Object Oriented Methodology

FOR DIPLOMA STUDENTS

Notes Prepared by
RABI KUMAR DARJI
IT Dept.

JHARSUGUDA ENGINEERING SCHOOL, JHARSUGUDA

JHARSUGUDA ENGINEERING SCHOOL
A GOVERNMENT OF ODISHA POLYTECHNIC

OBJECT ORIENTED PROGRAMMING (OOPS) CONCEPTS Dt-21/02

i) Programming Languages:-

Object-oriented Programming is a paradigm that provides many concepts, such as inheritance, data binding, polymorphism, etc.

Simula is considered the first object-oriented programming language. The programming paradigm where everything is represented as an object is known as a truly object-oriented programming language.

Smalltalk is considered the first truly object-oriented programming language

The popular object-oriented languages are Java, C, PHP, Python, C++, etc.

The main aim of object-oriented programming is to implement real-world entities, for examples object, classes, abstraction, inheritance, polymorphism, etc.

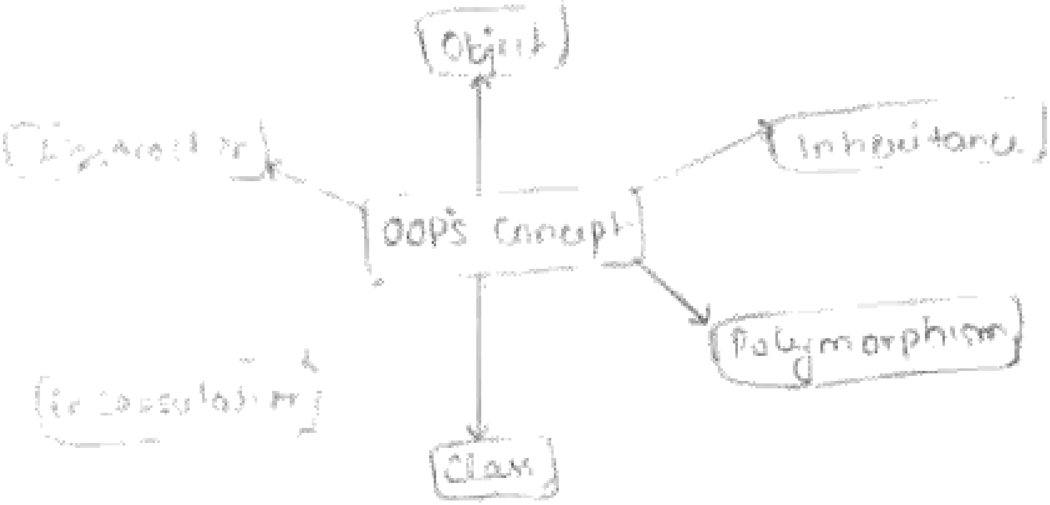
ii) OOPS (Object-Oriented Programming System):-

Object means a real-world entity such as a pen, chair, table, computer, watch, etc. Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

Apart from these concepts, there are some other terms which are used in Object-oriented design:

- Coupling
- Cohension
- Association
- Aggregation
- Composition



OBJECT

An entity that has state and behaviour is known as an object. For example, a chair, pen, table, keyboard, bike etc. It can be physical or logical.

An object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.

Example: A dog is an object because it has states like colour, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

CLASS

Collection of object is called class. It is a logical entity.

A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space

INHERITENCE

When one object acquires all the properties and behaviors of a parent object, it is known as inheritence. It provides code reusability. It is used to achieve runtime polymorphism.

POLYMORPHISM

If one task is performed in different ways, it is known as polymorphism. For ex. to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Java, we use method overloading and method overriding to achieve polymorphism.

Another ~~exam~~ can be to speak something; for ex. a cat speaks, dog barks woof, etc.

ABSTRACTION

Hiding internal details and showing functionality is known as abstraction. For example phone call. we don't know the internal processing.

In Java, we use abstract class and interface to achieve abstraction.

OR

Keeping necessary data important & discarding unnecessary data unimportant known as Abstraction

CH-2 INTRODUCTION TO JAVA :

What is Java ?

Java is a high level programing language.

It is also complied or interpieten programing languag

Java devloped by " James Gosling " in the year 1991

Java is a case sensitive [ex → $x = 5$ / $z = x + y$ Error]

The first version of Java is (JDK 1.0) was released o

23rd Jun, 1996 by Sun missolystem.

Syntex :
```
Class Class name
{
 public static vaid main(String []args)
 {
  System.out.print("     ");
 }
}
```

Public → Access specifier.

Class → object

Helloworld → Class name

static → object not required then use static

vaid → data type / return type

main() → function

String → Class

args → array name.

System → class predefined.

Out → object / referance variable.

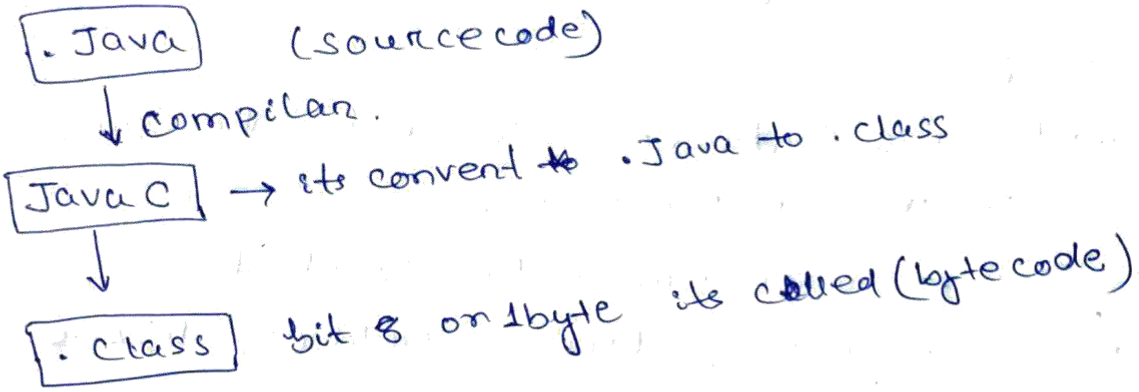
Printf → Function Predefined.

JDK : Java devlopment kit it contains tools needed to developed to pragram.
This tool could be compilar or [Java.exe), application louncher that is (Java.exe)

JRE : Java run time environment contain JVM (Java vertual machine) and Java pakage class (Java library.

Execution Model of JAVA :

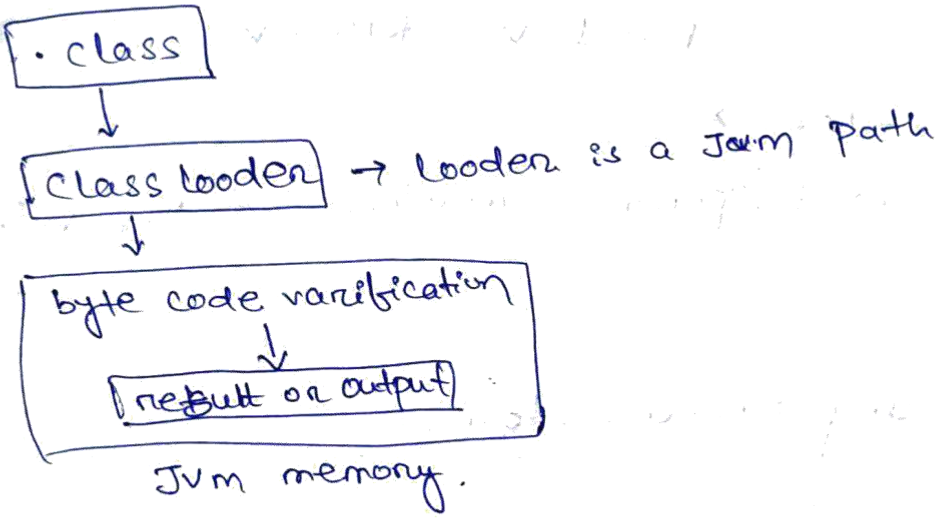
* compilation process



→ its convent ~~to~~ .Java to .class

bit 8 or 1byte its called (byte code)

* Interpreters (JVM)



→ looder is a Jvm path

What is JVM ?

JVM stand for Java virtuall machine, it is the software ~~in the language through~~ form of interpreter written in 'c' language through which can execute our Java Program.

Java pragram :-

```
Public class Helloworld.
{
   Public static vaid main (String [ ]args ]
    {
    System. out . vaid main ("Hello world");
     }
   }
```

Output → Hello world.

Identifiers :- Identifier is a smallest unit of program.

Variable :-

- A variable is a Container which holds the value while the Java program is executed.

Variable is a container that contain constant.

- It is a name of memory Location / address where data or constant get store.

Rules :-

Ram , - Ram ✓ Ram1 ✓ , R1am ✓

@Ram , 1Ram , #Ram ×

The First latter con't special character, Constant value.

Constant :-

Constant is a any numbers store.
it is 4 types.

(i) Integer → it's hold only numbers → {1, 2, 3, 4, 5 - - ,}

(ii) Real → its hold only real value / no. → {1.2, 2.0 - - 9.0}

(iii) Character → its hold only alfabats → {'a'}

(iv) String → its hold One - more words → {'Swati'}

3. ~~Key~~ Date types

Data-type specify the different sizes of value that can be stored in the variable.

DATA TYPE

Primitive

(i) Byte → (1 byte or 8bit)
(ii) short → (2 byte or 16bit)

int const,

(iii) int → (4 byte or 32 bit)
(iv) long → (8 byte or 64 bit)
(v) float → 4 byte or 32 bit)
(vi) double → 8 byte or 64 bit)

Non-Primitive

~~Char → (2 byte or 16 bit)~~
Boolean → (1 bit)
Array
Class
string

Integer → byte, int, short, long
Real → Float, double.
Char → (2 byte or 16 bit)

4. keywords :-

Java keywords are also known as reserved words. These
Predifined word by Java so They can't be used as a veriable or objectt name o
class name.

