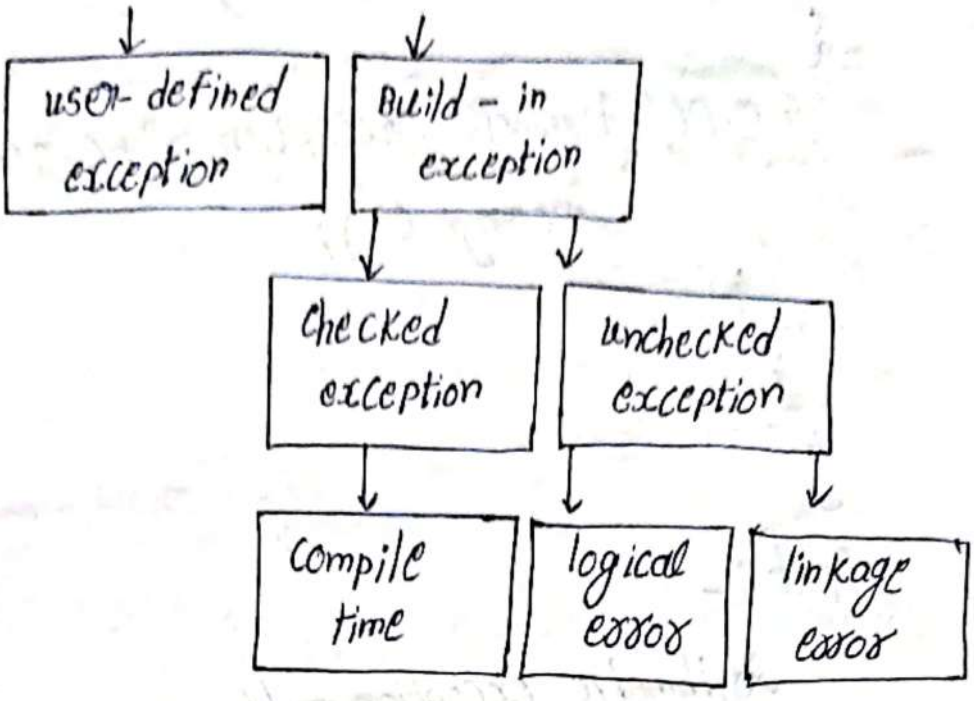
```
        int b=0;
        int z;
        z = a/b;
    }
    catch (Arithmetic Exception Ae)
    {
     s.o.p("Runtime divide by zero error");
    }
  }
}
```

o/p:

Runtime divide by zero error.

Exception Hierarchy flow chart :-





try :

scope of any exception associated with it.
It detects the exception.

Example - 2 :

Program :

```
import java.lang.*;
class main
{
public static void main(String args[])
{
 try
 {
  int divide by zero = 5/0;
  S.O.P(" Rest of code in try block:");
}
```

```
catch (Arithmetic exception e)
{
 s.o.p(" Arithmetic exception => "+e.get
        message());
  }
 }
}
```

output :-

Arithmetic exception => / by zero.

Throw :

The throw keyword is used in java to explicity throw an exception from a method or any block of code. It can be used to throw either checked or unchecked exceptions.

Program :-

```
import java.lang.*;
class throw example
{
  static void validate Age (int age)
  {
    if (age < 18)
    {
      throw new Arithmetic exception("age is
not valid, must be in order");
```

```
    }
    else
    {
      s.o.p("Welcome to the club!");
    }
  }
  public static void main(String args[])
  {
    try
    {
      validate age(15);
    }
    catch (Arithmetic exception e)
    {
      s.o.p ("Exception caught:" + e.get message());
    }
  }
  validate Age(20);
 }
}
```

output :

Exception caught Age is not valid must be 18 or older.

Welcome to the club!

