JAVA (Core)

History of Java :-

→ Java is object oriented Programming language.
→ This is introduced by James Gosling at sun micro in 1991.
→ Java 1st version J.D.K 1.0 introduced June 03 1996.
→ Java is actively maintained by oracle corporation 2010 onwards.

Features of Java :-

1. Simple :-

→ Java is very simple language.
→ we no need to learn before learning Java C, C++.
→ Java is explain 'I am single' to people.

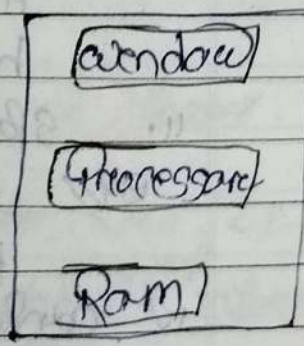
2. Portability :-

→ we can carry the Java byte code to any platform without making any changes.
ex: mobile no. Portability in andra.

3. Architecture Neutral :-

when I execute byte code we get different types of POP UP.

i. Idale window operating system.
ii. Need to improve Processor.
iii. Need to update Ram size.



Where we can use Java ?

1. Embed System
2. Machine learning
3. Artificial intelligence.
4. Banking Domain.
5. insurance Domain.

Shot on OnePlus

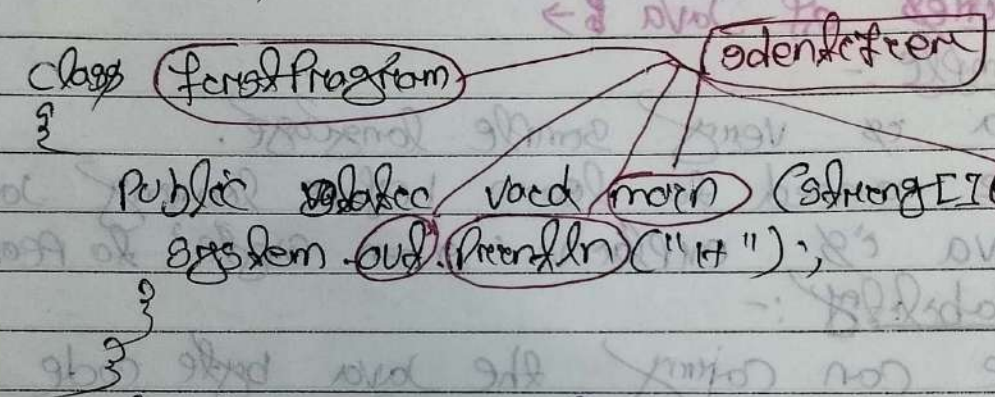
Bookbird
Date
Page

Java Comments :-

→ Comment means developer the understand the purpose.
→ Three Types :-
  i. single line ⇒ //
  ii. multiline ⇒ /* */
  iii. Document comment ⇒ /** **/

Identifiers in Java :-

→ Identifiers is name of Variable, method, class, Package etc & use of purpose identifier



a) Do not start with digit :-
ex// 1hello → ✗
hello1 → ✓

ii. Identifier is case sensitive.
Hello, hello are different

ii. Space not allowed :
ex// First Program ✗
FirstProgram ✓

iv. Contain only alphabet, & character, _ character and digit
ex// Na_Student ✓
First$Program ✓

v. Should not contain keyword.
ex// Public, static, void, class etc.

Keyboard Java :-

→ Keywords are predefined identifiers.

→ Keyword must be used in lower case.

eg/

byte	if	Public	try	class	new
short	else	static	catch	interface	instanceof
int	switch	private	finally	extends	super
long	case	protected	throw	implements	this
float	default	final	throws	package	
double	while	abstract	assert	import	
boolean	do	synchronized			
char	for	native			
	break	strictfp		void	
	continue	transient			
	return	volatile			

1. Project :-

→ Project is nothing but a application or collection of Packages or collection of class.

2. Packages :-

→ Packages is nothing but collection of classes.

3. Class :- eg Car is object, class is Break, driver, color. etc.

→ class is nothing but collection of data, object & information

→ Blueprint of which object created.

4. object :-

→ object is nothing but reference of the class.

→ that execute the class

→ object is real world entity

Shot on OnePlus
Amita

Variable is the name of memory location.
→ It is user defined name which is given by user.

5. Variables :-

→ A variable is a container that stores a value.
→ This value can be changed during the execution of Program. If we want store any data/value in storage place must be created variable - Before variable creation need creation of data type. Data type is under that data value.

int number = 8; → value is store.
Data type Variable name

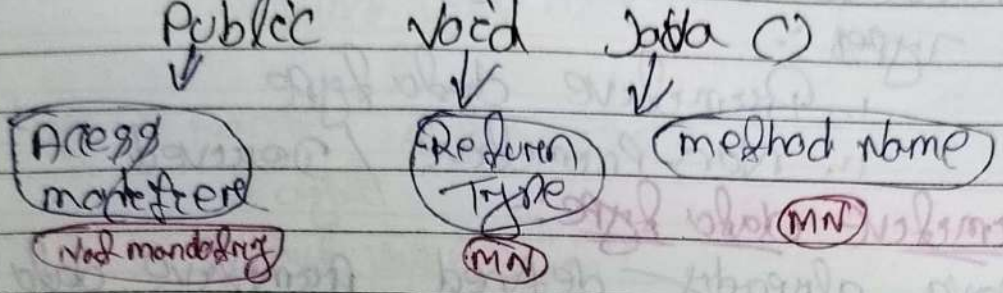
Rules Here declaring a variable name.
i. not started with digit. ex// 1hello ✗, hello1 ✓
ii. identifiers is case sensitive.
ex// Hellow ✓
hellow ✓ are different.
iii. space not allowed
ex// first Program ✗
firstProgram ✓
iv. should not be a keyword
ex// Public, static, void, class etc.
v. contain any alphabets, & character, _ character, digit.
ex// No_student ✓
first Program ✓

Method :-

→ method is nothing but some implementation.
→ If we want write any code in Java we need to write through method over within class only.
→ within class method ~~or~~ we want to execute only we need to execute within main method then consider.

Syntax

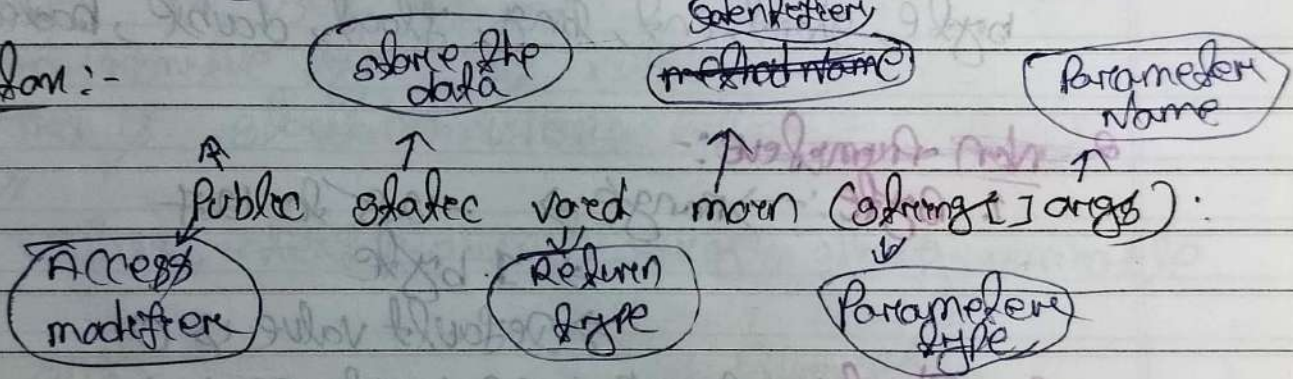
Access_Returntype_methodname.



Main Method :-

→ If we want create the code inside of main method only we need to provide information.

Syntax :-



Return Type (void)

Often using the return type (void) is not return value.

→ We are unable to create method within the main method

Return statement

→ Return statement is return the value.

Shot on OnePlus
Amita😍

Data Types:-

→ If we want to store any data or value in database we need to use data type.

→ 2 Types:-

i. Premitive data type

ii. Non-Premitive / Derived

1. Premitive data type:-

→ Java already defined premitive data type. ~~we no need to~~

→ premitive type has always a value.

→ Premitive starts with lowercase letter.

eg:

byte, short, int, long, float, double, boolean, char.

~~2. Non-Premitive~~:-

1. Byte:- → Range → -128 to 127
   → 1 byte
   → Default value is 0
2. Short:- → Range -32,768 to 32,767
   → 2 byte
   → Default value 0
3. int → Range -2,147,648 to 2,147,647
   → 4 byte
   → Default is 0
4. float →
   Range 1.40012... to 3.40282...
   → 4byte
   → Default value is 0.0f
5. long:-
   → Range -9,223,372 ... to 9,233,372 ...
   → 8 byte
   → Default is 0

6. doubl ⇒ Range = 4.940... to 1.7973...
⇒ 8 byte
⇒ default value 0.0d

7. char :⇒ Range 0 to 65535
⇒ 2 byte
⇒ Defaut value '\u0000'

8. boolean :⇒
value can have are false
Default value is false

Non-Primetive :-
→ They is user defined
→ non-Primetive type has null.
→ They is stored as uppercase letter.
ex//
class, interface, object, array, string, variable.

int :-
→ If u want store any value without decimal are one to use by using int data type.
ex
int a = 456

String :-
→ If u want store any data are need to provide the length either the double code and are need to declare one variable and before the variable are need to declare string data type.
→ They is uppercase

Types of codes :-
1. either the main method → 2'.
2. either the class through main method are.
3. either the class through method are arguments

1. Without the main method :-

Prog

```
Public class Practice {
 Public static void main(String[] args)
{
    int a = 45;
    int b = 15;
    int c = a+b;
System.out.println("The sum of Two no.s :"+c);
```

output

→ The sum of two no.s : 60.

Types of Variables / method :-

They are different types of variable.

1. local variable
2. Static variable
3. instance variable / Public / Global :-

1. Local Variable :-

→ If u want create one method inside of class are called local variable.
→ If we want execute local method first we need create object for class inside of main method.
→ If we want execute all local method we need to call inside of the main method through object name eg/

```
ob.add();
```

2. Static variable: (method

→ once we create the method inside of class through static keyword.

→ If we want execute static method inside of main method we no need create object for the class.

→ If we want execute static method we need to execute through class name.

or//

3. Instance variable / Public / Global :-

→ Instance variable is nothing but once we Provide any value / data within the class out of method.

→ If we want to execute instance variable we are able to call inside of method.

1. Local variable :-

→ A variable declared inside the body of the method or method Parameter called local variable.

Syntax :- class Program {
P.S.V.M void Sonal()
{
int a = 5;
SOP.out..
}
}

Shot on OnePlus
Amita😍

2. instance variable :-

→ A variable which is declared inside of the class but outside of all the method called instance variable. → If want to execute instance variable call inside of main method through object

Syntax :

```
class Saroj {
    int a;
    public static void main(String [] arg) {
        Saroj s1 = new Saroj();
        System.out.println(s1.a);
    }
}
```

3. Static variable :-

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

→ If we execute static variable inside of main method no need to create object.

call through class name

Syntax :-

```
class Naresh
    static void Saroj() {
        int a;
    }
    P.V.m {
        sout(Naresh.a)
```

Shot on OnePlus
Amita😍

2. write a Program on the class through methodwise / user defined meth.

```
class Saroj {
    Public void Add() {
        int a = 45;
        int b = 25;
        int c = a+b;
        Sout (c);
    }
    Public static void main (String [] args) {
        Saroj s1 = new Saroj();
        s1.Add;
    }
}
```

(3) write a Program the class through method with argument

```
class Saroj {
    Public void Add (int a, int b) {
        int c = a+b;
        Sout (c);
    }
    Public static void main (String [] args) {
        Saroj s2 = new Saroj();
        s2.Add (4, 5);
    }
}
```

Shot on OnePlus
Amita

Bookbird

Methods in Java :- → A set of code which performs a particular task.

→ method is a group/block of code which take input from the user processed it & give output.

→ If want write any code in Java we need to create through method either class only.

→ If want execute we need to call from main; we are unable to write method another method.

Types of method :-

Predefined	User defined
→ Print();	→ Add()
→ Sort();	→ Sub()
→ Sqrt();	→ Mul()
→ NextInt();	→ Disp()

Syntax:

```
Return type _ method name
{
    statement
}
```

Why we use method?