* INSTRUCTION :-

(i) Data declration ~~instruction~~ :-
which is used to declare a variable by specifying its data type and name. int x;

(ii) Initialization :-
Given initial value to a variable.

Ex - int x = 5

if the datatype and declration written in same line That is known as dynamic initialization.

int x = 5.

(iii) Input Output instruction :-

For input instruction → Scanner class

for output instruction → Print.

System. Out. Print - Only print used for output.

(iv) Arithmatic operation-

This instruction used ~~for~~ to perform mathematical operation

Operator = Add, Sub, +, -, *, %, ++, --, %,

	Operation	Associative,	Procedence
( )	Function call	Lelf to Right	.14.
[ ]	Array subcript		
.	Dot (Member of Structure)		
->	Arrow (Member of structure)		
!	Logical not	Right to left	13
~	One's complement		
-	Unary minus (-ve)		
++	increment		
--	decrement		
&	address of		
*	Indirection		
type	Cast		
sizeof	sizeof		
*	Multplication	Left. to Right	12
/	devision		
%	Modules (Remainder)		
+	addition	Left to Right	11
-	subtraction		
<<	left shift		10
>>	Right shift	''	
<	less then <= lessthan equ		9
>	greater then >= Grethen equ	''	
==	equal to		8
!=	Not equal to	''	
&	Bitwise AND		7
^	Bitwise XOR	''	6
|	Bitwise OR		5

&&	Logical AND	left to Right	4
||	Logical ~~conditional~~ OR	"	3
?:	Condition	Right to left	2
=, +=, *= etc	Assignment operators	"	1
,	Comma	left to Right	0

Operand :- A value involved in an operation is called an operand.

$\underbrace{2 + 3}$ are operand.

opcode :- An opcode specifying the operation to be performed.

$2 \underbrace{+} 3$ opcode

<u>Modules</u> (%)

Ex ⇒ $x = 5 \% 2;$

$x = 1;$

$2 \overline{)5(}2$
$\underline{4}$
1 → remainder

Precedence and Associative

Two types Unary operator -

Pre = ++ (increment)

Post = -- (decrement)

Ex : $x = 6;$

$x++ \quad (x+1)$

$x = 7$

$x = 5$

$x-- \quad (x-1)$

$x = 4$

~~Java Arithmetic operator Example~~

Java Unary operator.

The java unary operator require ~~any~~ only one operand. Unary operators used to perform various operation . .

increment (++) ,
decrement (--),
negating an expression
inverting the value of a boolean.

Ex-
```
 public class operator Example
 {
  public static vaid main (string []args)
 {
   int x=10
  System.out.println (x++); // 10(11)
  System.out.println(++x); // 12
  System.out.println (x--); // 12(11)
  System.out.println (--x); // 10.
}
}
```

Output → 10, 12, 12, 10

Arithmatic operator

It is Used to perform addition, Subraction, multipl
division . They act as basic mathematical operat

Ex -
```
   Pubic class OperatorExample {
   publi static vaid main (string args[]) {
  int a=10;
  int b=5;

 S.O.Pln (a+b); // 15
 S.O.Pln (a-b); // 5
 SOPln (a*b); // 50
```

```
sopln (a/b); // 2                                Output → 15,5,50,2,(
sopln (a%b); //0
}
}
```

Ex→ ~~Syst~~ public class Oprator Example {

```
        public static void main (String args[]) {
    System.out.println(10* 10/5 +3 -1* 4/2);
```

out put - 21

left shift (<<)

It is used to shift all of the bits in a value to the left side of specified number of times.

```
Ex→  System.out.println (10<<2); // 10*2^2 = 10*4 = 40
     sopln (10<<3); // 10* ~~2~~*3 = 10*8 = 80
     Sopln ( 15<<4); // 15* 2^4 = 15*16 = 240
  }}
```

Right shift >>

It is used to move the value of the left operand to right by the number of bit specified by the right opera

```
 Ex→  sopln ( 10>>2);    // 10/2^2 = 10/4 = 2
       sopln (20>>20);  // 20/2^2 = 20/4 = 5
      } }
```

JAVA AND oprator ~~and~~ logical && and Bitwise &

The logical && operator doesn't cheak the second condition if the First condition is false.

The bitwise & operator alway cheak both condition ~~whe~~ wheter 1st is true or false.

Example

```
int a= 10 ;
int b= 5 ;  — False
int c = 20 ;
sopln (a<b && a<c) ; false && ~~true~~ = false.
sopln (a<b & a<c) ; False & true = False
```

6 OR operator : Logical and bitwise

(1) The logical || operator. doesn't check the second condition if the first condition is true. . it check the second condition only if the first one is False.

(2) The bitwise ~~ope~~ | operator always check both condition whether first condition is true or false

```
Ex- int a = 10;
    int b = 5 ;
    int C = 20 ;

sopln (a>b || a<c); // true || true = true.
sopln (a>b | a<c) ; // true | true = true.
// || vs |

sopln (a>b || a++ <c) ; true = true
sopln (a) ; // 10 because second condition.

sopln (a>b | a++ <c) ; true | true = true
sopln (a) ; // 11 because 2nd condn
```

ξ
3

Ternary operator

It is used as one line replacement for if-the-else statement and used a lot in Java program.

It is only conditinal operator which takes 3 operands

```
Ex- int a=2;
    int b=5;
  int min = (a<b) ? a:b;
  system.out.println(min);
 }
```

output - 2

Java Assignment operator

It is one of the most common operator.

It is used to assign the value on its right to the operand on its left.

```
Ex- int a=10;
        b=20;

  a+=4;     // a=a+4 (a= 10+4)
  b-= ;     // b= b-4 (b=20-4)
  sopln (a);      14
  Sopln (b);      16
```

Type casting

Type casting is when you assign a value of one Primitive data type to another type.

There are two types of casting :

Widening Casting (automatically)

It automatically converting a smaller type to a larger type size.

byte → Short → char → int → Long → float → doubl
8 bit 16 bit 16 bit 32 32 ~~64~~32 64

Narrowing Casting

Manually converting a larger type to a smaller size.

double → float → long → int → char → short
byte.

Syntex
datatype 1

Ex–

```
double x = 7.230;
int = x;
Output = Error
```

```
* double x = 7.230
      int y = (int) x;  → manually
  sopln (x);
  sopln (y);
}
}
```

output

x = 7.230

y = 7.

Widening

~~Short x~~ =
byte x –
short ·

Ex — ~~public class~~
Cas

byte → shon

```
public class Type casting
{
public static void main (String []args){
{
 byte x = 2;
 short y = x;
}
}
```

Norrowing Conversion

byte → short

Short → byte

short x = 2;
bgte y = ~~byte(x)~~(byte) x;

Control Flow statement :

if , else
else , if
~~else~~ if else if ladder
Nasted if .

Systex For if

```
if (condition)
{
    ______
    ______ code
    ______
}
```

Syntex for else

```
if condition
{
    ______
    ______    code
    ______
}
 else
   {
     code
   }.
```

Systex of ~~eg.~~ if-else

```
~~if~~ if (condition)
    {
        ======== code
    }
    else if (condition)
    {
        ======== code
    }
    else
    {
        ======== code
    }
```

1. Write a Java program to fine out the given number is positive or nagative.

```
Public class Print
{
Public static void main (String [] args)
{
int x = 5;
if (x > 0)
{
System. Out. Println (The no. is positive);
}
~~if~~ else if (x < 0)
{
System. Out. Println (The no. is negative);
}
else
{
System. Out. Print ln (The no is ~~net~~ neutral);
}
}
}
```

O/P

-ve or +ve.

given number is Odd

```
Public class Sum
{
Public static void main (String [ ] args);

int a = 2;
int b = 3;
int x = 7
int y = 2
int i = 6
int j = 9

~~System.out.Prin.~~
System.out.Println ("Sum of a and b is"+a+b );
System.out.Println ("Sum of x and y is"+x+y) ;
System.out.Println ("Sum of i and j is"+(i+j));
}
}

int a = 2
int b = 3
int x = 7
int i = 6
int j = 9

sopln ("sum of" + a+"and"+b+ "is" + a+b);
sopln ("sumof" + x+"and"+ a +"is" + (x+a);
sopln ("sumof" + i + "and" + j +"is"+ (i+j);

}
}
```

8. x = 10 (positive or nagitive

x = 7 (odd or even)

x = 3 (positive or nagitive)

```
public class Print
{
public static vaid main (String [] arg)
{
 int x ~~ce~~ = 10 ;
~~System. out. print ("Enter the number")~~;
if (x > 0)
{
 System. out . Println( The no. is positive);
}
else
 {System . Out . Println ("The no. is nagitive");
 }

int x = 7
 if (x % 2 == 0)
{
 System . Out . Println ("The no. is even");
 }
 else
 {
 System . out . Println ("The no. is odd");
 }

int x = 3
 if (x > 0)
 {
 System . out . Println ("The no. is positive");
 }
 else
 {
 sopln ("The no. is -ve")
```

Unit-3.

OBJECTS AND CLASS

CLASS :-

(i) Class is a user defined datatype

(ii) class is a collection of object and it doesn't take any space on memory.

(iii) Class is also called Blueprint / logic entity.

Class :-

Pre-defined	User defined
→ Scanner	→ Dog
→ Console	→ A
→ System	→ Test
→ String	→ Demo

→ Before we create an object, we first need to define the class.

Syntax :-

```
Class ClassName
{
    ________
    ________    // data
    ________

    ________
    ________    // method
    ______
}
```

OBJECTS :-

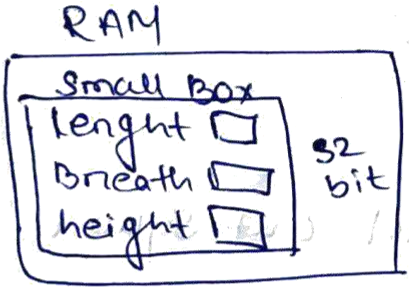
Object is an instace of a class that execute the class. ~~One~~ Onece the object is create, it take space like other variable in memory.