Throws keyword:

The throws keyword in Java is method signatu
to indicate that a method might throw one or mor
exception.

syntax:-

throw new exception type("Error message");

program:-

```
class Throw Key1
{
public static void main(string args[])
{
try
{
check age(15);
}
catch(exception e)
{
S.O.P("caught an exception:"+e.get
                    message());
}
}
static void check age(int age)
{
if(age<10) throw new illegal argu
ment exception("Age must be is or older");
```

```
    }
}
```

output :-

caught on exception : Age must be 18 or older.

Finally :

The finally block is used to put important codes such as clean up code eg closing the file or closing the connection.

A finally contains all the crucial statements regardless of the exception occurs or not.

Program :-

```
import java.lang.*;
class OTF or
{
public static void main (string args [])
try
{
S.O.P ("Inside try block");
S.O.P (34/0);
}
catch ( Arithmetic exception e )
{
S.O.P (" catch : exception handled.");
}
```

```
        finally
        {
         s.o.p("finally: I excute always.");
        }
    }
}
```

output :-

inside try block
catch: exception handled
finally: I excute always.

14/10/24

5. SQL (structure query language):

-> SQL stands for structure query language
-> SQL is backend language
-> SQL support different commands versions are:
-> create // create table table name ().
-> insert insert into table name values (attribute, value attribute 2.000)
-> delete
-> update
-> select // stat from table name

eg: select star from // - cs & D.S

Here retriving all attribute values from all the cs & ds values.

eg:

S.roll	Sname	S marks
1901	X	
1902	Y	
1903	Z	
:		
:		
:		

Swing X package :-

* It is extended package of swings
* It is supported light weight components
* swing components are identity. 'j'

Math package :

Math package supported different mathematical function are square roots, sine, tan, cosine.

IO package :

Io stands for Input output package

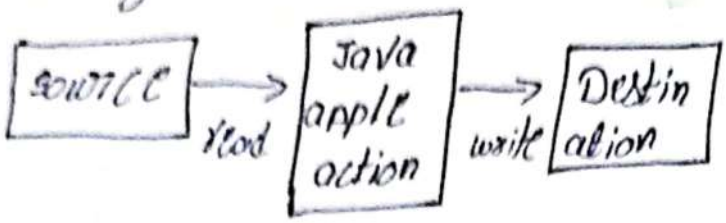
Io supported streams

Io packages is divided into 2 types.

* Input stream
* output stream

stream:

Stream is a linear flow of data that data travels from stately source to output stream definations.



util package:

util package supported scanner object and using list util package is also supported.

1. Vectors
2. Random number
3. linked list
4. Date
5. Time.

APPLET :-

→ Applet is a small application program

→ Applets are supported web applications

→ Applets are supported different types of HTML Tags and creating of a graphics.

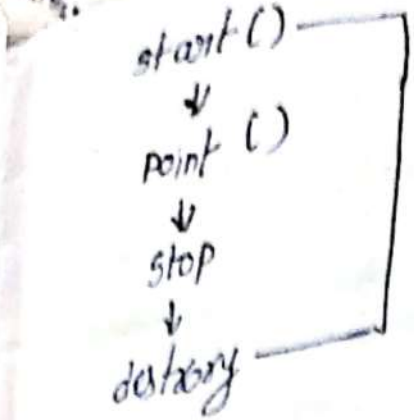
Ex: < applet > < /applet >

→ Applet tag supports 3 Attributes.

1. code - java filename.class
2. Height
3. installation

↓

start ( )



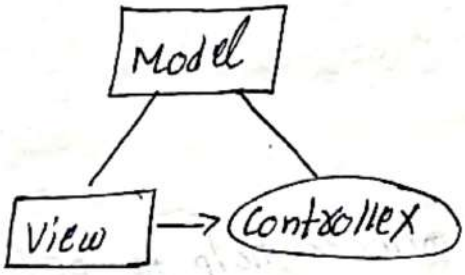
swings:

-> It is the extension of AWT

-> swing package supported light weight components

-> swing supported J components

ex: 1. J button
2. J label
3. J Textfield
4. J Text Area
5. J choice Box
6. J check Box
7. J scroll Box
8. J list
9. J Radio Button
10. J combo box



Enumaration:

Enumalation are special data type in java that enable for a variable to be a set of the pre-defined constant. It includes compass ~~diff~~ directions (values of North, south, EAST, west).