→ Decrease line of code.

→ Readability → If any see our Project he will understand our coding.

→ Repeat no. of times.

Constructor :-

→ Constructor is nothing but special type of method whose name same as class name.
→ every Java class has a constructor i.e Default constr.
→ The main purpose of constructor is initialize the object
→ If we don't execute the code we need to create object for class and automatically new keyword will call the constr for the obj
→ constructor never contain any return type including void.

Syntax :-

```
class classname{
    classname()
    {
    soul.
    }
}
```

→ 3 Types 1. Default cons.
2. Parameterized cons.
3. No-args cons.

1. Default constructor :

→ A constructor which does not have any parameter is called default constructor.
→ A default constructor is created only when we do not declare any constructor in our code. Then compiler automatically creates a default cons.

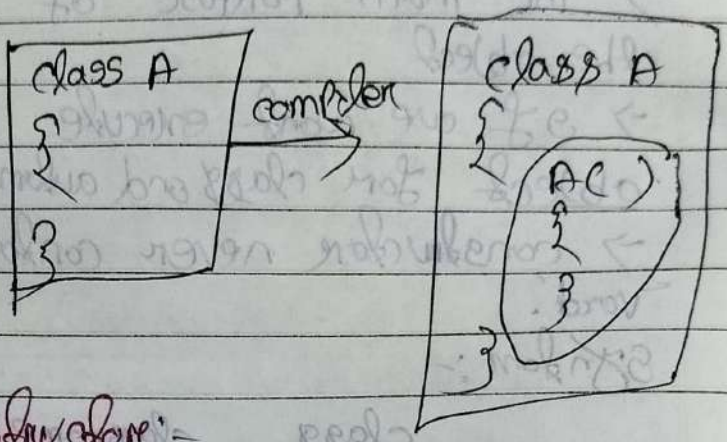
Syntax :-

```
class Soul
{
A
}
```

Shot on OnePlus
Amita😍

Bookbird
Date
Page

class A
{
AC → no any parameter.
{
}
}



2. Parameterized constructor:-
→ A constructor which has a specific no. of parameters / one or more parameters is called Paramet

syntax:-

```
class A{
    A(int x, int y)
    {
      statement
    }
}
```

3. No-args constructor:-
no-args constructor like default constructor. constructor which don't have any parameter is called no-arg cons. when we don't create a constructor that time compiler automatic create a no-args cons

syntax

```
class A{
    AC(){
    }
}
```

Shot on OnePlus
Amita

Array en Java :- Arrays es an object.

→ Array is nothing but storage place or one container of we want store the value/data we need to store similar order/type data.

→ The array index start with zero and end with n-1.

→ If we want store the multiple values inside of the database we need to use arrays.

→ once we provide size we unable to decrease & increase.

→ 3 parts :- Syntax :-

DataType [] VariableName;

→ DataType [] Var.nam = new ~~datatype~~[3];

→ 3 parts

i. → int[] marks;
marks = new int[5];

ii. int[] a = new a[5];

iii. int[] a = ~~new~~ {15, 20, 30, 40, 50};

→ 2 Types :-

1. one dimensional :-

0	12
1	23
2	30
3	45
4	55

```
class Sana {
  P.S.V.m(String[] args){
    int[]a = new[5];
    a[0] = 12;
    a[1] = 23;
    a[2] = 30;
    a[3] = 45;
    a[4] = 55;
    S.O.P(a[3]);
  }
}
```

Shot on OnePlus
Amita😍

2. multi Dimensional arrays:

2D:

	0	1
0	API	selenium
1	java	manual

```
class Java {
P.S.V.m(S) {
String [][] langs = new String [2][2];
langs[0][0] = "API";
langs[0][1] = "selenium";
langs[1][0] = "java";
langs[1][1] = "manual";

So.P(langs[1][0]);
}
}
```

3D:-

	0	1	2
0	15	40	30
1	50	20	80
2	80	70	90

```
class Game {
P.V.m.(S) {
int [][][] a;
a = new a[3][3][3];
a[0][0][0] = 15;
a[0][1] = 40;
a[0][2] = 30;
;
```

Input and output :-

Input (Scanner class) :-

→ Scanner is a Pre-defined class in Java which is available in java.util package. It is used to get user input.

Rule :-

(1) If we use Scanner class, must have to create object of Scanner class.

Syntax :- Scanner objectName = new Scanner(System.in);

(2) Scanner class methods :-

i. nextLine(); → String
ii. nextInt(); → Integer
iii. nextFloat(); → Floating
iv. nextBoolean(); → True or False
v. nextDouble(); → double.

(3) Import Scanner class Package at the top line of Program.

Syntax. import java.util.Scanner;

Syntax

Scanner objectName = new Scanner(System.in);
variableName = objectName.nextFunctionName();

Bookbird

Date
Page

Out Put : (System.out.Print())
→ It is an output statement in Java through which we can Print the variables, expression and many more content.

(Q) Addition Two number :-

```
import java.util.Scanner;
class Java {

Public static void main (String [] args){
 S.o.P(" Addition Two number ");
Scanner ob = new Scanner(System.in);
S.o.P(" Enter first no. ");
 int a = ob.nextInt();
S.o.P (" Enter 2nd no. ");
 int b = ob.nextInt();
int sum = a+b;
S.o.P(" sum of two no. is:");
S.o.P(sum);
}
}
```

Variable Shadowing :-
when class level variable (instance var) and local variable both are same :- Variable shadowing.

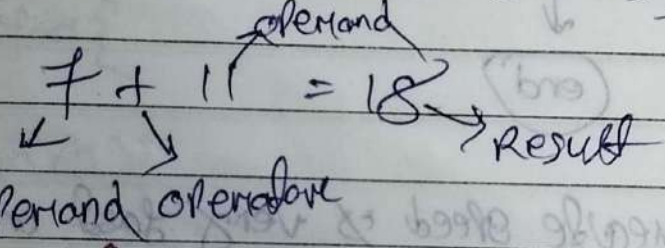
```
class A{
    int a = 20
void m() {
    int a = 30
```

Operators in Java:

→ Operator is a symbol that is used to perform operations on variables and values.

Types of operation:

(1) Arithmetic → (+, −, *, /, %, ++, −−)
(2) Relational → (<, >, >=, <=, !=, ==)
(3) logical → (&&, ||, !)
(4) Increment [Pre incr / Post (++var, var++)
Decrement] Pre dec / Post dec (−− var, var −−)
(5) Assignment → (=, +=, −=, *=)
(6) Bitwise → (AND, OR, XOR,) → (0, 1)



New Keyword :-

→ new is keyword in Java.
→ It is used to create new objects and it is used to allocate dynamic memory at runtime.

Usages:

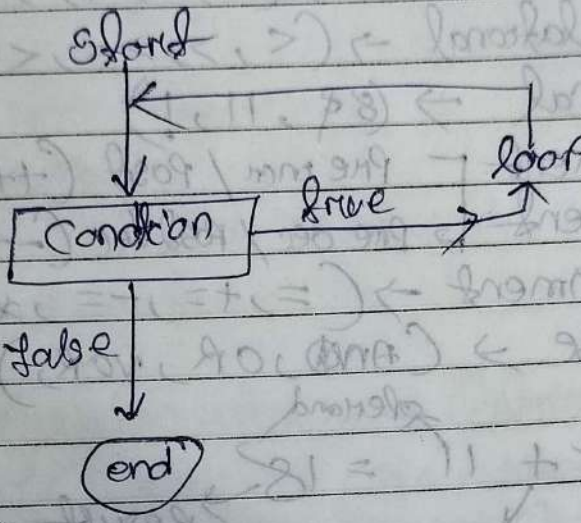
1. call instance variable
2. non-static method
3. call constructor
4. Array

Shot on OnePlus
Amita

Loop in Java:

→ Sometime we want to execute a particular part of the program repeat several time is called loop when statement is true.

Flow chart:



Advantages:

→ To execute speed is very fast in our program.
→ Not execute statement repeatedly.
→ If we use loop length of code decrease.

Types of loop:

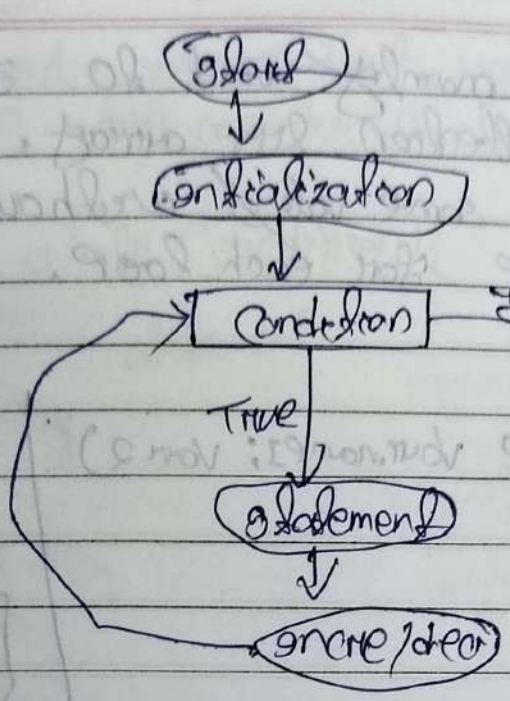
1) For loop
2. For each loop
3. while loop
4. Do-while loop

1) For loop:-

→ for loop is usually used to execute a piece of code for specific no. of times.

Syntax:

```
For (initialization; condition; inc/dec)
{
    statement
```



class for{

P.S.V.m(S){

for(int a=1; a<=10; a++){ | for(int a=10; a>=1; a-

S.O.P(a);

}

}

}

Variable hiding:

when we create variable in sub class with same variable as superclass, i.e variable hiding;

output

1
2
3
4
5
6
7
8
9
10

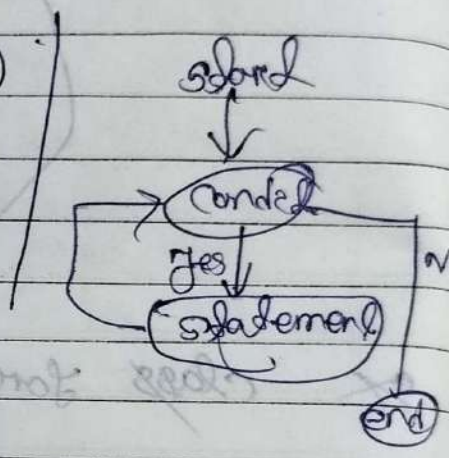
Shot on OnePlus
Amita

For each loop:-

→ for each loop mainly used to fetch the values from a collection like array.
→ If want print some values without index then we need use for each loop.

Syntax:

```
for(datatype var.name1 : var 2)
{

}
```



eg:

```
var2[] = {10, 20, 30, 40, 50};
```

eg:

```
class for_each{

P.S.V.m(S){
int a[] = {10, 20, 30, 40, 50};
for(int b : a){
  S.o.P(b);
}
}
```

or String a = {"Sona", "Sarika", "Para"};

```
S.o.P(Arrays.toString(a));
```

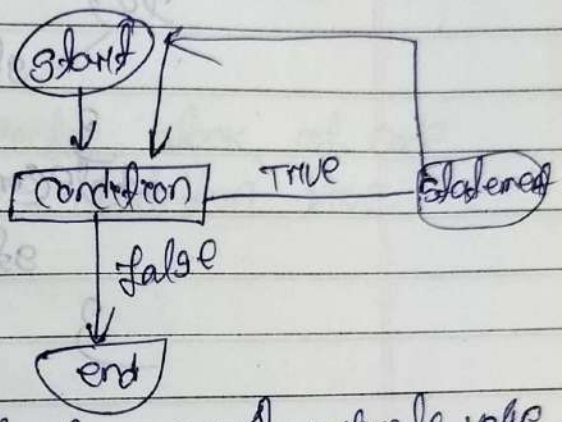
Shot on OnePlus
Amita😍

while loop : Pre test

→ It is used when we don't know the no. of iteration in advance. when condition true enter and false exit loop.

Syntax:

```
while (condition){
    statement
}
```



→ with out name/declare in while loop we get infinite value.

eg

```
class Sam{
P.S V.m(S){
int a=1;
while (a<=5){
    S.o.P(a);
    a++;
}
}
}
```

Compiler time error:-
when we write wrong syntax i.e compile time error.