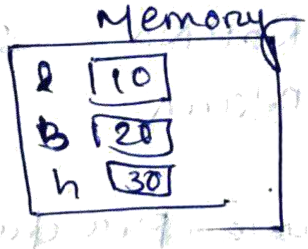
Syntex :- Class name object name = new class na
()

class Program in Java :-

```
class Box
Private int length , Breath , height ;
Public void setDimension ( int l, int b, int h);
{
 lenght = l; breath = b;* height = h ;

}
Public void showDimension ()
{
System.out.println ( lenght );
sopln (breath);
sopln (height);
}
Public static void main (String []args)
{
  Box     Small Box   = new Box ();
  class   object        new class
  Small Box. setDimension (10, 20, 30);
  Small Box . showDimension ();
 }
}
```

O/P





lenght - 10
Breath - 20
height - 30.

∴ Parameter value pass to defination is known as Call by value.

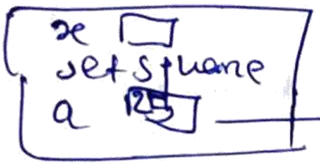
```
class Student
{
 Private String name, int RollNo, int mark;
 Public void setDetails ( String n; int R; Mark)
 {
     name=n; RollNo =R, Mark = M;
 }
 Public void showDetail ()
 {
   sopln (name)
   sopln (RollNo)
   sopln (Mark)
 }
 Public static void main (String []args)
 {
   Student swati = new student ();
   swati. setDetails ("Pooja", 20, 340);
   Swati . show Details ()
 }
}

 O/P ⇒   name - pooja
         RollNo - 20
         Mark = 340
```

* Write a Java program to find out square of a number :-

```
class Square
{
Private int x;
Public void setsquare (int a)
{
  x = a;
}
}
```

```
Public void showsquare ()
{
 sopln (x* x)
}
 Public static void main (String []args).
{
square. swati = new square ();
swati. setsquare (5);
swati . showsquare ();
}
}
```

O/P →



First assign then shift the value to x

O/P - 25.

```
class percentage
{
    Private float x;
    Public vaid setpercentage


class Student
{
Private int CSA, OOM, EVS , DE ;
Public vaid show Avg ( int C , O , E , D;
{
  CSA = C ; OOM = O ; EVS = E ; DE = D ;
  double avg , percentage ;
 int securedMark ;
securedmark = CSA + OOM + EVS + DE ;
avg = secured mark / 4 ;
```

Percentage = $\frac{\text{secured mark}}{\text{total mark}}$ * 100 ;

```
sopln (" percentage is" + percentage );
}
Public static void main ( String [] args ) ;
{
```

```
~~Ram~~ Student Ram = new Student ();
  Ram . ~~show~~ Setavg = (50, 30, 70, 75) ;
  Ram . showavg ();
 }
}
```

Output :-

METHODS :-

(i) Method is a group / block of code which take in from the User, Processed it & give output.

(ii) Method represent the state and behaviour of the object respectively.

(iii) Method are Use to Perform some operation.

(iv) Method run only when it called.

Syntex :-

```
return-type function name (Parameter)
{
    ======  Statement
}
```

Why we Use method.

(i) Decrease line of code

(ii) Redability (~~easi~~ easily to understand)

(iii) Repeatation (

ex
```
class Becycle
{
 state or field //
 Private int year =5;
// behavior void breaking ()
{
 sopln ("working of Breaking");
}
}
```

```
class Lamp
{
// Store the value for light
// true if light is on
// false if light is off
boolean is on;
// Method to turn on the light
  void turn on()
 {
  is on = true;
System.out.println (~~[illegible]~~("light o
}
// Method to turn off the light
 void turn off ()
{
   is on = false ;
 System.out.println ("light on ?" + is on)
}
}
class main
{
 Public static void main (String []args)
{
// Create object led and halogen
   Lamp led = new Lamp ();
   Lamp halogen = new Lamp ();
// turn on the light by
// calling method turn on()
   Led. turn on();
// turn off the light by
// calling method turnoff()
halogen.turnoff();
}
}
}
```

Scanner class :-

Scanner class is pre-difined class in java whic is ~~as~~ available in java. util package.

It is used to get user input.

Rule :-

* if we use scanner class, must have to create. object of scanner class.

Syntex :- Scanner object name = new Scanner (system.

② Scanner class method

```
(i) nextLine ();        // string
    nextInt ();         // Integer
    nextfloat ();       //floating
    nextBooleam ();     // True or false
    next Double ();     // double
```

③ Import scanner class package of the top line of of the program

Syntex :- import java. Util Scanner ;

~~④ Wrong input~~

Write a java program to take user value.

```
Importx Java. util. scanner;
 Public class TakeInput
{
 Public static void main (String []args).
{
 system .out.printen ("Taking input from user");
 Scanner Sc = new Scanner (system.in);
 System. Out. Println ("Enter 1st no");
 ~~Scanner~~ int a = sc. next Int ();
 system. out. printen ("Enter 2nd no");
  int b = sc.nextInt();
int sum = a+b;
system.out.println("sum of" +a + "and" +b + "is:"+sum
                                                      );
}
}
```

output → Taking input from user

```
Enter 1st no  20
Enter 2nd no 30
Sumof 20 and 30 is:50
```

Accessing class member :-

```
Class Helloo.
{
   Public int a,b,c;
   Public static void main (String []args)!
   {
     a=5;
     b=6;
     c=a+b;
    System.out.println(c);
    }
   }
   output → 11
```

~~Int~~ Instance data and class data :-

* Class Data in terms of static class is the data that particular class holds in its structure.

* Instance data can refer to different object of the same class that hold different value using the same class structure in memory heap.

Instance data

```
class Test
{
  int mark;
}.
```

belong static

Class data

```
class Test
{
  static int mark;
}
```

not belong any object

Static variable :-

→ A variable which is declared with the help of static keyword called static variable.

syntex :- static intx;

→ It is are by default initialize to its defoult value

→ It is has singal copy for the whole class and doesn't
→ depend of the object.

```
class Student
{
int Rollno, String name;
static string college = "JES"
Student (int r, String n)
{
 RollNo = r;
 name = n;
}
 void display();
 {
  System.out.println (rall no + "  " + name + " " + college)
  }
 }
 Public class Test
 {
  public static void main (String [] args){
  {
  Student  s1 = new Student (31, "Swati");
  Student  s2 = snew student (21, "pooja");
  }
 }
class Counter
int count=0;
counter ();
{
count ++;
System.out.println(count);
}
Public. static.void main (String args[])
{
 counter c1 = new counter ();
 counter c2 = new counter ();
 counter c3 = new counter ();
}
}
}
```

Output = c1 = 1 , c2 = 2 , c3 = 3

count = ~~0~~ ~~1~~ ~~2~~ 3

CONSTRUCTOR :-

constructor is a special type of method whose name is same as class name.

- The main purpose of constructor is initialize the objec
- Every java class has a ~~cont~~ constructor (defalt).
- A constructore never contain any return type include void.
- A constructor is automatically called at the time of creation.

Systex :

```
Class class name
{
    class-name ( ) → constructor
    {
    }
}
```

Write a Java program to use constructor :-

```
Class A
{
    int a; string name;
    A()
    {
        a=0; name="null";
    }
    void show()
    {
     System.out.print(a+"  "+name);
    }
}
Class B
{
    Public static void main (String args[])
    {
    A ref = new A();
    ref.show();
}
```

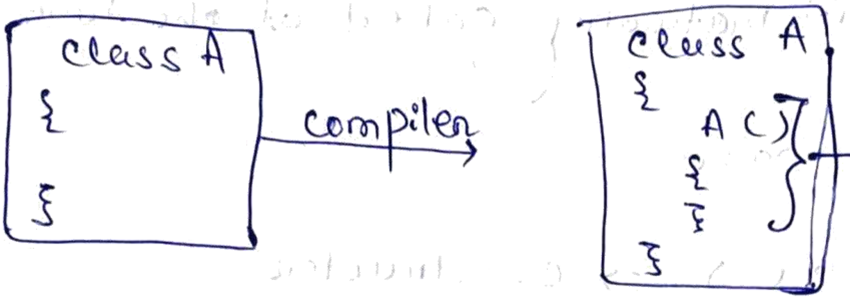
O/P = 0, null

Default Constructor :-

A constructor which does not have any paremeter is called default constructor.

Syntax :-

```
class A
{
   A() → No any perameter.
   {
      ____
      ____
   }
}
```



compiler automatic add costructor par (initialize)

```
class A
{
  int a; string b; boolean c;
  A() // default
  {
    a =1000; b="swati"; c=true;
  }
  void Display ()
  {
   system.out.println ( a+" " +b+" "+c);
  }
}
class B
{
  Public static void main (String []args)
  {
    A r=new A();
      r.Display ();
  }
}
```

∴ new is keyboard using create object

O/P = 1000 . Swati True.