Program :-

```
import java.lang.*;
Class Enum Example
{
 // Declare and enum type
 Enum Day
 {
 Monday, Tuesday wednesday Thusday friday
saturday Sunday.
 }
 // Create a day object
 Day day. day. Monday;
 // print the day
 S.O.P (day);
}
```

output :

Monday

Math class :

Java.lang.math class methods help to perform numeric operations like square, square root, cube, cube root, exponential and trignometric operations

Declaration :

final class math
extends object

program :

```
class Java math example
{
```

```
public static void main (string args [])
{
  double X = 23;
  double Y = 4;
S.O.P(" maximum number of x and y is:"+ math.max(x,y));
S.O.P(" square root of y is :"+ math.sqrt(y));
S.O.P(" power of x and y is:"+ math.pow(x,y));
}
}
```

output :-

```
maximum number of x and y is: 28.0
square root of y is: 2.0
power of x and y is: 614
```

Wrap per class:

Wrapper classes are classes that encaps wate primitive data types.

program:

```
public class main
{
public static Void main (string args[])
{
  integer my int = 5;
  Double my Double = 5.99;
  Character my char = 'A';
  S.O.P (my int);
  S.O.P (my Double);
  S.O.P (my char);
```

```
    }
}
```

O/P :-

5 5.99 A

Auto Boxing :-

In Auto Boxing the java compiler automatically conssents primitive types into their corresponding wrapper classes objects.

Program :-

```
import java.util.Array list;
class maintest
{
public static void main (string args[])
{
integer my int = 5;
Double my Double = 5.99;
character my char = 'A';
S.O.P (my int);
S.O.P (my Double);
S.O.P (my char);
}
}
```

output :-

5 5.99 A

Auto Boxing:-

In Auto Boxing the Java compiler automatically converts primitive types into their corresponding wrapper classes.

Program:-

```
import java util.Array list;
Class main test
{
Public static void main (string arg
{
Array list< integer> list = new Array list
                              <>();
// autoboxing    list.add(6);
list add(5);  s.o.p("Array list :"+list);
}
}
```

17/10/24

own package creation:

step 1: Declare the package

syntax: package packagename;
(package → Keyword; packagename → depends on you)

Eg: Package sai;

step 2: Declare the class

syntax: class classname;
(class → Keyword; classname → depends on you.)

eg : class sai

Step 3: class Declared as public

Modifier ←

```
Public class sai
{
}
```

step 4 : create sub directory under the main directory

Eg: C:> my plict > cc5/cc1/cc2 > MD. sub directory (main directory)

Note: Package name same as the subdirectory name

(or)

(Packagename and subdirectory names are same)

C:> mypic > cc1 > MO. sai;

C:> myplic > cc1 > co. sai ;

step 5: save the package class under the subdirectory

eg: sai1. java

step 6: complie the package file of package class under the subdirectory

eg: javac sai1. java

step 7 : checks atleast once created or not created. class file under the subdirectory.

write a java program to create a own package

```
Package sai;
class sai 1
{
 Public static void main (string args[])
 {
  void display()
  {
   S.O. PL(" Hai");
   S.O.P (" Welcome");
   S.O.P (" Java Lab");
 }
}
```

How to access own package into user classes:

1. To access own package into user classes using syntax &

syntax: `import own package name.package.class name;`

Ex: import sai.sai1;

Write a java program to implement own package access into user classes.

```
import sai.sai1;
class sai2
{
Public static void main(string args[])
{
 sai1 s1 = new sai1();
 s1.display();
}
}
```

output: Hai
Welcome
Java Lab

2. Shift from subdirectory to main directory. save the sai2 application under the main directory.

sai2.java

3. Compile the sai2 Application under the main directory

javac sai2.java

4. Run the sai2 Application under the main directory

java sai2

stream :-

* stream is a linear flow of data
* linear flow of data travel from source to destination.
* streams are again divided into mainly two types.
    1. Input
    2. output
* streams are accepted into 2 ways
    1. byte streams
    2. character streams
* streams are created with help of using different
    1. create the stream (any type of stream at creating using objects
    2. call the method some methods are

Ex :- read the method → this method supports to reading at string information from key board to user application.