Run time error:-
The runtime error that occure during the execution the program

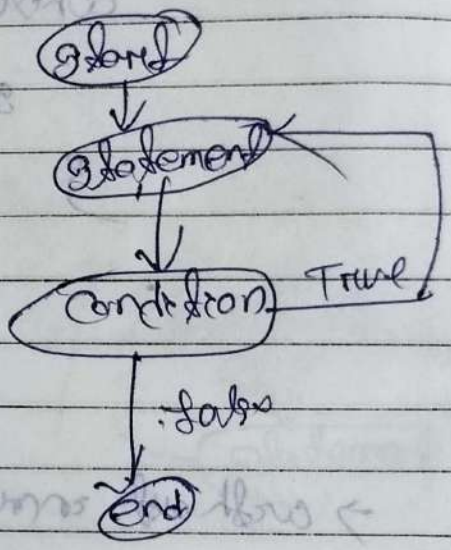
Shot on OnePlus
Amita😍

4. Do while loop: Post-test

- Do while is used when we want to execute loop body atleast one even condition is false.

Syntax:-

```
Do {
    statement
}
while(condition) {
    statement
}
```



eg

```
class Sona {
    P.g.vm(S) {
    int n=1;
    do {
        S.o.P(n);
        n++;
    }
    while (n<=5);
    }
}
```

Shot on OnePlus
Amita😍

Conditions / Decision making in Java:

There are diff types :-

1. if statement
2. else
3. else if condition
4. switch

(1) if :-

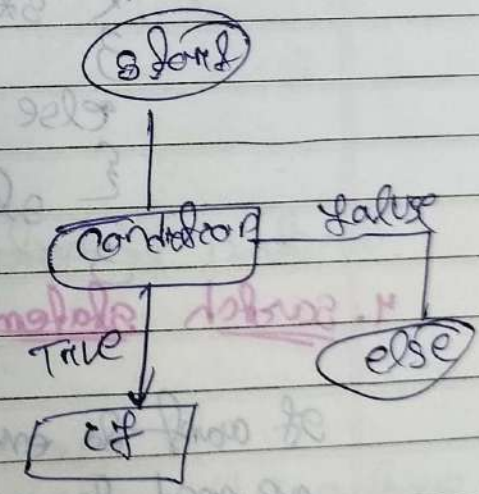
If we want to execute a specific block of code then we use if. if a specific condition true.

Syntax :

```
if (condition)
{
  statement
}
```

if specific condition false.

```
if (condition) false
{
  statement
}
else
{
  statement
}
```



Shot on OnePlus
Amita

Bookbird

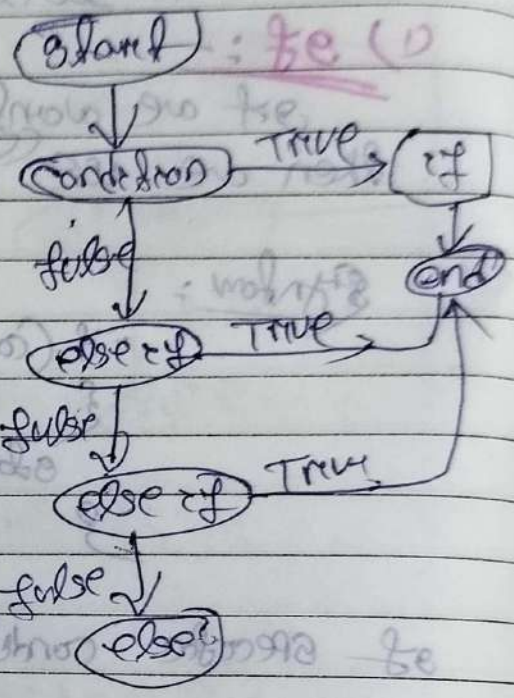
Date

3) else if :

If want to execute one more condition we need to else if. multiple condition fail then else execute.

Syntax:

```
if (condition) x false
{
  statement
}
else if (condition) x false
{
  statement
}
else if (condition) x
{ statement
}
else
{ statement;
}
```



4. Switch statement:

If want to execute many alternate block of code. we need to use switch condition.

→ If we select only one case out many multiple then use switch.

Syntax:

```
switch (condition Exp)
{
    case 1: statement;
            break;
    case 2:
    default:
```

Shot on OnePlus
Amita😍

Break statement:

when we use 'break' keyword break out the loop when a match is found the execution will break.

ex:
out of 10 we enable 5 without making changing diverse break.

ex: stop the loop when a equal to 5

```
for (int a=0; a<=10, a++)
{
    S.o.P(a);
    if (i==5)
    {
        break;
    }
}
```

Continue:

~~No break~~

~~when we use continue condition is continue.~~

Continue statement:

when we use continue keyword

→ continue statement used for skip the current iteration of a loop after the continue the loop.

→ It is used only loop

ex:
10 testcases is there. we no need to enable 5 but all testcases enable

```
for (a=1; a<=10; a++)
{
    if (a==5)
        continue
}
```

Shot on OnePlus
Amita😍

Bookbird

Date

OOPs in Java: (object-oriented Programming system)

→ The main purpose of oop is to deal with real world entity using Programming language, inheritance, Polymorphism.

→ The main aim of oop is to bind together data & function

There different concepts in oops.

1. class
2. object
3. inheritance
4. Polymorphism
5. Abstraction
6. Encapsulation

Diff:- Pillars, Terms, Type.

Object means a real-world entity such as a pen, chair, table, computer etc.
oop is a methodology to design a Program using classes and objects.

1. Class :-

→ class is nothing collection of data & information or object
→ class is blue Print when object create

class

Predefined	user defined
→ Scanner	→ Dog
→ console	→ Sonu
→ System	→ A
→ String	→ etc

Shot on OnePlus
Amita

(3) Inheritance :-

When we create a new class from existing class in such a way that the new class access the all features & properties of existing class called inheritance.

→ 'extends' keyword is used to perform inheritance
→ It provide code from existing class.
→ If we want execute parent class only able to execute inside child class.
→ method overriding only possible through inheritance.

→ Syntax:

```
class A
{
    statement 1
}
class B extends A
{
    statement 1
}
```

Types :-

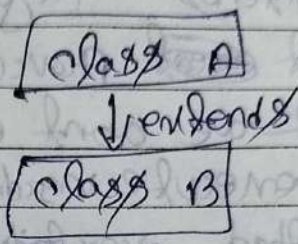
1. Single inheritance
2. multilevel inheritance
3. Hierarchical inheritance
4. multiple inheritance.
5. Hybrid.

Shot on OnePlus
Amita

Parent class extends child class
i.e call ing.

1) Single inheritance :-

i) Single inheritance is nothing but two classes in there are parent class & child class is called single...
→ If and to execute parent class only we are able to execute inside of child class through extends keyword.



syntax :

```
class Parent{
        statement
}
class child extends Parent{
        statement
}
```

eg

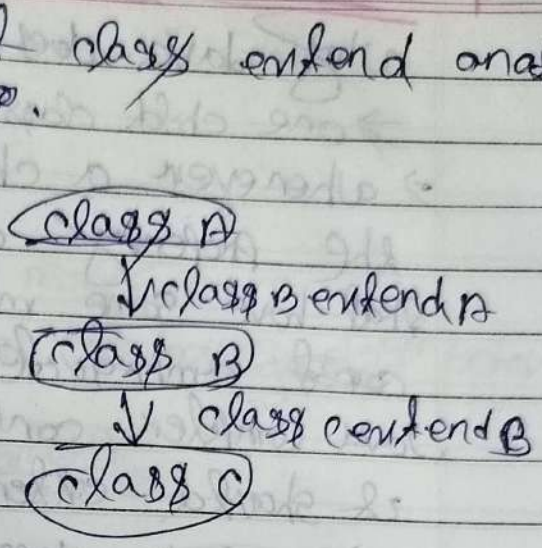
```
class Saral{

  void add(){
    int a=45+5;
    System.out.println(a);
  }
  void sub(){
    int b=45-5;
    System.out.println(b);
  }
}
class Sarva extends Saral{
    P.V.M(9)
    Sarva s2=new Sarva();
    s2.sub
```

(2) multilevel inheritance:

→ A class extends a class that class extend another class i.e multiple inheritance.

eg:

There 3 classes i.e Here classA, classB & classC. are ~~extend~~ classB extend classA, classC extend B.



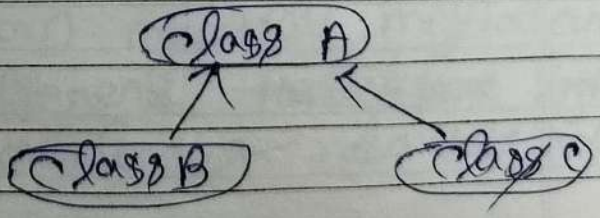
Syntax:

```
class A{
    ==
}
class B extends class A{
    ==statement
}
class C extends class B{
    ==
}
```

(3) Hierarchical inheritance:

→ An inheritance which contain only one Parent class and multiple child class and all child class directly extends Parent class called hierarchical inheritance.

Syntax:




```
class A
{
}
class B extends class A
{
    ==
}
class C extends class A
{
    ==
}
```

4. multiple inheritance :-

why java doesn't support multiple inheritance.

→ one child class and two or more Parent class.

→ whenever a child class wants to inherit the property of two or more Parent class. that have same method. the java compiler get ambiguity (confusion) Problem that means Java compiler can't decide which class method it should extend.

ie Java doesn't support multiple inheritance through class.

But interface supported

class A class B

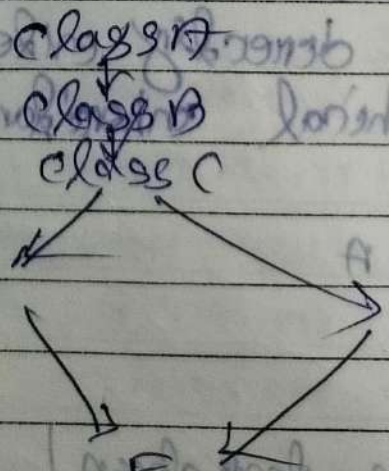
class C extends A,B.

5. Hybrid:

Hybrid is nothing but more than class is there.

There all inheritance is there single, multilevel, hierarchical, multiple hybrid.

main diff advantage multiple inheritance not support that is hybrid not supported.



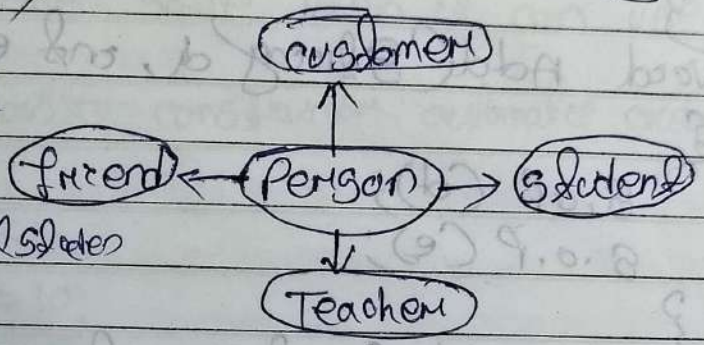
Shot on OnePlus
Amita

2. Polymorphism:

→ Poly ⇒ many
morphism = form

⇒ Polymorphism is nothing but many forms.
→ Polymorphism is same object having diff behaviour.
eg: single action

eg: Person going to married
he is customer
and going to school student



⇒ There are two Types:-

(1) Static / compile / early ⇒ eg: overloading
(2) Dynamic / run time / after ⇒ eg: overriding

(1) compile / static Polymorphism:

⇒ A Polymorphism which is exists at the time of compilation is called compile time are static/early

eg: Method overloading:

Method overloading:
whenever a class contain multiple method with same name and but different types of Parameters called overloading. also diff return type

Syntax:

```
return type method name (Para1);
return type method name (Para2);
```

Shot on OnePlus
Amita

eg method overloading.

```
class Sarra {

    Public Void Add(int a, int b)
    {
        int c = a+b;
        S.o.P(c);
    }
    void Add(String d, int e)
    {
        S.oP (d);
        S.o.P (e);
    }
    Void Add (int x, String y)
    {
        S.o.P (x);
        S.o.P(y);
    }

    String Add (String s, String p)
    {
        return s;
        return p;
    }
Public S.V.main(String[] args)
{
    Sarra SS = new Sarra();
    SS.Add(5, 10);
    SS.Add ("Sarra", 10);
    SS.Add (10, "Sarra");
    S.o.P (SS.Add("Sarra", "Sarra"));
}
}
```

Super Keyword :-

Super keyword nothing but refers to the object of super class.