Parameterized constructor :-

A constructor through which we can pass one or more parameter pass is called parameterized constructor

Systex :-

```
class A
{
    ~~class~~A( int x , string y)}
   {
     ____
     ____
     ____
   }
}
```

```
Class  A
{   int x ,y;
   A (int A , int b ) //parameterized
   {
     x= a;    y=b;
   }
   void show ()
   {
    System.out.println(x+"  "+y);
   }
 }
 class B
 {
   Public static void main (string []args)
   A r=new  A(100, 200);;
   r.show ();
  }
}
```

O/P = 100   200

ek ya ek se Jayde
parameterized const
class mei bana skt
hai (Yes)   dat
another per- ~~take~~ take anoth

COPY CONSTRACTOR :-

Whenever we pass object reference to the constr then it is called copy constructor.

```
class name (obj)
{
    Class name (obj ref)
    {


    }
}
```

* another constructo data copy.

ek object ka same content ko dusre object mein copy by the (obj ref)

```
Class A
{
   int a ; String b ;
   A ()
   {
     a = 10 ; b = " swati" ;
   }
   A (A ref)          ⟶ Create other contructor to
   {                         copy .
      a = ref . a ;
      b = ref . b ;   } — copy
      System . out . println (a + "   " + b) ;
   }
}

 Class B
{
   Public class void main (String []args)
   {
     A r = new A () ;
     A R2 = new A(r) ;
   }
}
```

```
O/P =   10   swati
        10   swati
```

Private constructor :

In java, it is possible to write a constructor as a private but according to the rule we can't access private member outside of class.

syntex :-

```
class    class name
{
    private class name ()
    {

    }
}
```

```
class A
{
  int a; double b; String c;
  private A()
   {
     a=10;  b=30.56;  c="swati";
     System.out.println(a+" "+b"  "+c);
   }
{ public static void main (String [] args)
  {  A r = new A();
  }
}
```

O/P = 10 30.56 swati

Access~~o~~ Modifier :

There are two type of modifiers in java
(i) Access modifier
(ii) Non Access modifier

→ The Access modifier in java specifies the accessib or scope of a field, method, constructor or class

→ We can change the access level of ~~feild~~ field, constructor, method, and class by appling the access modifier on it.

There are, four type of java access modifier

(1) Private :-

The access level of a private modifier is only within the class. it connot be accessed from outside the class.

2 Default :-

The access level of a ~~[illegible]~~ default modifier is only within the package. it connot be accessed from outside the package. if you do not specify any class (level, it will be the default.

3. Protected Modifier

The protected modifier is with in the package and outside the package through child class. if you do not make the child class, it connot be accessed from outside the package.

1. Public :-

Public modifier is access everwhere. it can be accessed from within the class, outside the class, with package and outsid package

			Outside class	
Access Modifier	within Class	within Package	Outside package by sub class only	Outside Package
Private	Yes	No	No	No
Default	Yes	Yes	No	No
Protected	Yes	Yes	Yes	~~Yes~~ No
Public	Yes	Yes	Yes	Yes.

There are many non-access modifier, such as static, abstract, synchronized, native, volatile, transient, etc

Private :

Using Java Object

What is String in Java ?

String is an object that represent a sequance of characters in java.

The java.lang.string class is used to creat a string object.

String Builder class :-

Java String Builder class used to create muteble (modifiable) string.

Constructor of String Builder class :-

String Builder() → It creates an empty string Builder with the initial capacity of 16.

StringBuilder(String str) → Creates a string Builder with the specified string

(int length) → It create an empty string Builder with the specified capacity as length.

String Builder Method :-

Public string Buider append string (s) :-

It is used to append the specified string to with this string.

append (char), (boolean), (int), (float), (double) etc.

StringBuilder append() Method :-

It given argument with this String.

```
 class StringBuilderExample {
 Public static void main (String args[])
{
  String Builder sb = new StringBuilder ("Hello");
  Sb. append ("Java"); // Original string is changed.
  System. out. println (Sb);
 }
}
```

O/P → Hello Java

StringBuilder Insert () Method :-

Insert the given string with this string at the given position

```
      Class Example 2
     {
      Public static void main (String []args)
    {
    Example2 e = new Example2 ("Hello");
     e. insert (1,"Java");
    System.out.println (Sb);
  }
 }
```

O/P → HJavaello

String Builder replace () Method.
The given string From the Specific beginIndex and endIndex replaced it.

```
  Class Example3
 {
 Public Static void main (String []args)
 {
 ~~String~~ Example3 e = new Example3 ("Hello");
e ~~Sb~~. replace (1, 3,"Java");
 System. Out . println (sb);
}
}
```

O/P → Javaelo

Delete () Method
It delete the string From the specified beginIndex to end Index.

```
  Class Example4
 {
 Public Static void main (String args [])
 {
 Example4 e = new Example4 ("Hello");
e. delete (1,3);
 System .Out . println (sb);
}

}
```

O/P → elo

reverse () Method

It reverse the current String.

```
  class A
{
 public static void main (String []args)
{
  A e = new A ("Hello");
  e. reverse ();
  system.out.println (Sb);
 }
}
```

O/P → olleH

Capacity () Method

→ It return the current capacity of the Builder.

→ The default capacity of the Builder is 16.

→ if the number of character increase from its curr capacity, it increases the capacity by (oldcapacit

Example → (16*2)+2=34

```
   class B
  {
  PSVM (  )
 {
  B x = new B ();
  System.out.println (x.capacity()); //default 16
  x.append ("Hello");
 System.out.println (sb.capacity()); // Now 16
 x.append ("Java is my favourite language");
 System.out.println (x.capacity()); // (16*2)+2=34
 }
}
```

```
         16
O/P →    16
         34
```

Java String Buffer class :-

It is used to Create mutable (modifiable) string object

* It is a thred-safe i.e Multiple threads connot access.

Constructor :-

String Buffer() → It create an empty string Buffer with the initial capacity 16

(String str) → It create a string Buffer with the specified ~~capacity~~ string.

(int capacity) → It create an empty String buffer with the specified capacity as lenght.

What is mutable string ?

A String that can be modified or changed is known as Mutable string.

String Buffer and String Builder ~~classes~~ classes are used to creating mutable string.

String Buffer	String Builder
String Buffer is synchonized i.e thread safe.	(1) String Builder is non-Synchronized i.e it not thread safe.
String Buffer is less efficient than String-Builder	(2) String Builder is more efficient than String Buffer
String Buffer was introduced in java 1.0	(3) It is introduced in Java 1.5

Ex→

```
Public class Test
{
PSVM( )
{
Test x=newTest("Hello");
~~buffer~~ x.append("Java");
SOPln (~~buffer~~ x);
}
}
```

O/P - Hello Java

Ex→

```
Public class ~~Builder~~ Test
{
PSVM( )
{
~~St~~Test x = new Test ("Hello");
x. append (" Java");
System.out.println ( x );
}
}
```

O/P → Hello Java

It is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is a important part of OOPs (object oriented programing system).

Inheritance represents the IS-A relationship which is also known as a parent-child relationship.

why use inheritence in Java

For Method overriding (so runtime polymorphism can be achieved)

For code Reusability.

Term used in inheritence :-

class :-

A class is a group of object which have common properties.

It is a template or blueprint from which object are created.

Sub class / child class :-

It is a class which Inherits the other class.

It is also called a drived class, extended class or child class.

Super class / parent class :-

where a subclass inherits the features.

It is also called as base ~~calles~~ class or a parent class.

Reusability :-

It is a mechanism which facilitates you to reuse the field and methods of the existing class when you create a new class.

You can use the same fields and method already defined in the previous class.

```
syntex :- class Subclass-name extends Superclass-name
          {
              // method and field
          }
```

extend ~~es~~ keyword indicate that you are making new class that ~~da~~ derives from an existing class.

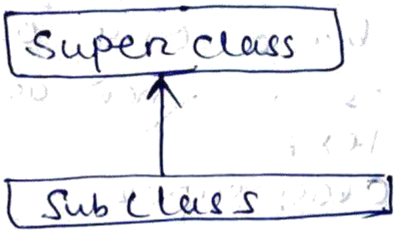
*extends meaning → increase the functionality.

TYPES OF INHERITANCE :

1. Simple Inheritance :-

which contain only one super class and only one subclass is called simple inheritance.

```
class super~~class~~
{
    DATA
}
 class sub extends super
{
    DATA extends
}
```



```
Class One    // Super class
{
 Public void print~~es~~ Hello ~~es~~
 {
  System.out.println ( "Hello);
  }
}

class Two extend one    // Sub class
{
  Public void print World
  {
   System.out.println ("world");
  }
}
```

```
Public class main
{
  Public static void main (String []args)
  {
    Two x = new Two ();
     x. Println Hello ();
     x. print ln world ();
  }
}
```

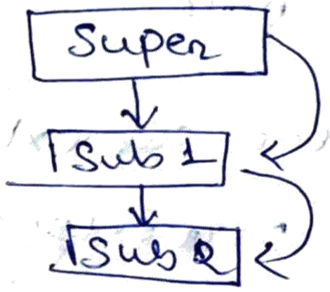
O/P → Hello
      World.

Muti-Level Inheritance :-

We have only one super class and multiple sub class is know as Multilevel Inheritance

Syntex :-

```
Class super
{

}

class sub 1 extend super
{

}

 class sub2 extend ~~super~~ Sub1
{

}
```



```
Class One
{
  Public void print Hello ()
  {
   System. out. Println ("Hello");
  }
}
```

```
class Two extend ~~class~~ One
{
 public void print.world()
 {
  system.Out~~put~~.println ("world");
 }
}

class Three extend Two
{
 public void printswati ()
 {
  System.Out.println ("swati");
 }
}
public class Main
{
 public static void main (string []args)
 {
  Three x = new Three
   x. print Hello ();
   x. print world ();
   x. print swati ();
 }
}
```

```
O/p> Hello
     world
     swati.
```

Multiple Inheritance :-

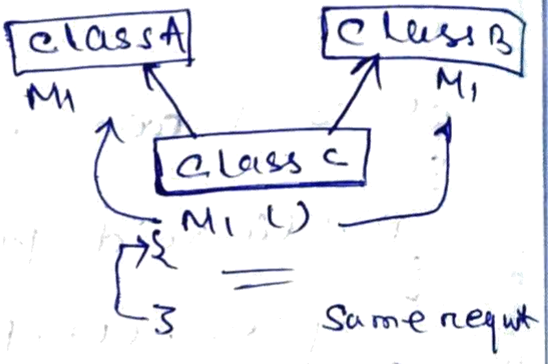
Why multiple inheritance is not suppoted in Java?