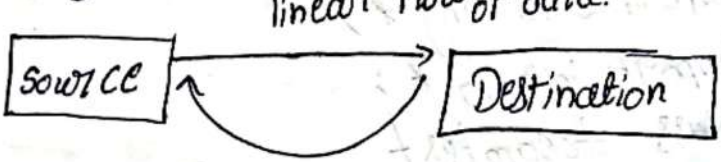
Integer.parseInt Method ;-

* this method supported to reading at integer values from keyboard to user application.

* some times read and we write methods are used in streams.

* Finally close the stream using close method.

* block diagram representation of stream.



① write a java program to create enter your name from keyboard using streams.

Sol

```
import java.lang.*;
import java.i.o.*;
class stream test
{
public static void main (string args [])
{
try {
Data Inputstream d=new Data Input Stream (system in);
string s name;
```

```
S.O.P("Enter the name:");
S name = d.readline();
S.O.P(" the Student name is:"+S name);
}
catch (exception e) {
S.O.P("Error");
}
}
}
```

Write a java program to create employe details (Ename, EId, E salary) using streams.

```
import.java.long.*;
import java.io.*;
class streamtest
{
public static void main (string args[])
{
try{
Data Input stream d=new Data Input stream (System.in);
string E name;
int E Id;
int E salary;
E name = d.readline();
S.O.P("the Employee name is:" + Ename);
E Id = Integer.parseInt(d.readline());
S.O.P(" the Employee Id is:"+ E Id);
E salary = Inter.parse Int(d.readline());
S.O.P(" the employee salary is:"+ E salary);
}
cat (exception e) {
```

s.o.p(

}

Multiple catch blocks :-

syntax :-

```
try
{
 statement / instruction 1 ⟶ loop 1
}
statement / instruction 2 ⟶ bop 2
}catch (exception e)
{
s.o.p("Enter 1").
}
```

① Write a j.p to implements multiple catch blocks using array

```
import java.lang.*;
import java.class multiple catch blocks test
{
  public static void main (string arg[])
  {
try{int a[] = new int[3];
  for(int i=1; i<3; i++)
    {
      a[i]= i/i;
    }
    for(int=0;i<3; i++
    {
```

```
        a[i]= i/i ;
    }
    }
    catch (Array out Bounds Exception e)
    {
      S.o.P("AOBE Xception");
    }
      catch (Array or Arithemetic exception)
      {
        S.o.P("AE");
      }
```

NOTE :- * one try block successfully executed ~~one~~ more than one catch blocks. Try block always requires at least one catch block.

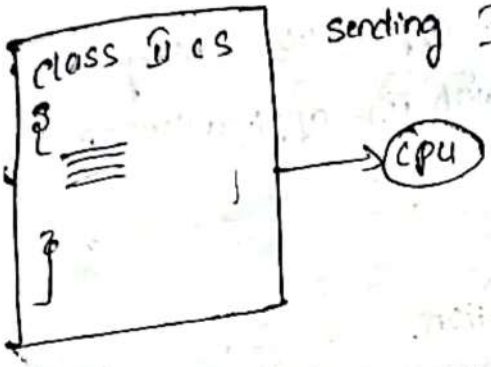
* Exception handling mechanism provides different ~~one~~ key words to the user for the purpose to find of runtime errors from any user application.