→ It is used when we want call super class variable, method & constructor through sub class.

→ whenever super class & sub class variable and method name are same then it can use only.

Syntax (Default constructor automatic create super keyword)

```
class A {
    int a = 10;
}
class B extends A {
    int a = 20;
    void show() {
        S.o.P(a);
        S.o.P(super.a);
    }
P.S.V.m(String[] arg)
{
    B ob = new B();
    ob.show();
}
}
```

super.show()

Shot on OnePlus

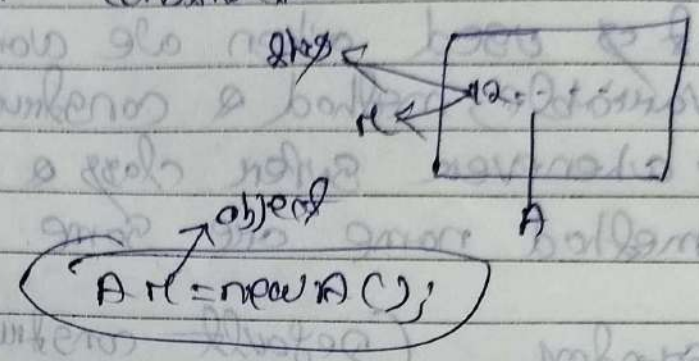
Bookbird
Date
Page

This keyword:-

→ this keyword refers to the current class instance variable are method or constructor.

ex/

```
class A {

}
```



→ when instance variable and local variable both are same then Java get confused about instance variable. to avoid this problem we should use this keyword.

ex//

```
class A {
    int a; //→ instance
    A(int a) //- local
    {
        a=a; → confused
        S.o.pln(a);
    }
}
```

Q// How many ways to create an object in Java:

⇒ There are five different ways to:

1. create object in Java using new keyword
2. // using newInstance() method
3. // newInstance() method of constructor class.
4. using clone() method
5. using Serialization and Deserialization.

2) Run time / dynamic / after Polymorphism

⇒ a Polymorphism which is exist at the time of execution of Program is called runtime Polymorphism.

Method overriding:

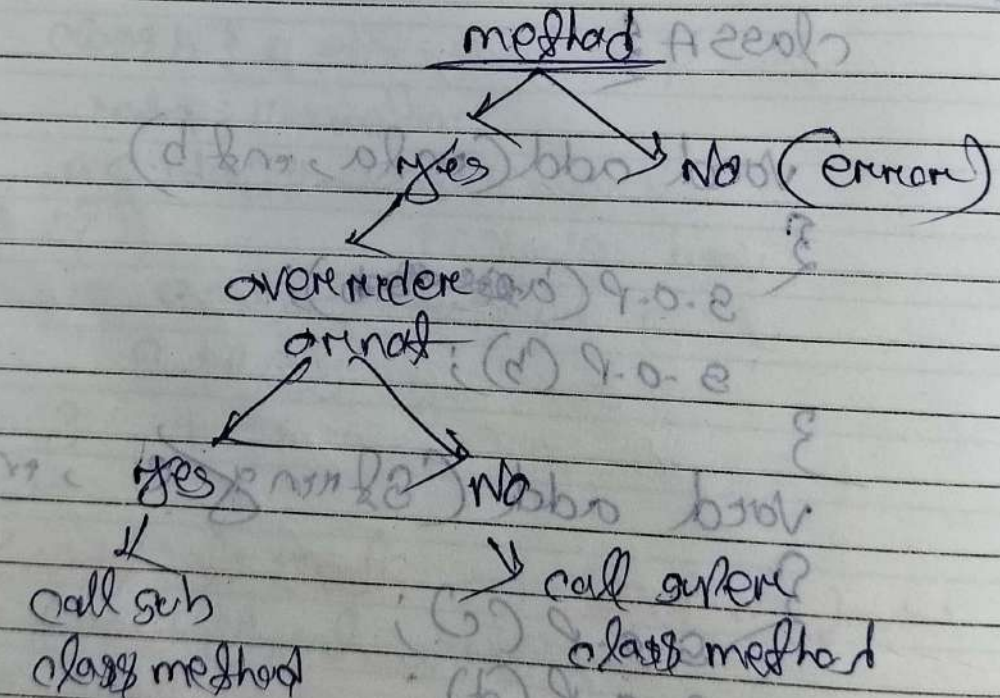
→ when we are write method in super and sub class. Both classes having same method & same Parameter is call method overriding

→ If we want execute method overriding we need to execute inside of child class.

Syntax:

```
class A {

void add(int a, int b)
{
  S.o.P(a);
  S.o.P(b);
}
void add(String c, int d)
{
  S.o.P(c);
  S.o.P(d);
}
}

class B extends A {
  void add(int x, int y){
    S.o.P(x);
    S.o.P(y);
  }
```

```
void add (String s, int n){
    s.o.P(s);
    s.o.P(n);
}
p.s.v.m(s){
    B ob) = new B();
    ob.add(10,15);
    ob).add("sara",10);
}
}
```



3. Encapsulation :-

→ encapsulation is process of binding code and data together into a single unit called encapsulation

→ declare the class variable (data) as Private.

→ the class method as Public.

→ we can use setter & getter methods to set and get data.

```
class Sorry {
    Private String name;
    Private String Adress;
    Private int age;
    Private int Rollno;
Public void setname(String name);
{
this.name = name;
}
Public String getname()
{
    return name;
}

Public void setadress(String adress)
{
    this.adress = adress;
}

Public String getadress()
{
    return adress;
}
```

// A driver know How to store the data by passing the store button and external process instead of storing hidden from customers.

Shot on OnePlus
Amita

```
Public void setage(int age)
{
    this.age = age;
}
Public int getage()
{
    return age;
}
Public void setRollno.(int Rollno.)
{
    this.Rollno = Rollno;
}
public int getrollno()
{
  return rollno;
}
P.S. V. main(String[] args)
{
   Sara) s = new Sara)();
   s.setname("Sara");
   s.o.p(s.getname);
   s.setadress("Bankamohan");
   s.o.p(s.getadress);
   s.setage(23);
   s.o.p(s.getage);
   s.setRoll(21);
   s.oP(s.getRoll);
}
}
```

5- Abstraction:

→ Abstraction is a process of hiding the implementation details from the user, only showing the functionality.

→ It is used for classes and methods.

~~→ we can't create~~

Abstract class:-

→ A class which is declared as abstract keyword is known as abstract class.

→ we can't create object for abstract class.

→ If we want access abstract class, you have to inherite it from sub classes.

→ It can have abstract and non-abstract method.

ex// 1. Sending sms ~~to~~ text to ~~friend~~ many where you don't know external process about the message delivery.

2. making coffe with coffee machine, ~~we don't know~~ we need to provide water & coffee beans and switch on then we don't forget.

```
abstract class Sarva{
    abstract void Sarv();
    abstract void Saro();
}
class Sarva extends Sarva{
    public void Sarv(){
        S.o.P("Saro");
    }
    public void Saro(){
        S.o.P("Sarvx");
    }
P.V.S(S){
    Sarva s=new Sarva();
    s.Sarv(); s.Saro();
```

Bookbird

Date
Page

abstract method :-

→ A method which declare abstract modifier at the time of declaration is called abstract method.

→ It can only be use abstract class.
→ It doesn't contain any body { }.
→ abstract method must be overriden in subclass.
→ when the action is common but implementation are different then we use abstract method.

Length Vs Length() :-

Length :- length is a final variable this is applicable for arrays.
→ length represent get the size of the array.

eg//

```
class Gana{
Public static void main(String[]args)
{
  int[]a = new int[4];
  int b = a.length;
  S.o.P(b);
   or
  S.o.P(a.length);
}
}
```

length() :- method

-> length() method is only applicable string.
-> To get the size of a string are How many characters a string contains.

eg

```
class Sorna{
Public static void main(String[] args)
{
String a = "Sorna";
S.o.P(a.length());
   or
int b = a.length();
S.o.P(b);
}}
```

Q How to print position with data by using arrays:

```
class Sorna{
  Public static void main(String[] arg){
String[] name = {"Sorna", "Sorna", "Pakru", "munda"};
for(int a=0; a<name.length; a++)
  {
    S.o.P(a + " " + name[a]);
  }
}
```

Diff" bet" Array list & Hash set :-

List:

-> Array list maintain the order collection
-> Array list allows duplicate keys or duplicate value:

eg Sorna);
Sorna);
Sorna);

Set:

-> set is an ordered collection. does maintain any order.
-> Set doesn't allow duplicate key.
-> Sorna);
-> Pakru;

Shot on OnePlus

Java Hash map:-

→ The Hash map is comes under collection concepts.
→ It stores elements in key / value pairs
Here key is unique.
→ Hash map does not allow duplicate key but values can be duplicate.
→ each key value pair is called entry
Here map is consider as collection entry objects
→ If we try to insert the duplicate key will replace value or in new value.
→ Hash map allow null key & null values.
→ Hash map is unordered map here no order.

(1)

Hash map a = new Hashmap();
default initial capacity 16 and load factor 0.75

(2) Hash map a = new Hash map(int initialcapacity)
specified initial capacity default ration 0.45:

(3)

Hash map a = new Hash map(int initial capacity, float fill ratio)

(4) Hash map m = new Hash map(map m)

random:

HashMap<String, Integer> a = new HashMap<>();

Shot on OnePlus
Amita

object → a → variable name / K → key, V → value

Method	Description
adding	1. a.Put(K,V); → If we want to insert/Add (specified pairs) in map
	2. a.PutAll(map); → If we want to insert group of pairs in map.
	3. a.PutIfAbsent(K,V) → If insert the specified key/value pair to the map if key is already not present in map.
returning	4. a.get(K); → It return the value of the specified key.
	5. a.getDefault(K,V); → If key is specified then return value. If key not specified then return default value
	6. a.entrySet(); → If return a set view of all the entries from the hashmap.
	7. a.keySet(); → If return set view all the keys in Hashmap.
	8. a.values(); → If return set view all the value in Hashmap
change/replace	9. a.replace(K,V(old),V(new)); :- a.replace(K,V) → If replace value for specified key. a.(K,V(old),V(new)); → If replace old value with new value for a key is found.
	10. a.replaceAll((K,V)->function()); eg function → value.toUpperCase(); → key*key; → If replace all map of the Hash map

Bookbird Date Page
Shot on OnePlus Amita

11. a.remove(K); → Remove the key from Hash map
or
a.remove(K,V) → True → Remove both key & value from Hash.
a.remove(K,V) → false → Did not remove.

12. a.clear();
→ It removes all the key/value pairs from hashmap.

13. a.clone(); → makes a duplicate or shallow copy and returns it.

14. a.isEmpty(); → check the map is Empty → it True
No empty → it false.
or
HashMap<String, Integer> a = new HashMap<>();
boolen result = a.isEmpty(); // True
S.o.P(result); output - True
a.put("Sona", 11);
a.put("Sonu", 12);
result = a.isEmpty(); // false
S.o.P(result); output // false

15. a.size();
→ It return the number of key/value pairs present in ma
or
HashMap<String, Integer> a = new HashMap<>();
a.put("Sona", 11);
a.put("Sonu", 12);
int size = a.size();
S.o.P(size);

output
→ 2

Shot on OnePlus
Anita

16 a.containsKey(K); : checks the key of map for the key is present or not

eg:

```
HashMap<String, Integer> a = new HashMap<>();

    a.Put("Sonu", 1);
    a.Put("Sonu", 13);
if(a.containsKey("Sonu")); // TRUE
  {
     S.o.P("Available");
  }
or
if(a.containsKey("Pakuno"); // false
  {
     S.o.P("Not Available");
  }
```

17. a.containsValue(V); :- check the map for the value is present or not the Hash map.

eg:

```
HashMap<String, Integer> a = new HashMap<>();
    a.Put("Sonu", 11);
    a.Put("Sonu", 12);

if(a.containsValue("11"));
  {
     S.o.P("available");
  }
or
if(a.containsValue(14));
  {
     S.o.P(Not available);
  }
```

Shot on OnePlus
Amita

18) a.forEach(key, value) -> { } is used to perform action to each map.

ans)

```
HashMap<String, Integer> Price = new HashMap<>();

Price.put("Alu", 20);
Price.put("onion", 30);
Price.put("Garlic", 40);

Price.forEach((key, value) -> {

    value = value - value * 10/100;
    S.o.P(key + "=" + value + "=");});
```

output :- Alu = 18 onion = 27 Garlic = 36

19 a.merge();

20. a.compute();

Hash Set :-

→ The Hash Set is comes under collection concepts.
→ Hash set allows to store multiple values in a collection using a hash table.
→ Hash set stores the elements by using mechanism called Hashing
→ Hash set contains unique elements only.
→ Hash set allows null value.
→ most useful in the search operations
→ Default capacity 16 & load factor 0.75 / storage reordered

Syntax:

```
HashSet<DataType> name = new HashSet(capacity, loadfactor);
```

(1) add(E e); //boolean
→ It is used to add element in the set if it is not already present.

or//

```
HashSet<String> a = new HashSet<String>();
a.add("Saroj");
a.add("Parul");
// add element
a.add("Sanya");
S.o.p(a);
```

output:- Saroj, Parul, Sanya

(2) a.clear();
→ It remove the all elements.