To reduce the complexity and simplify the languag Multiple inheritance is not suppoted in java.

We can achieve multiple inheritance through interface becouse interface contains only abstract method, which implimentation is provided by the sub class.

class C extends A, B ✗ ("class inherite then write extends.)

class c impliment A, B ✓ (Interface method implime then write impliment)

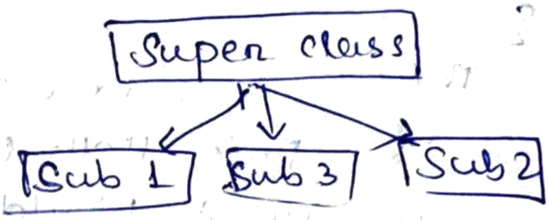


Hierarchical Inheritance :-

which Contain only one super class and Multiple Sub class and all sub class directly extend Super class' called hierarchical Inheritance.

Syntex :-

```
Class A
{
  =
}
```

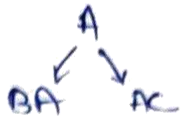


```
class B ex A        | Class c extends A
{                   | {
   A data exten     |     A data extends
}                   | }
```

Code reuse

```
class A
{
 Public void print swati ()
 {
  System.out.println("swati");
 }
}
Class B extend A
{
  public void print ~~Roll At~~ 25 ()
  {
   System.out.print ("25");
  }
}
```

```
class C extend A
{ public void print IT ()
  {
   system.out. println ("IT");
   }
}
& ~~class void main ()~~
 & Public Class Main ()
 {
  Public static void main (String [] args)
  {
   ~~class~~ C x = new C ();
    x. print Swati ();
    x. print 25 ();
    x Print IT ();
  }
}
```

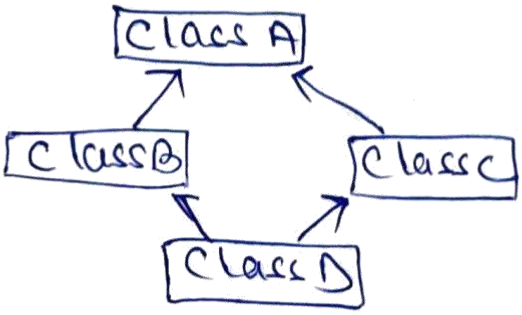


O/P → Swati →A
25 —(B)
~~BA~~. Swati →A
IT. c→

→ HYBRID INHERITANCE -

It is the combination of more than One type of Inheritance is called hybrid Inheritance.

∴ simple + Multi level = hybrid.



```
Class A
{
   member base class
};

Class B
```

CH-6 Polymorphism

Polymorphism :-

Polymorphism meening is same way function having different object and different result are called as polymorphism.

It is basically two type.

i) compliile time polymophism

ii) Run time polymorphism.

Compliile time polymorphism :-

A polymorphism which is exist at the time of compilation is called compliile time or early binding or static polymorphism.

Method Overloding

Whenever a class contein more the one method with same name and different type of parameter called method overloding

ex - i) display (int a)
display (float a)

Method overloding increases the readability of the program

ii) void display (int a)
int display (int a)

(iii) display (int a)
display (int a, int b)

Syntex :- return type method name (parameter 1)
return type method name (parameter 1, 2)

```
Class A
{
  void add ()
  {
      int a=10, b=20, c;
       c=a+b;
       system.out.println(c);
  }

   void add (int x, int y)
   {
       int c;
       c= x+y;
      System.out.println("c");
   }

     void add (int x, double y)
     {
        double c;
        c = x+y;
      System.out.println (c);
    }

     Public static void main (string [] arg)
    {
     A r=new A();
     r.add();
     r.add(100,200);
     r.add(50, 45.32);
   }
}
```

Method is one (ad
but behavior is
different

```
O/P -   30
        300
        95.32
```

Runtime polymorphism :-

A polymorphism which exist at the time if execution OF program is called run time polymorphism.

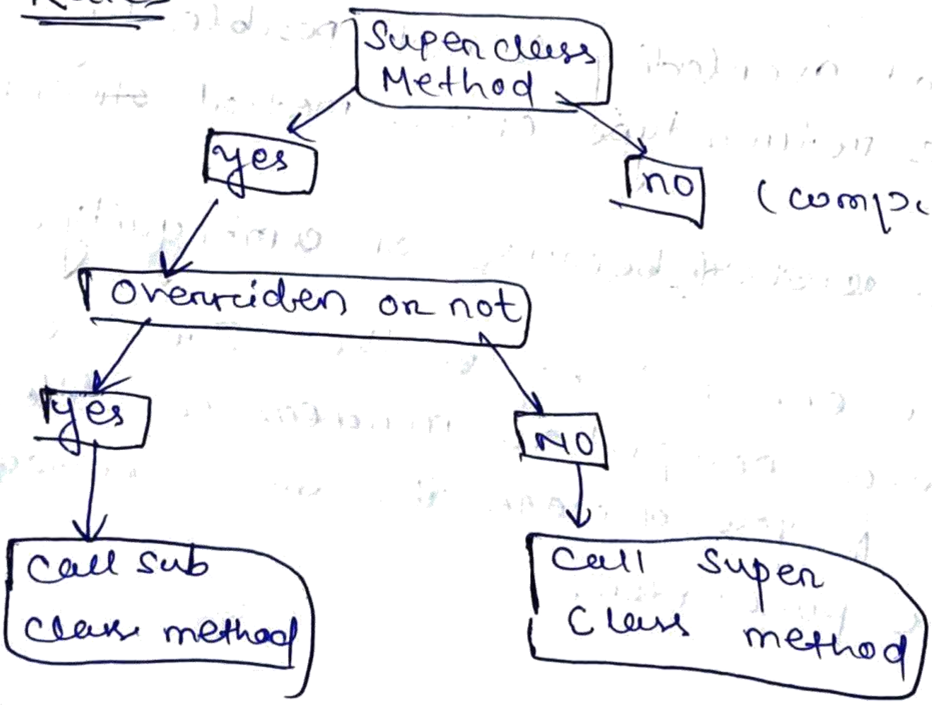
Method Overriding :-

Whenever we write method in super and sub classes in such a way that method name and parameter must be same called method Overriding.

Syntex :-
```
class A
{
  void show()
  {
     =
  }
}
class B extend A
{
  void B
  {

  }
}
```

Method overring we can not perform by using Inheritence.

Rules



Rules :-

Method must have same name and parameter as parent class

There must be an IS-A relation ships (inheritance)

```
Class shape
 {
   void draw()
  {
   sopln ( "Can't say shape Type");
  }
 }
```

parent class

```
 class Square extends Shape
 {    override
      void draw ()
    {
      sopln ( "square shape");
    }
 }
```

Subclass

```
 class Demo
 {
   ~~PVS~~ PSVM ( static [] args ) {
   Shape r = new shape ();
   r. daraw ( );
 }
```

O/P → Square shape.

We Override static Method ? why ?
NO, becouse the static method bound with class wheres instance method is bond with an object

Method overriding	Method overloding.
→ Method overrinding is used to provide the specific implementation of the method that is already provided by its super class.	Method overloding is used to increase the readability of the program.
→ Its occurs in two class that have IS-A relation ship. (Inheritance)	It is performed within class.
~~Incase of~~ → Parameter must be same.	parameter must be different.
→ It is example of run time polymorphism	It is exam of complile time polymorphism.
→ Return type must be same or covariant in method overriding	It can't performed by changin return type of the method only * return type can be same or different * you must change parameter

PACKAGE :-

Packge is a group of Similar type of classes, Interfaces and sub class.

package is two type.

i) Built-in package or pre defined

ii) User - defined package.

Built-in package, or pre defined

The package which are already Create by java developer are called pre-defined or built-in package.

ex → Java.long , Java.applet, java.awt, Java.io, java.util, java.net, and Java.SQL

1) Java.long :- It is the default package also known as heart of the Java.

becouse without using this package we can't write even a single program, and we need to import this package.

ex → System, string, Object, Intege etc...

2. Java.Util :- It is used to implement data structure of java.

It is contain utility class also know as collection framework.

ex → Linkedlist, stack, ~~[illegible]~~ ~~A~~ etc.

Java.IO :- This package is very usefull to Perform input output operation on File

ex- file, file write, fileReader etc.

Java.awt :- AWt stand for obstract window toolk

It is ~~also~~ used to developed GUT applicatio
The awt program are stand alone ·
(Creation and excution some system) program
& it . contein main () method.

ex- frame , Button , Textfeid etc.