unit - IV

Introduction of Multi threading

process :- process is a execution entire application is technically under the single processer or multi processer

ex:- consider a 1 cs application

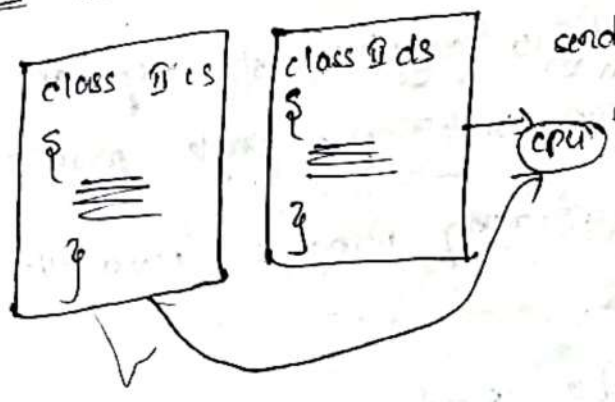


sending 1 cs application processer

for the purpose of execution single cpu or multiprocesser

multiprocess :- any user executed more than one process successfully under the single processer or multiprocesser is called multiprocess

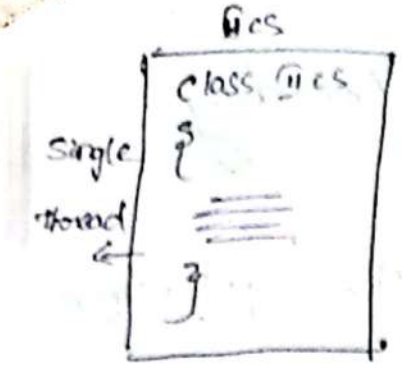
ex:- consider a 1 cs and 2 ds application



sending 1 cs and ds application into processer

Thread :- Thread is a single sequential flow of control with in application

ex:- consider a 1 cs application

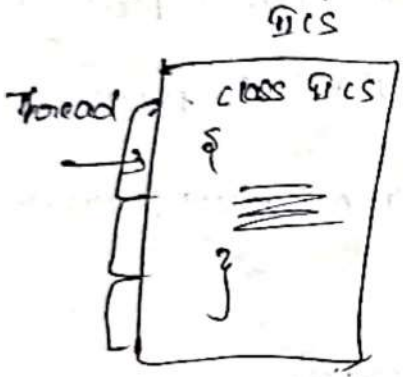


single sequential flow of control

Multithreading :-

Multithreading means any application using more than one single flow of control with in application is called Multithreading

Ex: consider a Tics application.



using more than Single one sequential flow of control

=) multithreading is used to complete of any applicati with in time duration under the single processer

=) Threads are ~~Creating~~ creating using of, two thread

1.-) Extends Thread

2.-) Implements runnable Thread.

=> thread objects are creating with the help of using new operator in java.

Ex: Thread t_1 = new Thread()

Thread t_2 = new Thread()