3. a.clone();
→ It return clone / bhallaa copy.

4. a.contains(object); // boolean
→ used to return true if an element present in set.
otherwise false.

Shot on OnePlus
Amita

eg
HashSet<String> a = new HashSet<String>();

a.add("Alu");
a.add("onion");
a.add("Jalebi");
S.o.P(a.contains("Alu")); output → True
or
S.o.P(a.contains("Amba")); // false

5. a.isEmpty();
→ It is used to return true if they set no elements.
Boolean

eg//
HashSet<String> a = new HashSet<String>();
S.o.P(a.isEmpty); output - True

a.add("Saroj");
a.add("Pallavi");
S.o.P(a.isEmpty); output → false

6. a.iterator();
→ It is return an iterator over the elements in set.

eg//
a.add("Saroj");
a.add("Pallavi");
Iterator itr = set.iterator();
for(String s:set)
{
S.o.p(s)
}

(7) a. remove(object);
→ remove the specified element in set if present.

(8) size();
→ is used to return the number of elements in set.

(9) spliterator();

Bubble Sorting :-

If we want do bubble sorting we need to use two loops → outer loop & inner loop.

```
int[] a = { 4,5,6,1,2,7,3};
int b= a.length;

for (int c=0; c<b-1; c++)
{
    for(int d=0; d<b-1; d++)
    {
        if(a[d] > a[d+1])
        {
            int temp = a[d];
            a[d] = a[d+1];
            a[d+1] = temp;
        }
    }
}
S.o.p (Arrays.toString(a));
```

Shot on OnePlus
Amita

Collections:

What is collections framework?
→ java.util.collection
→ java.util.map

→ Java collections are the set of pre-defined classes and interfaces that helps programmers to perform different kinds of data structure operations like storing, searching, traversing, storing and processing data efficiently.

collection: interface
→ A collection represents a single unit of objects i.e group or which can store multiple data.

Different type of collection class.
→ Hashmap
→ HashSet
→ TreeSet
→ ArrayList
→ LinkedList
→ TreeMap
→ PriorityQueue

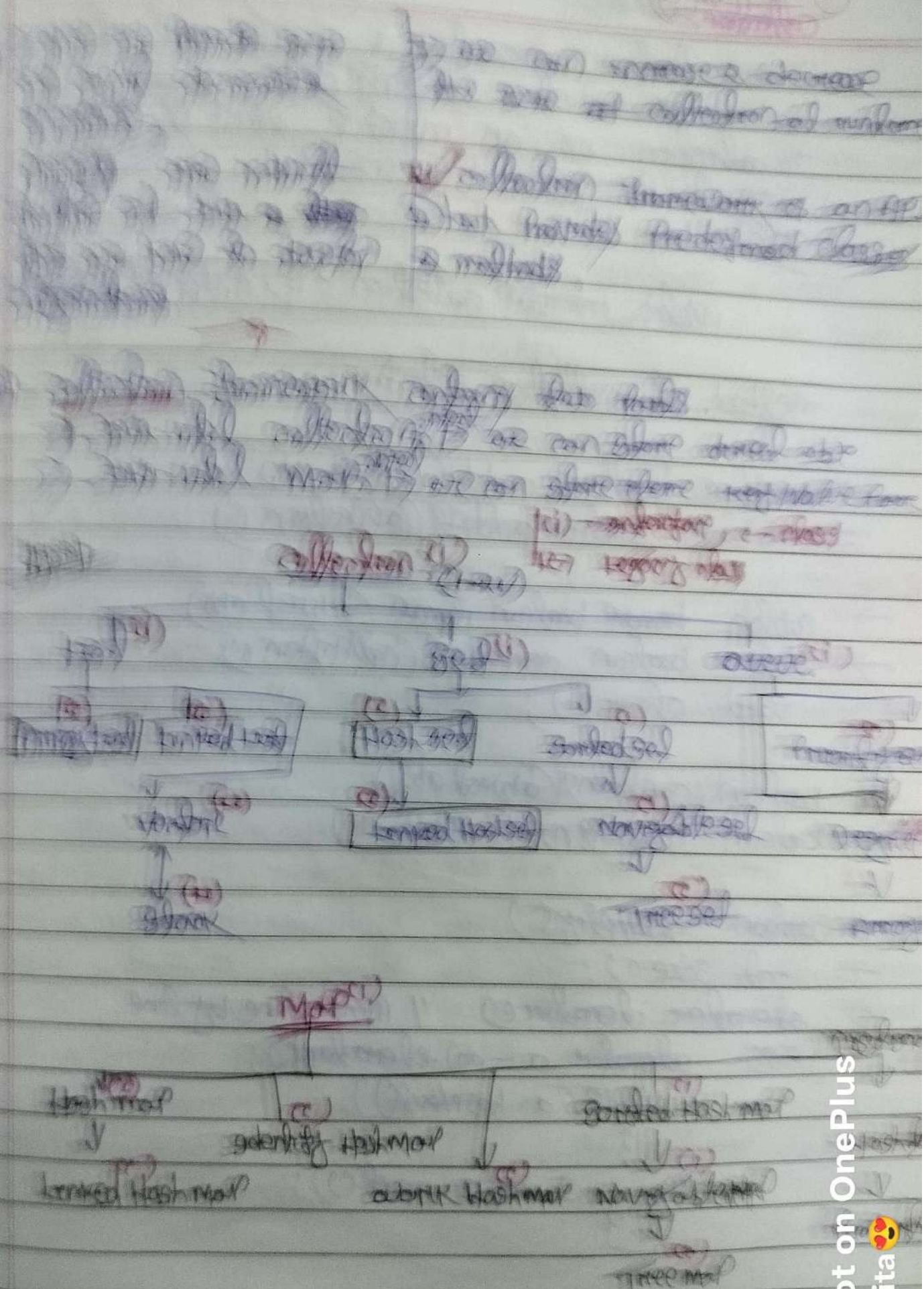
Arrays

→ Arrays can store Primitive & non-Primitive type of data.
e.g
int [] a = {10, 20, 30, 40};

2 → Arrays can store
→ Homogeneous same data type data.
eg: {10, e, 20, 30} ✗

collection Frameworks:

→ collection framework can store only non-Primitive type of data.
e.g
ArrayList a = new ArrayList();
a.add(obj)
a.add(10); → autoboxing
→ we only use wrapper class
→ we can store Heterogeneous different type of data.



Shot on OnePlus
Amita

Collection interface :-

→ A collection represents a single unit of object which can store multiple data.
→ collection is an interface which present java.util Package (1.2 version)

Syntax :-

public interface collection<E) extends Iterable<E>

method of collection interface :-

1) public boolean add (object ob))
2) public boolean addAll (collection c)

→ public boolean remove (object ob))
→ public boolean removeAll (collection c)
→ void clear ();

→ boolean contains (object ob)
→ boolean containsAll (collection c)

→ boolean isEmpty ()
→ int size ();
→ iterator iterator () // most used
e.g iterator a = ob).iterator();
while (a.hasNext())
{
S.o.p(a.next());
}

Shot on OnePlus

index means arr[0] = 2 11 B, 32, 3e3 [0] [1] 0 1 2 3 4 5 6 7

List	Set
1. → List is an index based data structure	1. Set is not an index based data structure. It store the data according to the Hash code values.
2. List can store duplicate element.	2. Set does not allow to store duplicate element.
3. List can any no. of null values.	3. set can store only one values.
4. List maintain the order collection.	4. set does not maintain order collection. i.e. unordered.
5. we can iterate (get) the list element by using a ListIterator (List a=new ArrayList())	5. we can iterate the set element by iterator. set a = new HashSet()

— x o —

Interface :-

→ Interface is just like a class, which contains only abstract method.

To achieve interface java Provides a keyword called implements. If we want achieve interface we need to implement to another class the implement keyword.

Note :-

(1) → Interface methods are by default Public & abstract

2. → Interface variables are by default public static final

3. Interface method must be overriden inside the implementing classes.

4. Interfaces nothing but deals between client and developers.

Shot on OnePlus

```
interface client
{
    void inPut();      // public abstract
    void outPut();     //    //
}
class Sara implements client
{
    String name;
    double salary;
    public void inPut()
    {
        Scanner s = new Scanner(System.in);
        S.o.p("Enter name");
        name = s.nextLine();
        S.o.p("Enter salary");
        salary = s.nextDouble();
    }
    public void outPut()
    {
        S.o.p(name+" "+salary);
    }
}
main(String[] arg)
{
    Sara g = new Sara();
    g.inPut();
    g.outPut();
}
```

Q// multiple inheritance using interface?

→ We can execute multiple inheritance through interfaces because interface contains only abstract method, which implementation is provided by the sub class.

eg//

```
class C implements A,B ;

=

interface Sorto{

  abstract void add();
  abstract void sub();
}
interface Sorta{

  abstract void mul();
}
class Patan implements Sorto, Sorta
{
  public void add(){
      S.o.P("Sorty");
  }
public void sub(){
    S.o.P("Sorting");
  }
  public void mul(){
    S.o.P("Mul");
  }
 main(){
    Patan s=new Patan();
    s.add();
    s.sub();
    s.mul();
}
```

Shot on OnePlus
Amita

Interface :

Interface	abstract class
→ interface contains only abstract methods.	→ abstract class contains both abstract and non-abstract method or concrete method
→ it supports multiple inheritance	→ it doesn't support multiple inheritance.
→ By default interface methods are public abstract	→ There are no any restriction on abstract class method modifier.
→ By default interface variables are public static final.	→ abstract class have final, non-final, static & non-static variables.
→ interface keyword is used to declare interface.	→ abstract keyword is used to declare abstract class.
→ interface can be implements using "implements" keyword.	→ abstract class can be extends using "extend" keyword

Iterator (cursor)	List Iterator
→ we can get iterator cursor by iterator() method.	→ we can get ListIterator cursors by listIterator() method.
Iterator itr = a.iterator();	ListIterator lt = a.listIterator();
→ it can used with any collection object	→ it can used only List implemented classes - ArrayList, LinkedList, Vector
→ iterator method are has hasNext(), next(), remove()	→ ListIterator methods are hasNext(), next(), hasPrevious(), remove(), set()
→ By using iterator cursor we can retrieve the elements only in forward direction.	→ By using ListIterator cursor we can retrieve the elements in forward & backward direction

ArrayList :-

→ ArrayList is an implemented class of List interface which is present in java.util package

Syntax: class ArrayList implements List {

→ ArrayList is created on the basis of growable or resizable array.

→ ArrayList are index based data structure.

→ ArrayList are store different data types or Heterogenous data.

1 → Array List can store duplicate value.

2 → ArrayList can be store any number of null values.

3 → ArrayList follows the insertion order.

4 → Array List does not follows the sorting order.

5 → Array List are non-synchronised.

eg//

```
ArrayList a = new ArrayList();

a.add(10); // (0,10) index
a.add(20);
a.add(30);
a.add(40);
S.o.p.(a);
a.add("Deepak");

a.set(2,30);
S.o.p(a);
```

Shot on OnePlus
Amita

Linked list:

→ Linked list is an implemented class of list interface class which is present in java.util package.

syntax:-

class LinkedList implements List, Deque.