Java.app~~let~~ :- It is also used to developed GUT application.
App~~let~~ programed are web related program Created at server but excuted it client machine

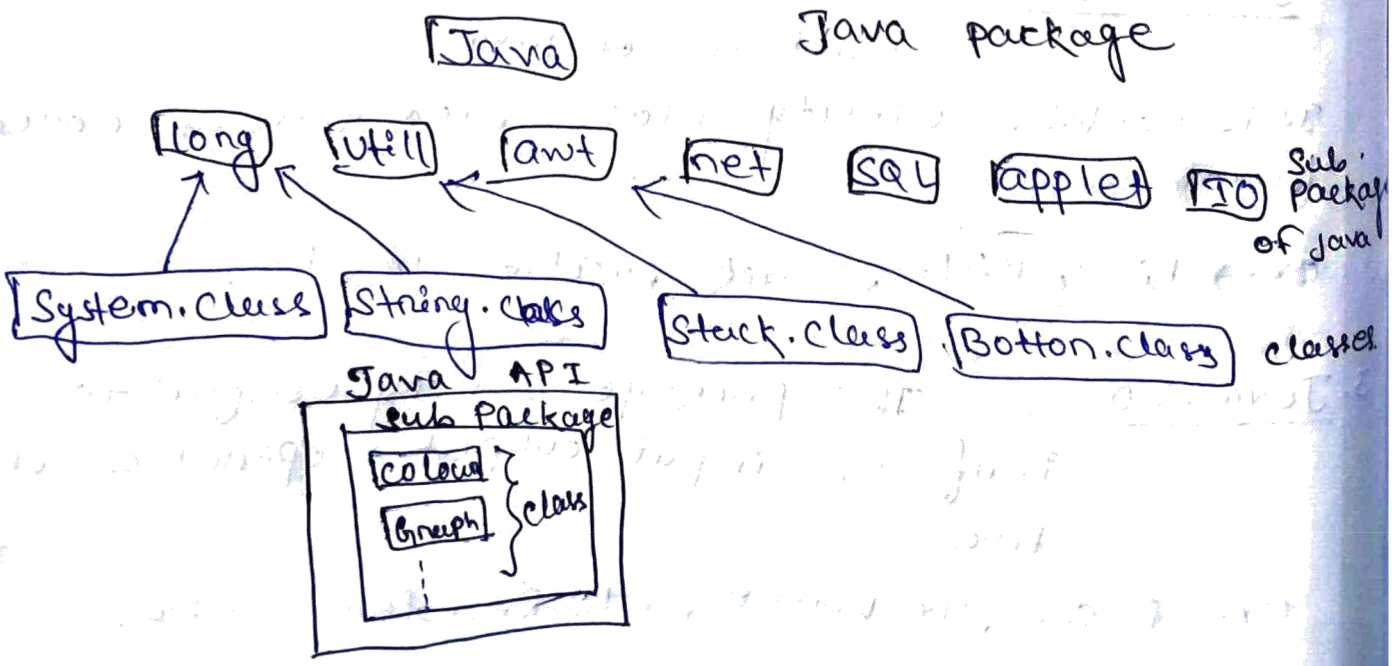
ex → Applet.

Java.net :- It is used to networking parposes

ex → URL, InetAddress , URL connection ~~and~~ etc.

Java.SQL :- It is used to Data base lelated any work.

ex → Connecting , Statement . Resultt elc.



Advantage :-

→ Java package is Use to categorize the class and Interface so that they can be easily maintained.

→ It is provide access protection

→ It's remove the naming collision.

The package keyword is used to create a package in Java.

```
Package mypack ;
 public class Simple
{
  public static void main (String args[])
  {
   System.out.printen ("welcome to package");
  }
}
```

O/P → welcome to package .

Syntex of complile → Javac-d. file name. Java

d → destination

run :- Java mypack. ~~Simple~~ filename

* How to access package from another package ?

1. import package.*;
2. import package.classname;
3. Fully qualified name.

Using packagename.*

if you use packagenam.* than all the classes and interfaces of this package will be accessible but not subpackage.

Ex →
```
  Package pack ;                    (A. Java)
   Public class A
  {
   Public ~~static~~ void msg ( )
  {
   System.out.println ("Hello");
  }
 }

 Package mypack;
 Import pack.* ;
 class B
 {
 Public static void mein (String args [])
 {
   A obj = new A ();
     obj.msg ();
   }
 }
```

Output - Hello

Using packagename. classname.

if you import package.classname then only decla class of this Package will be accessible.

→ Then
accessible.

→ There is no need to import.

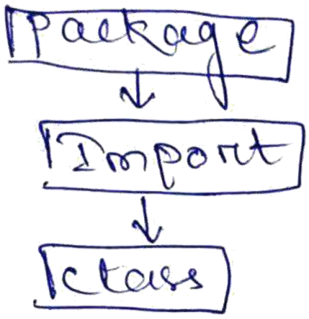
→ But you need to use fully qualified name every time when you are accessing the class or interface.

→ It is generally used when two package have same cla name.

eg. class name java.util and java.sql packag contain Date class.

* if you import a package subpackage will not be imported.

* sequence of the program must be package then import then import then class



Subpackage in Java

package inside the package ~~[illegible]~~ called packa
It should be created to categorize the packag further.

The standard of defining package is domain. company. package

eg. com. javatpoint. bean.

org. sssit. dao.

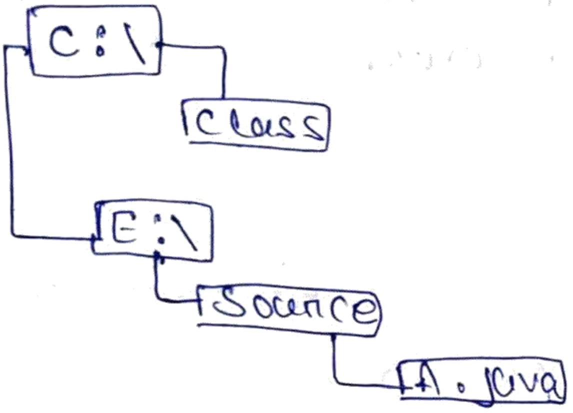
```
Package com. javatpoint. core;
class simple
{
 Public static void main (String args [])
{
 System . out. println ("Hello subpackage");
}
}
```

To compile :- Java - d. simple.java

To run :- Jana com. javatpoint. core. simple.

How to send the class file to another directory On drive ?

There is a scenario, I want to put the class file of A. java source file in class folder of c: drive.



```
Package mypack;
Public class simple
{
Public static vaidmain (String args [])
{
 System. out. printen ("welcome to package");
}
}
```

To compilee :-

```
e:\sources > javac -d c:\ classes Simple.Ja
```

To Rum :-

```
e:\source >set classpath =c:\ classes;
e:\source > Java mypack . simple.
```

Java Static Import :-

The static Import feature of java 5 facilitate the java pragrammer to access any static member of a class direetly.

There is no need to qualify it by the class name.

Advantage :-

less coding is required if you have access any static member of a class oftenly.

Disadvantage :-

if you overuse the static import feature, it makes the pragram unreadable and unmaintainal

```
import static java.long.system.*;
Class StaticImportExample
{
public static void main (String args[J).
{
out.println("Hello"); // no need of system.out
}
}
```

O/P - Hello

Access Modifiens in Jana :-

~~There are two modifien in Java~~

i) Access Modifien

It is specifies the accessibility on scope of a field, Method, construeton on class

we can change the access level Fields, constructon, methods and class by uppeing the access modifien on it.

1. Private :-

The access level of a private modifien is only within the class. it connot be accessed from outside the class.

2. Default :-

It only with the package. it connot be accessed from outside the package.

if you do not specify any access level, it will be the default.

3. Protected :-

It is within the package and outside the package Through child class. if you do not make the child class, it connot be accessed from outside the package.

4. public

The access level of a public modifien is every where. it can be accessed from within the class outsid the class, within and outsid the package

Access Modifier	within class	within Package	outside class	outside Package
Private	✓	✗	✗	✗
Default	✓	✓	✗	✗
Protected	✓	✓	✓	✗
Public	✓	✓	✓	✓

Advantage :- Pakage

(i) Reusability

(ii) Security

(iii) fast searching

(iv) naming conflecting

(v) Hiding.

Dis-advantage

We can't pass parameter to package.

API :-

API stand for application program interface.

It is a set of routines protocals and tools for building software and application.

It may be any type of system like a web-based system, operating system or data base system.

What is File Handing in java ?

File handling in java implies reading From and writing data to a file.

The file class from the java.io package, allows us to work with different formats of file.

Example :- import java. io. file

// Specify the file name

file obj = new file ("Filename.txt");

In Order to use the file class, you need to create an object of the class.

What is Stream ?

Stream is a sequennce of data.

which can be divided into two type.

1. Byte stream :-

This mainly incorporates with byte data. When an input is provided and executed with byte data, then it is called file handing proces with a Byte stream.

Byte stream is also called Binary streams which read and write data in the format of byte.

There are also two type

(i) ByteInput Stream

(ii) ~~ch~~ Byte Output Stream

2. Character Stream :- (unicode)

The character stream which read and write data format of character x is called character stream.

Character stream again divided into two type

(i) character Input stream

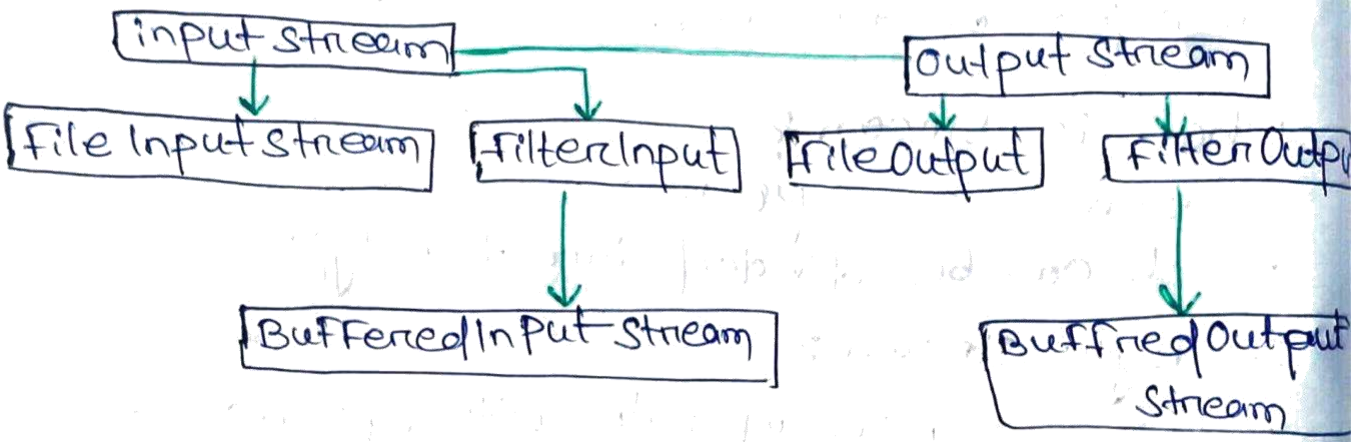
(ii) character Output stream.