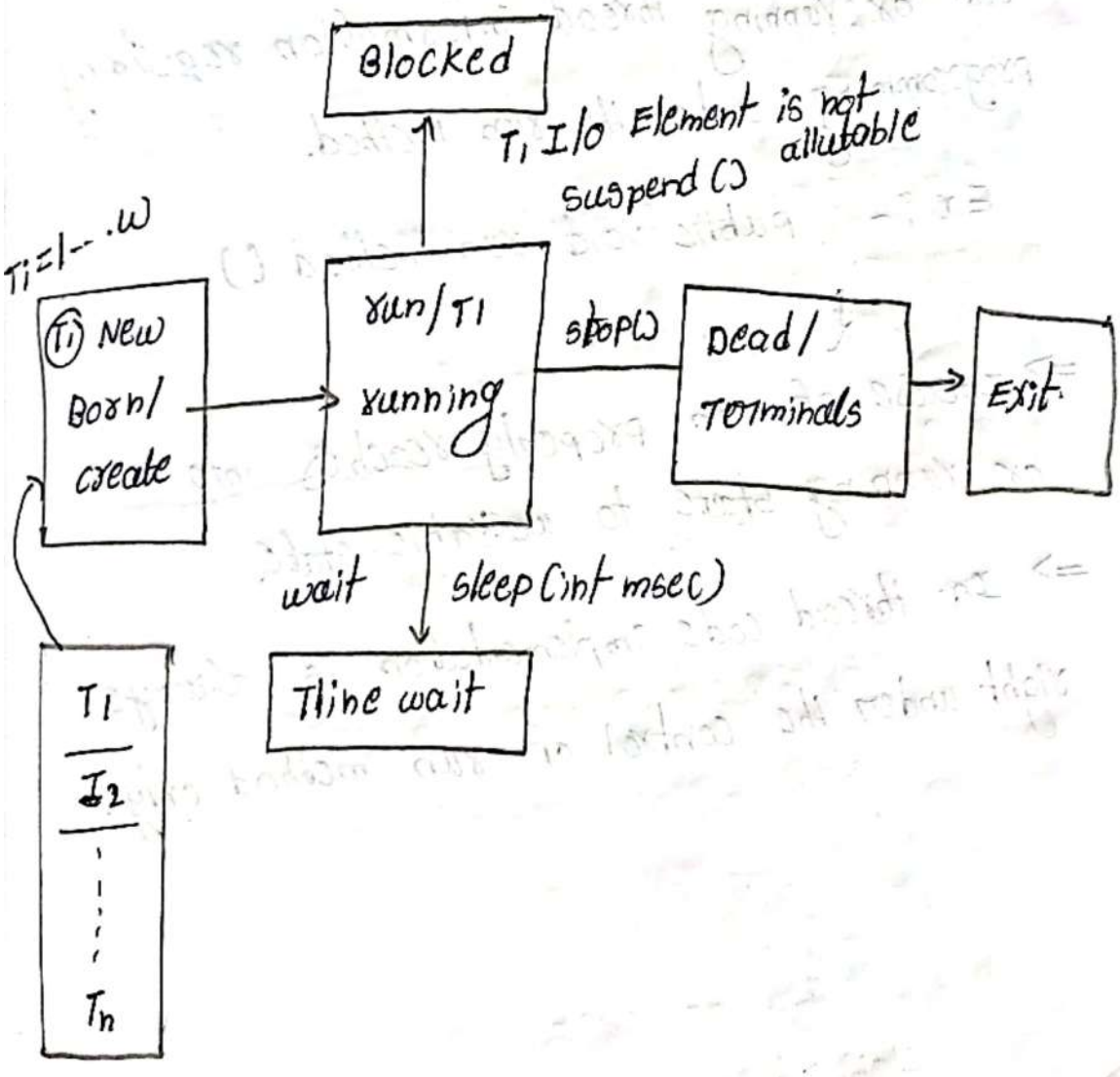
Threading concept is truly DS business

⇒ DS Threading is maintain some drawbacks.

(any thread is not properly executed under the processor at the time need to restart the entire application)

⇒ To overcome about drogback using java multi-threading concepts.

Thread life cycle or Thread state diagram:



state 1 : create

Initiating

⇒ No. of threads are accepted from private area

⇒ private area threads creating or initiated with the help of using $T_i = 1 \ldots n$ $[T_1, T_2 \ldots T_n]$

⇒ Run or Runnable state accepted different threads from create the using start method.

Run or Running acts as a processor

⇒ run or running thread information requiring programming under the run method.

Ex ;- public void run method ()

{

⇒ Incase of non properly reaches into run or running state to waikable state

⇒ ~~In~~ thread code implementation is always right under the control of run method only.

⇒ In case of any Io event are not available under from Io device to run or runner state. at time needed to send threads information temporarily send run or running state to blocked state. using suspend method.

⇒ Incase of any threads are successfully executed under the run or running state at that time needed to send threads information from run or runner state to terminate state using stop method.

⇒ Incase of any threads information needed to seeing of threads information on screen at time corresponding threads information propagate from at using that time run or running state using sleep or weight method.

Blocked state :

⇒ Blocked state accepted threads information from run or runner state.

⇒ Any threads information needed to send from blocked state to run or running state using resume method.

⇒ In case of any Io units available at that time thread information sending to run or running state using resume method.

⇒ Terminate state accepted threads information from run or running state.

⇒ successfully executed core accepted from run or running state using stop method.