{

=

=

}

→ insertion & deletion operation are efficient are fast.

→ Duplicate element

→ store multiple null value.

same as array list

0, 1, 2, 3, 4, 5

e.g.

```
LinkedList a = new LinkedList();
a.add("dekar");
a.add("gora");
a.add(15);
a.add(
s.o.p(a);
a.addFirst("aaa");
s.o.p(a);
```

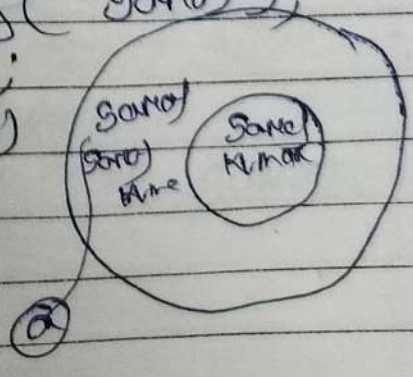
String :-

→ String is a non-primitive data type.
→ String is sequence of characters
ex// char[] a = {'S','a','r','o','j'};
String b = new String(a);
→ String is immutable object.
→ String is a class.
→ we can create String there 3 classes
→ String
→ StringBuffer
→ StringBuilder

→ Java Strings are object that allows us to store sequence of character which may contain alpha numeric values enclosed in double quotes.
→ Strings are immutable object
→ 2 ways of create String object.
→ string literal // String a = "Saroj";
→ "new" keyword // String a = new String("Saroj");

→ what is String immutable object :-
→ because once String object is created its data can't be changed but a new String object is created.

ex// String a = new String("Saroj");
a.concat(" Kumar");
S.o.P(a); // Saroj
a = a.concat(" Kumar");
S.o.P(a); // Saroj Kumar



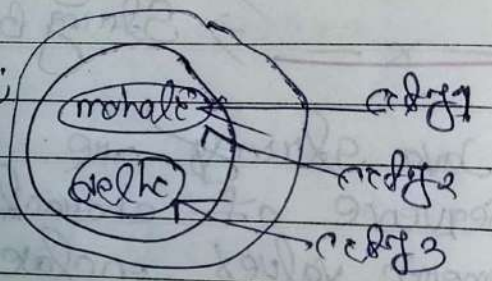
Shot on OnePlus

Why String is immutable?:

→ Because String objects are cashed in String Pool. Since String literals are shared between multiple person. So there is always a risk, where one person's action would affect all another persons.

ex// If one person changes city from "mahali" to "delhi", all other persons will also get affected.

P1 String obj1 = "mahali";
P2 String obj2 = "mahali";
P3 String obj3 = "delhi";



Final keyword:

⇒ final keyword in Java is used to restrict the user.

If used in:
Variable, method, class.

Variable:- once we provide any variable as final we can't be change the value.

method:- If we make method as final we can't be override.

Class:- If we make any class as final we can't extends.

Shot on OnePlus
Amita

Bookbird
Date
Page

== operator

⇒ == operator is used for the reference comparison. means check the both object point to the same memory location.

.equals method:

⇒ .equals method is used for content comparison on strings objs. means check object value.

ex//

```
P S V m ( ) {

    String a = "SAROJ";
    String b = "PAPUNU";

(1)  S.o.P (a.toLowerCase());  // saroj
(2)  S.o.P(b.toUpperCase());  // PAPUNU
(3)  S.o.P(b.concat(a));  // PAPUNU SAROJ

     String c = "   saroja   ";
(4)  S.o.P(c.trim());  // saroja  // No space (01)

     String d = "  ";
(5)  S.o.P(d.isEmpty());  // false

(6)  S.o.P(b.charAt(2));  // P
(7)  S.o.P(a.indexOf('A'));  // 1

(8)  S.o.P(b.equals(a));  // false
(9)  S.o.P(a.replace('A', 'U'));  // SURE).
```

Bookbird

Date
Page

String Reverse Name :-

```
Scanner n = new Scanner(System.in);
S.o.P("Enter name");
String a = n.nextLine();
String b = "";
int c = a.length();
for(int i=c-1; i>=0; i--)
{
    b = b + a.charAt(i);
}
S.o.P(b);
```

How to print sentence of any word ?

```
String name = "Saroj Kumar Sethi";
                0 1 2 3 4 5 6 7 8 9 10 11 12 ...
```

(10) ✓
```
String a = name.substring(6, 11);
S.o.P(a);   // Kumar
```

String Builder :-

(11)
or
```
String b = name.charAt(6);
S.o.P(b);  // K
```

(12)
or
```
name.getChars(6, 9);
S.o.P(name);  // Saroj Kumar Sethi
```

String a = "0 1 2 3 4 5 6 7 8 9 10 mg name sarra";

(12) S.o.P(a.subSequence(3, 7)); // name

(13) S.o.P(a.substring(3, 7)); // name
S.o.P(a.substring(3)); // name sarra

(14) S.o.P(a.replace("mg", "9")); // 9 name sarra
String a = "mg name name sarra";

(15) S.o.P(a.replaceFirst("name", "mv")); // mg mv name sarra

(15) S.o.P(a.replaceAll("mg", "9")); // 9 name sarra
→ add specified regular expression.
S.o.P(a.replaceAll("mg(.)", "TU")); // TUnamesarra
S.o.P(a.replaceAll("mg(.*)", "TU")); // TU erase

16. S.o.P(a.contains("me")); // true

(17) int b = 10;
String s = String.valueOf(b); // string 10

(18) char c = a.toCharArray()[3];
S.o.P(c); // m g name sarra (charat)

(19) int length(); // return string length.

when we use diff.

String:
If we doesn't change the data one or more time then we use String

StringBuffer:
If we want frequently change like calculator then we use buffer.

StringBuffer :-

→ StringBuffer class is used to create mutable String objects. The StringBuffer class in Java is the same as String class except it can be changed.

StringBuffer a = new StringBuffer("Sara");

1) a.append("Kumar");
   S.o.P(a); // Sara Kumar

Constructor: StringBuffer capacity = 16;

→ StringBuffer a = new StringBuffer();
   S.o.P(a.capacity); // 16

→ StringBuffer b = new StringBuffer("Sara");
2) S.o.P(b.capacity); // 21 // 16 + length.

3. S.o.P(a.length()); // 5
4. S.o.P(a.delete(2,5)); // Sa
   S.o.P(a.delete(start)(3)); // Sare
5. S.o.P(a.insert(3,"Alu")); SarAluJ
6. S.o.P(a.replace(2,5,"m")); Sam

Same as String buffer

Difference between String, Buffer, Builder

String :-
→ String memory allocate Heap area.
→ String is create immutable object.
→ If we change the value of string a lot of time then it will allocate more memory.
→ not thread safe.
→ slow Performance.
→ If we don't change the data then use.

String Buffer	String Builder
→ String Buffer memory allocate Heap area.	→ same also //
→ It is mutable object	→ mutable object
→ If we change the value of string a lot of time then it will allocate less time	→ // //
→ all synchronized all methods.	→ It is not synchronized.
→ Slow as string	→ Fast as string Buffer.
→ If we want frequently change then use.	→ //

Exception Handling:

→ exception handling is one of powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.

exception are

→ exception Handling nothing but unexpected / unwanted / abnormal situation that occured at runtime called exception.

→ exception Handling is a machanism to handle runtime errors such as classNotFoundException, IOException, SQLException, RemoteException.

→ we should have an alternative source through which we can handle the exception.

eg/

we work a project 90% already but suddenly current out which project not save i.e we can handle if we have a power backup then we can save then that data.

eg/

```
class A {
    P.S.V.m() {
        int a=10, b=0, c;
        c = a/b; // → Exception: Arithmetic exception
                    we could devide other with zero.
        S.O.P(c);
    }
}
```

Error:- Some time Programmer not any mistake but we got error i.e error.

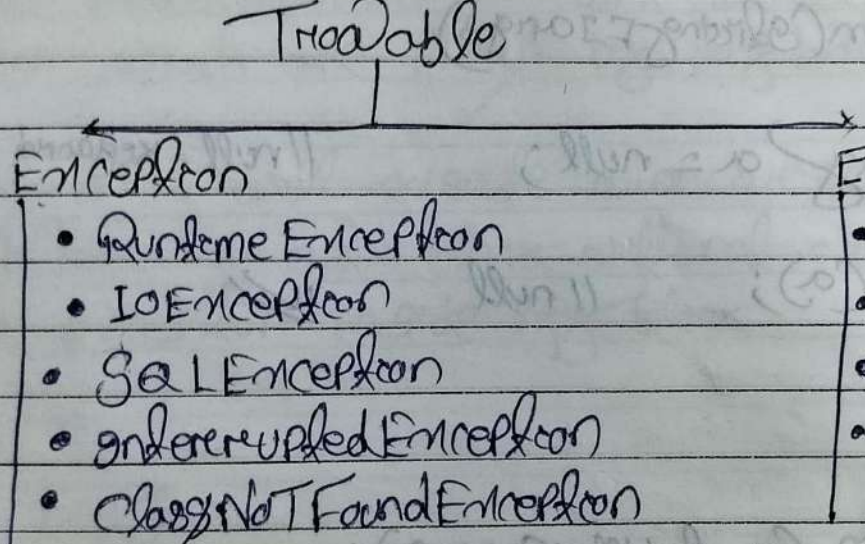
Java Provided some mechanism to check with error

* The object oriented mechanism has Provided the following techniques to check with exception:-

1) Try 2) catch 3) throw 4) throws 5) finally

Exception Hierarchy:-

→ Throwable class is the supere class of exception hierarchy. which contains two sub classes that is
1) exception 2) Error



RuntimeException:-

→ ArithmeticException
→ NullPointerException
→ NumberFormatException
→ IndexOutOfBoundException → ArrayIndexOutOfBoundsException
→ StringIndexOutOfBoundsException

IOException: (input output exception)

→ EofException
→ FileNotFoundException

Test.java (Source code)

↓

[Javac] → Errore (eg: self Errore)

↓ successful compile

↓

[Test.class]

↓

[JVM] → Exception / Errore

```
eg: P.V.S m(String[] args)
{
    String a = null;        // null = keyword use value/lit
    S.o.P(a);               // null
}
```

one

```
try {
    S.o.P(a.toUpperCase());
}
catch (NullPointerException n)
{
    S.o.P("gf exception");   // gf ex
}
```

one

```
String a = "Sara";
try {
    S.o.P(a.toUpperCase()); }
catch (NullPointerException n)
{
    S.o.P("This exception");   // Because
}
// SARA
```

(1) try block :-

→ whenever we write a statement and if the statement is error suspecting statement or risky code then put that code inside of the try block.

→ try is a block that contains only risky code.

eg

```
main(){
try{
    int a=10/0; → (exception)
    S.o.p(a);
}
```

(2) catch block :-

The main purpose of catch block is to handle the exception which are thrown by try block.