Output stream :-

To use Output stream to write data to a destination.

Input stream

To use Input stream to read data from source.



File Methods

The various method that are used for performing operation on java file.

* Can Read ()	Boolean	It tests whether the file is readable and writable or not.
* Can ~~R~~Write ()		
* Create NewFile ()	"	It is to create an empty file.
* Delete ()	"	Delete a file.
* exists ()	"	It test whenther the file exists. (Prenious file is available or not)

* getName()	String	Return the name of the fi
* getAbsolutepath()	"	location of the file
* Length ()	Long	size of the file in byte
* List ()	String	Array of the file in th directory.
* Mkdir()	Boolean	Creates a directory.

File class : An abstract representation of File

(i) File → File is a superclass to all otherfile

(ii) FileReader → it used to Read data from fi

(iii) FileWriter → it is Used to Writer date from

(iv) File InputStream → It is also used to Read data but byte form.

(v) FileOutputstream → It is also used to Write data

(vi) BufferedInputstream → To perform Buffed operation then used to

(vii) BufferedOutputstream → this.

operation of file

(i) Create file

(ii) Read

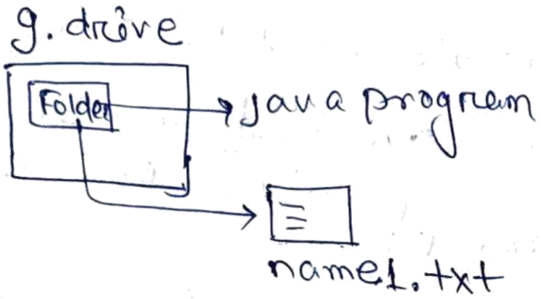
(iii) Write

~~File class~~

~~Import Java.~~

Create File :-

```
Import java.io.file*
class FileExample1
{
Public static void main (String []args)
{
 File f1 = new File("g:/Java program/name1.txt")
 System.out.println("can file Read"+f1.can Write()
System.out.println("Is file exist"+F1.exists());
System.out.println("file name"+F1.get Name());
System.out.println("Legth of file"+F1.lenght ());
}
}
```



Path → g:/Javaprogram/name1 txt

```
Public static void main (String []args) throws IO
{                                            Exception
File f1=New File ("g:/Java program/name1.txt");
 F1. Create NewFile ();
System.out.println("is exist:"+F1.exists());
}
```

Complite → javac File Example1.java

O/P → Isexist : true

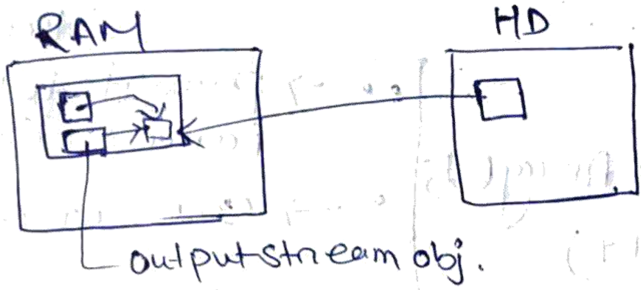
Length of file

```
System.Out.println("File size :" +f1.lenght());
```

O/P → Hello world
11

∴ Writing to file Using FileOutput Stream :-

Writing data to file means storing data in the file.



* FileOutputStream is meant for writing stream of raw.
* file Outputstream is subclass of OutputStream

Constructor

(i) FileOutputStream (File)
Creates a file Outputstream to write to the file represented by the specified file object.

(ii) FileOutputStream (File file, boolean append)
Creates a file outputStream to write the file represented by the specified file object.

(iii) (String name)
~~Crete~~ Create a file to write to the file with the specific name.

(iv) FileOutput Stream (String name, boolean,append)

Example :-

```
    import java.io.* ;
    class file Example
   {
   Public static void main (String []args) throws
                                            IOException
  {
  int i ;
  File OutputStream fout ;
  fout = new file Output Stream ("../ files /name 3.txt",
                                                true);
  String s= "TATA"
  char ch[] = s s.toChar Array[];
 for (i=0 ; i<s.lenght (); i++)
     fout . write (ch[i]);
     fout.close ();
 }
}
```

.. → parant folder Parent directory

. → current directory

NewFile → save (file Example.Java) → open (Notepad ++)

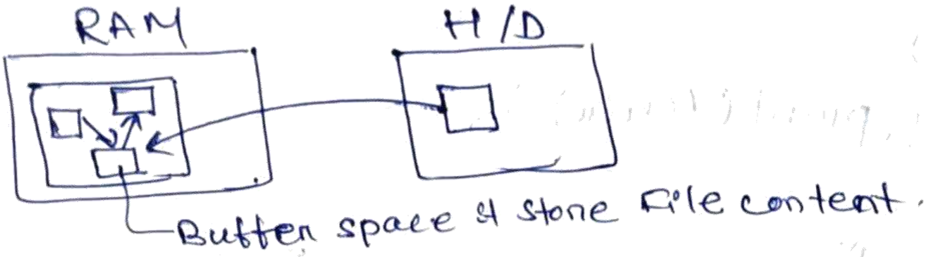
→ coding

Complile → javac file Example.class

O/P → TATA

Reading frrom file

Reading data from mean extracting data strored in the file (without deletting it from the file).



Buffer space & store file content.

File Inputstream

* FileInputstream meant for reading stream of raw byte.

Constructor

* FileInputStream (File 'file)

By opening a connecting to an actual File, the file named by the file object file in the file System

* File InputStream (String name)

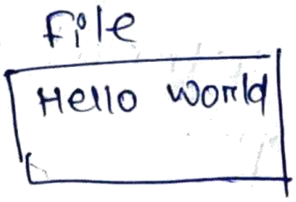
The file named by the path name, name in the file system.

Example :-

```
        import java.io.* ;

class FileExample
{
public static void main (String []args) throws IO
                                        Exception
{
int i ;
FileInputstream f1; ~~fl new FileInputStream~~
f1 = new FileInputstream ("../Files/name2.txt");
```

```
 do
{
  i = f1.read();
  if (i!=-1)
  System.out.print((char)i);
}
 while (i!=-1);
 f1.close();
}
}
```

∴ -1 is special symb

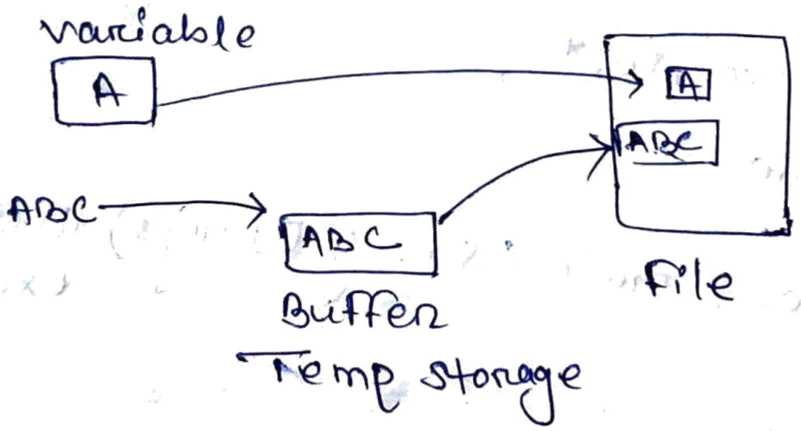


O/P → Hello world

BufferedWriter

→ Writes text to a character-output stream, Buffering character so as to provide for the efficient writing of single character, array and string.

→ The Buffer size may be specified.



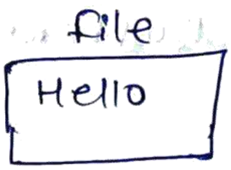
Constructor

Buffered Write (Writer Out)

Creats a buffered character - output stream that uses a default - size output buffer.

Example →

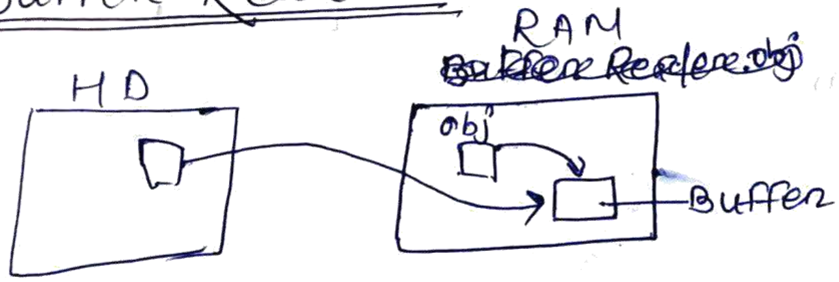
```
        Import java.io.* ;
        class FileExample
    {
    Public static void main (String [] args) throws IOExcep
  {
  BufferedWriter b=new BufferedWriter( ~~new File~~
          (newFileWriter (".. /files/ ~~writer~~ name4.txt
                                                  true)
b. write (" Hello ");
b. close ( );
 }
}
```



Folder → File → File.java → coding

O/P → Hello

Buffer Reader



Read text from a character - input stream, providing for the efficient Buffering character reading of character, ~~string~~ arrays and lines.

The Buffer size may be specified, or the default size may be used.

(i) BufferReader (Reader in)

Create a buffering character-input stream that use a default-sized input buffer.