* Write a java program to create details using inheritance.

```
Import java.long.*;
Class thread 1 extends thread
{
   class public void run()
   {
     for(int i=1;i<3;i++)
     {
       system.out.println("Hai");
      }
     }// end of run method
     }// end of first class
     class thread 2 extends thread
```

```
{
 Public Void run()
 {
  for(int i=1; i<6; i++)
  {
   system.out.println("welcome");
  }
 } // end of run method
} // end of second first class
class thread3 extends thread
{
  public Void run()
  {
   for(int i=1; i<6; i++)
   {
    S.O.Pln(" TO Java lab");
   }
  } // end of run method
} // end of third class
class thread3 test
{
 public static void main(string args[])
 {
```

```
        Thread t1= new Thread;
        Thread t2 = new Thread;
        Thread t3 = new Thread;
        t1.start();
        t2.start();
        t3.start();
    }
  }
}
```

* Write a java program to create 3 thread using runnable interface.

```
import java.lang.*;
class Thread tests
{
   implement Runnable
  {
    Thread t1= new Thread (this);
      "  t2 = new Thread (this);
      "  t3 = new Thread (this);
        t1.start();
        t2.start();
        t3.start();
```

```
public void run()
{
If (thread.currentthread() == t1)
{
for(int i=1; i<3; i++)
{
  s.o.pl(" Good morning");
 } // end of for loop
} // end of if statement
if (thread.currentthread() == t2)
{
 for(int i=1; i<5; i++)
 {
  s.o.pln("Good afternoon");
  }
}
public static void main(string args[])
{
 Threads test H = new Threadtest();
  }
}
```

Thread priority synchronization:

Thread priority:-

Thread priority determines the order in which threads are executed by JVM. Java threads have a priority range of 1 to 10.

Thread priority levels

1. MIN-PRIORITY (1): Lowest priority.
2. NORM-PRIORITY (5): Normal priority (default).
3. MAX-PRIORITY (10): Heighest priority.

Thread synchronization:

synchronization ensures that only one thread can execute a block of code at a time.

synchronization Methods.

1. synchronization keyword: Locks a block of code or Method.
2. Lock interface: provides more flexibility than synchronized keyword.
3. Atomic variables: update variables atomically.

Deadlock and race situation:

A deadlock occurs when two or more threads are blocked indefinitely. waiting for each other to release resources.

Deadlock conditions:

1. Mutual exclusion : Two or more threads require exclusive to a common resource.
2. Hold and wait : one thread holds a resource and waits for resource held by another thread.
3. No preemption. The operating system does not preempt one thread to give resource to another.
4. Circular wait : Threads form a circular chain. each waiting for resources held by next thread.

Example :

```
public class Deadlock {
Public static void main (string [] args)
  object lock 1 = new object ();
  object lock 2 = new object [];
```

```
object loc
Thread thread 1 = new thread [] -> {
synchronized [lock 1] [ synchronized Clock 2)[]
});
Thread thread 2 = new thread() -> {
 synchronized Clock<2 )( synchoronized Clock)[]
});
thread.start(); thread 2.start();
}
}
```

Race situation:

A race situation occurs when multiple threads access shared resources simultaneously. leading to inconsistent results.

```
Class counter {
static int count = 0;
Public static void main (string[] args) throw
interp interrupted Exception {
 thread thread 1 = new thread() -> count ++);
  "   "  2 =  "        "   "   "
```

```
thread.start().thread2.start();
thread1.join():thread2.join();
system.out.println(count);  ))
```

Inter thread communication - suspending:

Inter-thread communication in Java allows threads to coordinate and exchange information.

suspending a thread temporarily pauses its execution.

Methods for inter-thread communication

1. wait(): causes the current thread to wait until another thread calls notify() or notifyAll().
2. wait(): causes the current thread waiting on the object's monitor.
3. notifyAll(): wakes up all threads waiting on the object's monitor.
4. suspend(): Temporarily pauses a thread's execution (deprecated).
5. resume(): Resumes a suspended thread (deprecated).