→ catch block will execute when we got exception on try block. and catch block will not execute when there is no exception inside try block.

eg

```
try{
    int a=10/20;        (not (exception)
    S.o.p(a);
}
X catch(ArithmeticException n); X  so catch not execute.
{
    S.o.p("sonu");
}
```

Shot on OnePlus
Anita

Bookbird

Date
Page

try & catch :

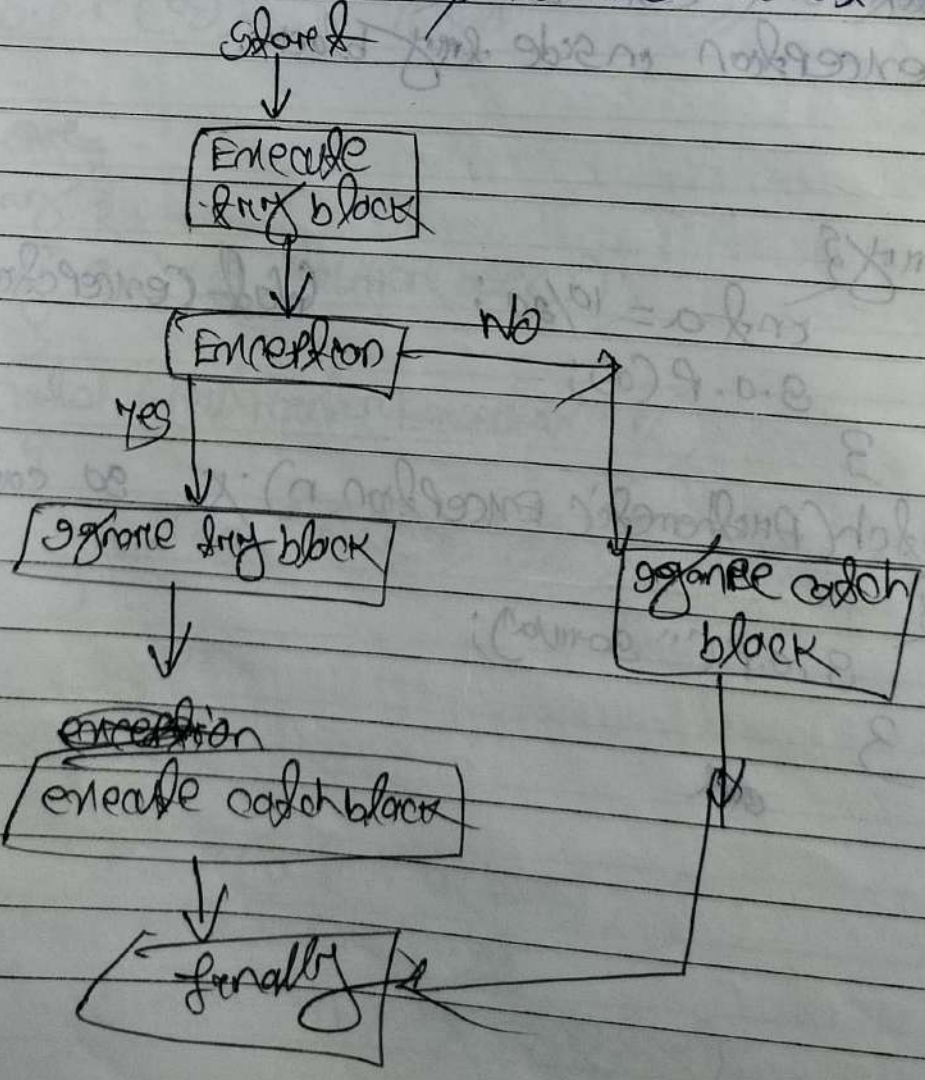
```
try {
    statement 1;
    stmt 2; //exception
    stmt 3;
}
catch(AE n)
{
    stmt 4;
}
stmt 5;
```

exception → 1, 4, 5

No exception → 1, 2, 3, 5

3) Finally block :-

→ finally block is used to execute the necessary code of the program. when an exception is handled are not finally execu



```
try
{
    stmt 1;
    stmt 2;
    stmt 3;
}
catch(Exception n)
{
    stmt 4;
}
finally
{
    stmt 5;
}
stmt 6;
```

No exception:
case :- 1, 2, 3, 5, 6
(exception → 2)
case :→ 1, 4, 5, 6
(No match exception)
case :- 1, 5,

multiple try-catch:

```
main() {
try {
    int a = 40/0;
    S.o.p(a);
}
catch(ArithmeticException e)
{   S.o.p("can't devide by zero");
}
```

```
try
{
    int [] a = {10, 20, 30, 40};
    S.o.p(a[5]);
}
catch (ArrayIndexOutOfBoundsException e)
{
    S.o.p("Beyond the array limit");
}
```

Bookbird
Date
Page

checked

→ The exception which are checked by compiler for smooth execution of program at the runtime called checked exception.
→ we get error at time compilation i.e called checked
eg
IOException
SQLException
FileNotFoundException etc.
all is checked at compile time.

unchecked

→ The exception which are not checked by compiler and it dynamically taken care by JVM.
→ once compiler completed then we get error at the run time i.e called unchecked.
eg
ArithmeticException
NPException
ArrayOOBException etc.
all is not checked compile time but they are checked at runtime.

---

error)
outOfMemoryError
VirtualMachineError, AssertionError.

Final

→ final is a key word.
→ it is used for variable, methods classes

finally

→ finally is a block
→ it is always execute, when exception handle or not

finalize

→ finalize is a method.
→ it is used to deallocate the resources which is allocated by unused object.

---

which exception should be declared?
→ checked only because
unchecked ex is occur under go we can correct our code

throw keyword:-

→ throw is a keyword which is used to throw the user defined or customized exception object to the JVM explicitly for that purpose we use throw keyword

→ suppose we write our code & the any exception that are error that checked at the of runtime and we get some message. that message we can create a custom exceptions by using throw keyword. eg:- ArithmeticException

→ also we can throw ArithmeticException explicitly.

```
p.s.v.m1(int age)
{
    if(age<18)
    {
        throw new ArithmeticException("person is not eligible");
    }
    else { s.o.p("eligible for vote"); }

p.s.v.main()
{
    m1(15);
}
```

// Exception in thread "main" java.lang.ArithmeticException: person is not eligible.

throws keyword :-

→ throws is a keyword is used when we doesn't want to handle the exception and they to send the exception to the JVM (JVM or methods).
→ throws keyword is used to declare an exception.
→ throws keyword is used to declare as well as ~~EX~~ we can throw from a method. and we can use throws keyword in method signature.
eg IOException, ArithmeticException.

syntax:

```
returntype_methodname() throws Exception_classname{
    ==
}
```

throw

→ throw keyword is used to throw an exception object explicitly.
eg/ void m1(){
    throw new AE();
}
→ throw keyword always present inside method body.
→ we can throw only one exception at a time.
eg// throw new AE()

→ Throw is followed by an object (instance).

throws

→ throws keyword is used to declare an exception as well as by pass the caller.
eg// void m1() throws AE
{
    ==
}
→ throws keyword always used method signature.
→ we can handle multiple exceptions using throws keyword.
eg/
m1() throws AE, NPE, SQL
{
    ==
}
→ Throws is followed by class.

Exception:-

→ An exception is caused by our Program.

→ Exceptions are recoverable.

e.g.

```
try{
    10/0;
}
catch(Ex n)
{ S.O.P }
```

→ In Java exception are classified as checked are unchecked type.

Error:-

→ An errors are not caused by our Program

→ Errors are not recoverable.

e.g.

→ out of memory

→ Errors are only unchecked type in Java.

throws eg:-

Bookbird
Date
Page

File Handling :-

→ File handling defines as How we can read and create data on a file.
→ Java IO Package contains all the classes through which we can use to all input & output operations on file.

Stream :

→ Stream is a sequence of data.
on the basis of java.io package all the classes divided into two Stream.
→ Byte → character.

methods of File handling

(i) canRead()
(ii) canWrite()
(iii) createNewFile()
(iv) Delete()
(v) Exists()
(vi) Length()
(vii) getName()
(viii) getAbsolutePath()
(ix) mkdir()
(x) List()
(xi) Read()
(xii) write()
(xiii) renameTo()

File Handling classes :-

(i) File
(ii) FileReader
(iii) FileWriter
(iv) FileInputStream
(v) FileOutputStream
(vi) BufferedInputStream
(vii) BufferedOutputStream

operations of file :

(i) Create file
(ii) get file information
(iii) read
(iv) write.

create file:

```
eg
class create_file {
    P.S.V.m() IOException {
        File a = new File("C:\\Users\\sanu\\OneDrive\\
                Desktop\\sanu.txt");
        if(a.createNewFile())
        {
            S.o.P("file successfully created");
        }
        else
        {
            S.o.P("file already exist");
        }
    }}
eg import java.io.File;
    class info {
        P.S.V.m() {
        File a = new ("C:\\Users\\sanu\\OneDrive\\Desktop
                \\sanu.txt");
        if(a.exists())
        {
            S.o.P("file name:"+a.getName());
            S.o.P("file location:"+a.getAbsolutePath());
            S.o.P("file writable:"+a.canWrite());
            S.o.P("file read:"+a.canRead());
            S.o.P("file size:"+a.length());
        }
        else { S.o.P("file doesn't exist"); }
    }
```

Static Keyword:-

→ Static keyword is used for memory management mainly.
We can use with variables, methods, blocks and nested class or inner class.

→ Static variables:-

→ A static variable is declared with help static keyword inside of class but outside of of method, constructor, block.

→ static keyword is used to refer to the common property of all object.

→ If we execute static no need to create object call through class name.

eg:

```
class Sana {
    int a;
    String name;
static String college = "KCA";
void show() {
    S.o.P("a=" + name + " " + college); }
P.S.V.m () {
Sana n = new Sana(10, "Sana");
Sana n2 = new Sana(11, "Samba");
n.show();
n2.show();
```

// 10 Sana KCA
11 Samba KCA

Static method:

→ A method is declared with help of static keyword that is called static method.
→ static method which belongs to the class not an object.
→ If we want to access no need to create object. the class
→ static method only access static data. not non-static.
→ It is not refer to the current keyword.

eg:

```
class Sarv {

    static void add() {
        S.O.P("Hi");
    }
P.S.V.m() {
}
Sarv.add();
}
```

```
class A {
    static int a=0;
A() {
    b++;
    S.O.P(b)
P.S.V.m() {
A a1=new A();
A a2=new A();
A a3=new A();
}}
```

```
1
2
3
```

Shot on OnePlus
Amita

Static block:-

Static block is a block of statement inside of class that is executed before than main method at the time of class loading when a class is loaded before than main method.

eg/
```
class A {
    static {
        S.o.P("Hellaa");
    }
    P.S.V.m() {
        S.o.P("Hi...");
    }
}
```
// Hellaa
Hi.....

Why we use java main method static ?
Because without creating object is not required to call a static method.
→ If we create non static method JVM creates an object first then call main() method so it will lead to extra memory allocation.

Can we execute a program without main() method?
No, but it will possible till JDK 1.6 → JDK 1.7 do not possible

Package in Java:

→ A Package arrange number of classes, interfaces and sub-Package of same type into a particular group.
→ Package is nothing but folder in windows.

Types:

Pre-defined
- java.lang
- java.util
- java.io
- java.applet
- java.awt
- java.net
- java.SQL

User-defined
- Package P1
- Package add
- Package myPack

Access modifier:-

Public :- access the all the classes, Packages, outside Package in our app.
Private :- only access classes which is defined.
Protected :- access some Package or subclasses in diff Pack
Default :- it is acces some Package and class.

Advantage:
(i) Reusability
(ii) security
(iii) fast searching
(iv) naming conflicting
(v) Hiding

Disadvantage:
we can't Pass Parameter to Package.

Shot on OnePlus
Amita

(1) Java.lang :-
It is default package also known as heart of Java. without using this package we can't write any single code and we no need to import this package.
ex// system, string, object, integer etc. ---

(2) Java.util
→ This package is used to implement data structure also called collection framework.
ex// LinkedList, Stack, Vector, HashSet, TreeSet etc.

(3) Java.io :
To use input/output operations on file.
ex// File, FileWriter, FileReader etc.

(4) Java.applet :- use to develop GUI related Package.
ex// Applet.

(5) Java.awt :- use to develop GUI related Package.
It is standalone Programme :
ex// Frame, Button ...

6. Java.net : ex// URL, InetAddress.

(7) Java.SQL :- Connection, Statement, ResultSet ...

8. Javax.swing : JFrame, JButton ...

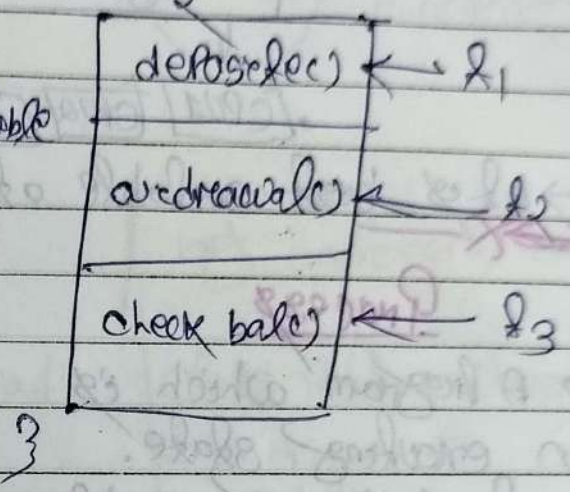
User defined Package :
These package are created by Programmer are user for their own use.

Syntax Package Package_name;

Multithreading:

→ multithreading is a Process to execute multiple threads at the same time without dependency of other threads called multithreading.

→ It is mostly used in games, animation etc.
→ multithreading is best suitable at Programming level



What is thread?