Example :-

```
        ~~class~~ import java.io.* ;
Public class fileExample
  {
  Public static void main (String []args) throws IOException
 {
   int ch ;
   BufferedReader b=new BufferedReader
                 (new FileReader ("../Files/name 9.txt"));
  while((ch = b.read())!-1)
  {
    System.out.print((char)ch);
   }
   b.close();
  }
 }
```

∴ [-1 indicate end of file]

file

Program

O/P → Program

More Method

- String readline()

  Read a line of text.

```
{
 BuffedReader b = new BufferReader
 (new fileReader ("file1.txt"));
 String S1 ;
 while ((S1 = b = readline())!= null)
 S1 = b. readline ();
 System. out. println (S1);
 b. close ();
}
}
```

File

```
Jharsuguda
Engineering
School.
```

O/P → Jharsuguda
Engineering
School

Exception Handling CH-9

Exception :-

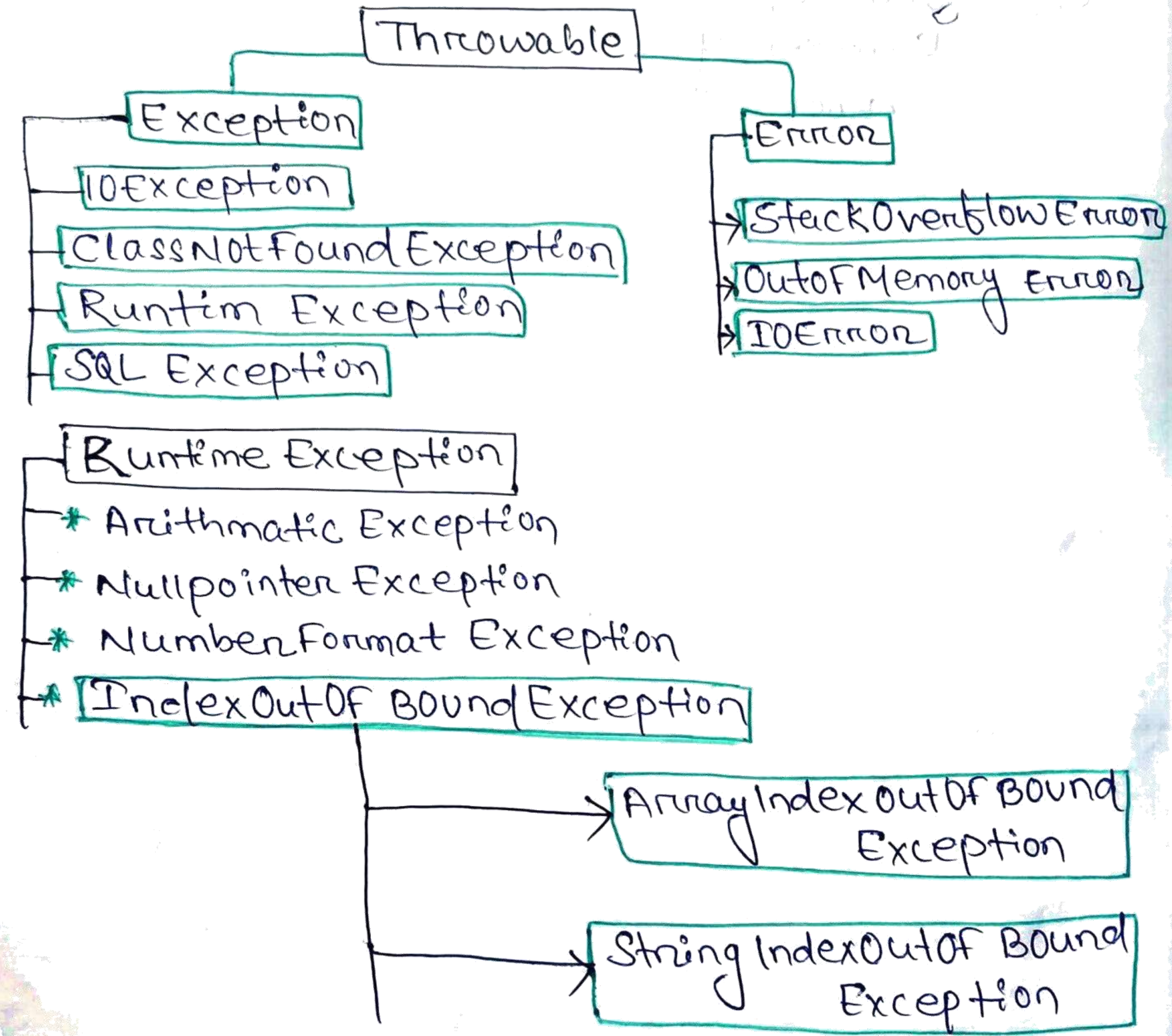
An Exception is unwanted or abnormal Situation that Occured at runtime.

Ex → 100/0 = Undefined

Exception handling :-

Exception handling is used to handle error condition in a program systematically by taking the necessary action.

CLASS HIERACHY :-



1. try catch block

The try-catch block is used to handle exception in java.

Syntex :-
```
try {
   // code
}
catch (Exception e) {
   // code
}
```

* Every try block is followed by a catch block.
* When an exception occurs, it is cought by the catch block.
* The catch block ~~no~~ cannot be used without try block.

Example :-
```
       class A
    {
    public static void main (String [] args)
  {
  try
  {       deviдeByZero
    int ~~a~~1 = 5/0;    // code that generate exception
    System.out.println ("Rest of code in try block");
  }
  catch (ArithmaticException e)
  {
    System.out.println ("ArithmaticException ="
                          + e.getMessage ());
  }
 }
}
```

O/P → ArithmaticException = / by zero.

Finally block :-

The finally block is ~~alwaely~~ always executed there is an exception or not.

Syntex :-
```
        try {
          // code
       }
       catch {
         //code
      }
      finally {
       //finally block alway executes
     }
```

if an exception Occurs, the finally block is executed after the try catch block.

Example :-
```
         class B
        {
      Public static void main (String []args)
   {
   try {
    int divideByZero =5/0 ;
  }
   catch ( ArithmaticException e) {
    System.out.println ("ArithmaticException ="
                                  + e.getMessage());
  }

 finally {
    System.out.println ("This is the finally block");
  }
}
```

O/P → ArithmaticException = / by zero
This is the finally block.

throw and throws keywords.

3. throw

The throw keyword ~~an exception~~, ~~the~~ used to explicitly throw a single exception

Syntex :-
```
void main () {
    throw new Exception();
}
```

Example :-
```
class A
{
  public static void main(String []args)
{
System.out.println(10/0);
throw new Arithmethic Exception("/ by zero);
}
}
```

O/P → Exception in thread "main". java.lang.
Arithmatic Exception : / by zero

throws keyword :-

It is used to declare the type of exception that might occur within the method.

It is used in the method declaration.

```
~~main()~~
~~{~~
~~try~~
~~{~~
      accessModifier returnType methodName()
      throws ExceptionType1..... {
      // code
}
```

```
{

Import java . io.* ;
class Main {
Public static void findFile ( ) throws IOException
{
 File newFile = new File ("test.txt");
 fileInputstream stream = new FileInputStream
                                   (NewFile);
}
 Public static void main (String []args)
{
 try
 {
   findFile();
 }
 catch (IOException e)
 {
  System.out.println (e) ;
 }
}
```

O/P → Java . io . fileNotFound|Exception : test.txt
(No such file or directory)

1. Arithmatic Exception :-
It is thrown when an exceptional condition has occurred in an arithmetic operation.

2. ArrayIndexOutOfBoundsException :-
It indicate that an Array has been assesed with an illegal index.
The index is negative or greater than or equal to the size of the Array.

3. ClassNotFoundException :-
~~When a File is not accessible or doesn't open.~~
~~4.~~ when we try to access a Class whose definition is not found.

4. FileNotFoundException :-
when a File is not accessible or does not open

5. IOException :-
when an input-Out operation Failed or interru

6. InterruptedException :-
when a thread is waiting, sleeping or doing some processing and it is interrupted.

7. Nullpointer Exception :-
when a method could not convert a string into a numetric format.

8. Runtime Exception :-
This represents any exception which occurs during runtime.

StringIndexOutOfBaundsException :-

It is thrown by string class method to Indicate that an index is either negative or greater than the size of the String.

Throwable

(i) It is provide a string variable that can be set by subclasses to provides a detail message that provide more imformation.

(ii) It's define a one parameter constructor that makes a String as the detail message.

(iii) It's provides getMessage()

Throw	Throws

throw keyword is used to throw an exception object explicitly
void m
throw new AE();
{
=
}
	(i) Throws keyword is used to declare an exception.
	void ~~main~~ m1 (Signature) throws AE
	{
	=
	}

) throw keyword always present inside the method body.	(ii) throw keyword alway used with method signature.
) we can throw only one exception at a time throw new AE ();	(iii) we can Handle Multiple exception using throws keyword. throws AE, NPE, SQLE etc.
) This is followed by an intance. ∴ it deal with an object	(iv) This is followed by an class.

Exception is basically two type
(i) Cheaked Exception.
(ii) Uncheaked Exception

* Uncheaked exception

are RuntimeException and any of its subclass.
Array Index OutOf Bounds
NullPointerException etc.
subclass of the java.Lang.Runtime Exception class
which is a sub class of the Exception class.
It is not cheeked at Compile-time.

* Checked Exception

That are cheaked at compile time.
IOEException, SQL Exception etc are cheeked exception.

* Error

Error is irrecoverable.

Checked	Unchecked
checked Exception are the exception which requires to be handle at compile time.	(i) Unchecked Exception are those exception which are not required to handle at complile time.
All class that inherit from class Exception, but not directly or indirectly from class Run time Exception.	(ii) All exception type that are direct or indirect subclass of Runtime Exception (package. java. lang) Unchecked Exception.
These exception are typically coused by condition which are not under control in program	(iii) These are coused by defect in program.
It is also known as Compile time Exception.	(iv) It is also known as Runtime Exception.
Checked Exception are propagated throws keyword	(v) ~~It~~ They are ~~ata~~ automatically propagated.
Example → IOException SQLException ClassNotFound Exp	Example :- NullPointerException Arithmatic Exception