→ Thread is a Pre-defined class
→ Thread is a lightweight subProcess, the smallest unit of Processing, so it is independent execute.
→ If we get exception in one thread, it doesn't affect another threads

How to create thread in Java?

→ By extending Tread class
→ By implementing Runnable interface.

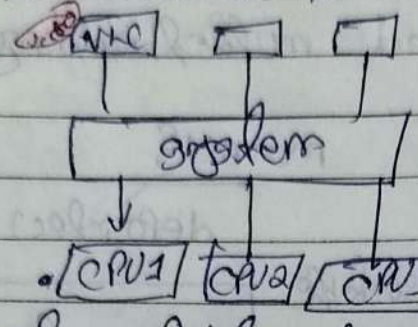
multitasking:

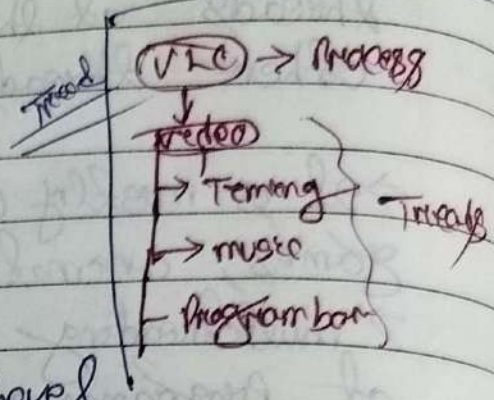
→ Performing multiple task at single time.
→ increases the Performance of CPU.
→ 2 types :-

(1) Process based multitasking (multiProcessing)
(2) Tread based multitasking (multi Treading)

multi Processing:

→ When one system is connected to multiple Processors in order to complete the task.





→ It is best suitable at system level

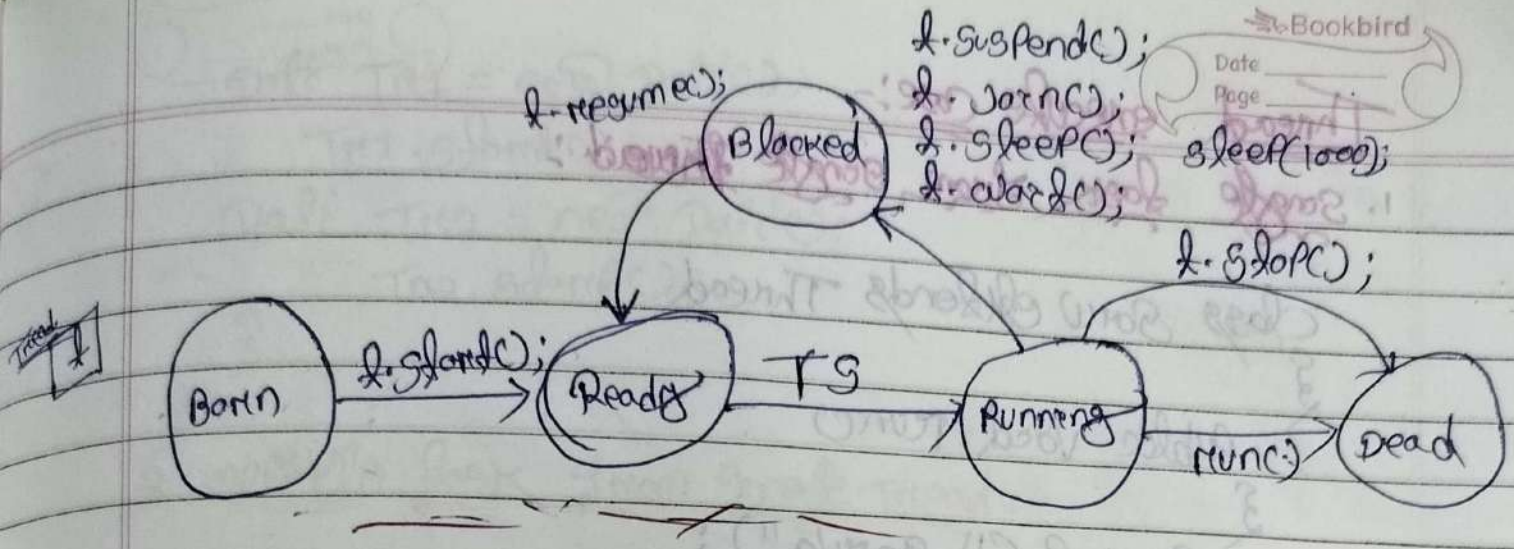
Process	Thread
1. → A Program which is in executing state.	→ It is subpart of Process (Small task of Process).
2. → It is heavy weight	→ It is light weight
3. It is more time	→ It is less time
4. each Process has different address space.	→ each Thread has same address space.
5. Process are not dependent on each other.	→ Threads are dependent on each other

Thread life cycle:-

As we know a thread is well known for independent execute. During the life cycle a thread can move from different states.

1. New states (Born)
2. Runnable states (Ready)
3. Running states (Execution)
4. Waiting states (Blocked)
5. Dead states (End)

Shot on OnePlus
Amita😍



2 ways of create thread:

By extending Thread class:-

```
class Test extends Thread
{
    (1) override
    public void run()
    {
        // Task
    }
    P.S.V.m()
    {
        (2) create object of the class
        Test t = new Test();
        (3) invoke the Thread:
        t.start();
    }
}
```

By implements interface

```
class Test implements Runnable
{
    (1) override
    public void run()
    {
        // Task
    }
    P.S.V.m()
    {
        (2) create object
        Test t = new Test();
        (3) create object Thread class
        Thread th = new Thread(t);
        (4) invoke Thread
        th.start();
    }
}
```

By interface:-

```
interface Runnable
{
    public abstract void run();
}
```

Thread create use:-

1. Single task from single thread :

```
class Garv extends Thread
{
  Public void run()
  {
    S.o.P ("Garva");
  }
}
Public class
{
  P.S.v.m ()
  {
    Garv Thread1 = new Garv();
    Thread1.Start();
  }
}
                    // Garva
```

2. Single task from multiple Thread :-

```
class Dali extends Thread
{
  Public void run()
  {
    S.o.P ("Daling");
  }
}
Public class
{
  P.S.v.m()
  {
```

```
Dali Thr1 = new Dali();
    Thr1.start();
Dali Thr2 = new Dali();
    Thr2.start();
  }
}
```

3. multiple task from single Thread :-

e.g. it is not applicable in because one de thread execute ek one by one by execute.
e.g. one VLC vedio Player is one thread
so task is running, music, scrolbar etc is multi task
so once we execute it all is vedio play after music, after scroll so i.e not use in thread.

4. multiple task from multiple thread :-

```
class Saru extends Thread
{
    Public void run()
    {
        S.o.P("my name Saruu");
    }
}
class maru extends Thread
{
    Public void run()
    {
        S.o.P("my name maruu");
    }
}
```

Shot on OnePlus
Anita

```
Public class Alv
{
  P.S.V.m()
  {
    Sarv m = new Sarv();
      m.start();
    marv n = new marv();
    m.start();
    // one ae start thread sub thread dead ae
    // can't declare same thread again.
    //n.start(); (exception)
  }
}
```

→ Basic method → t.run(), start(), currentThread(), sl

→ Naming method :-
  → getName(), setName(String name)

→ Demon Thread method :-
  → setDemon(), getDemon(boolean b)

→ Priority thread method :-
  → getPriority(); setPriority(int pr)

→ Preventing Priority thread execution methods :-
  → sleep(), yield(), join(), suspend(), resume(), stop(), destroy()

→ Interrupting thread method :-
  → interrupt(), isInterrupted(), interrupted()

→ Inter thread communication :-

```
class object
{
  abc()
  notify()
  notifyAll()
}
```

Naming:

How to change main Thread name:-

```
Public class Sara
{
    P.S.V.m()
    {
        S.o.P("Before:"+Thread.currentThread().getName());

        Thread.currentThread().getName("Sara");
        S.o.P("After:"+Thread.currentThread().getName());
        S.o.P(10/0);   (check exception)
    }                  // Before: main
}                      // after: Sara
```

Deman Thread:

Which run in the background of another thread.

→ use - gc provides service to the Threads.

eg:

Garbage collection, finalizer, ...

i) Public final void setDeman(boolean b)

ii) Public final boolean isDeman()

```
class Test extends Thread
{
    Public void run()
    {   S.o.P("child");
    }
}
P.S.V.m()
{
    S.o.P("main");
    Test t = new Test();
    t.setDeman(true);
    t.start();
```

case:- we have create demon thread before start. If we add after so I will exception.

Sleep() method:-

Sleep is a static method of Thread class which throws checked exception i.e InterruptedException.

The main purpose of sleep method is to put a thread into temporary waiting state.

Syntax:-

Thread.Sleep(1000)

public static native void sleep(long mls)

mls in sec

Thread.Sleep(1000)

or

Thread t = new Thread();
t.sleep(1000);

eg:-

```
class Test extends Thread
{
  public void run()
  {
    for(int i=1; i<=5; i++)
    {
      try
      {
        Thread.sleep(1000);
        S.o.p(i);
      }
      catch(Exception e)
      {
        S.o.p(e);
      }
    }
  }
}
```

```
p.s.v m(){                    //1
    Test t = new Test();      //2
    t.start();                //3
}                             //4
}                             //5
```

Yield() method:

→ Yield is a static method of Thread class that allows to run another thread which has same priority and stops the current execute thread.

syntax:-

```
Public static native void yield();
```

working:-

Thread provides the hint to the thread scheduler then it depends on Thread-scheduler to accept or ignore the hint.

eg

```
class Test extends Thread
{
    Public void run()
    { String name = Thread.currentThread().getName();
      for(int i = 1; i<=5; i++)
      {
          S.o.p(name);
          Thread.yield();
      }
    }
}
```

Shot on OnePlus
Amita

```
class Test1 extends Thread
{
    public void run()
    {
        String name = Thread.currentThread().getName();
        for(int i=1;i<=5;i++)
        {
            s.o.p(name)
        }
    }
public class Demo
{
    P.S.V.M()
    {
        Test t = new Test();
        t.start();
        Test1 r = new Test1();
        r.start();
    }
}
```

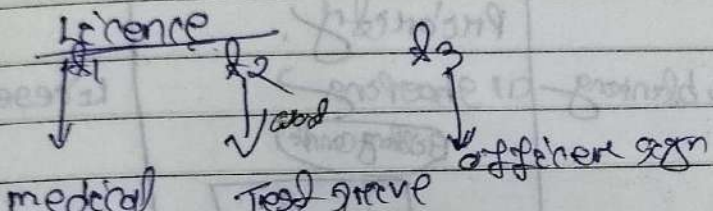
// out put of scheduler accord // not accord randomly comp

Thread 0	
Thread 1	0
Thread 1	1
Thread 1	1
Thread 1	1
Thread 1	0
Thread 0	0
" 0	0
" 0	1
" 0	1
	0

join() method :

⇒ If a thread wants to wait for another thread to complete its task, then we should use join() method.
⇒ join() also throw a checked exception.
i.e InterruptedException

eg:

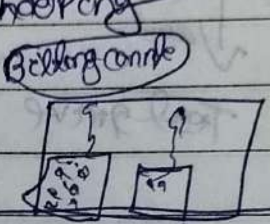


Syntax:-

a. public final void join() throws InterruptedException
b. " " synchronized void join(long ms) . .

```
class Test extends Thread
{
  public void run()
  {
    for(int i=1; i<=5; i++)
    {
      S.o.p(i);
    }
  }
}
public class Test1
{
  p.s.v.m() throws Exception
  {
    Test t = new Test();
    t.start();
    t.join();
    for(int i=0; i<=10; i++)
    {
      S.o.p(i);
    }
  }
}
```

Shot on OnePlus
Amita

Properties:	sleep()	yield()	join()
Purpose	If any thread does not want to perform any operation for particular time.	It stops the current executing thread & provide chance to another threads of same priority.	If a thread wants to wait for another thread to complete task
ex	Timer, PPT, blinking bulbs	ex shopping, 	Licence. Process.
Exception	InterruptedException yes	NO exception	yes (InterruptedException)

isAlive() :-

isAlive() is a pre-defined method of thread class through which can verify whether a thread isAlive or not.

→ If thread isAlive then it will return true otherwise false.

→ If we use isAlive() before start it will print false after start print true.