2/10/24
Tuesday.

Abstract Class:-

- A class declared with abstract keyword is called Abstract class

```
abstract class Demo
{
}
```

- Abstract class is like a normal class which contains data-mem, constructor, block and method but only difference is we cannot create obj for abstract cls & we can declare abstract method.
- abstract class can cantain concret() & also abstract().
- We can't creat obj for abstract cls but we can use it as a reference to refere sub-class obj. (up-casting).
- abstract cls is used to achieve Abstraction
- When we declare abstract cls it is not mandatory to have abstract() i.e we can have abstract class without any abstract() [with only concrete method].
- Real time example for abstract cls which contains only concrete() is HttpServlet.
- If the class contains abstract() then it is mandatory to declare class as abstract.
- By using abstract-cls we can't achieve 100%. Abstraction.

Ex:-

```
abstract class Keyboard
{
 void pressQ()
 {
  s.o.p("print Q");
 }
 void pressR()
 {
  s.o.p("print R");
 }
 abstract void pressEnter();
}
class NotePad extends Keyboard
{
 @Override
 void pressEnter()
 {
  s.o.p("goes to nextime");
 }
}
class MusicPlayer extends Keyboard
{
 @Override
 void pressEnter()
 {
  s.o.p("play|pause music");
 }
}
class Test
{
 psvm()
 {
  Keyboard k1 = new keyboard(); // cannot instan
                                Error:
  Keyboard k1 = new NotePad();
   k1.pressR();
   k1.pressEnter();
```

```
    Keyboard k2 = new MusicPlayer();
      k2.pressEnter();
  }
}
```

Q&A.

01. Can we overload abstract()?

→ Yes

```
public abstract class Facebook
{
abstract void login(long contact, int otp);
abstract void login(String uname, String pwd);
}
```

02. Can we Override abstract()?

→ Yes. it is mandatory to override abstract().

03. Can we declare static(r) as abstract?

→ No. we cannot declare static method as abstract.

```
//Error saying
->The abs() login in type Facebook can only set a visibility modifier; one of public or protected
Public abstract class facebook
{
abstract void login(long contact, int otp);
static abstract void login(String uname, String pwd);
}
```

04. How to avoid creation of obj for a cls?

→ By declaring class as abstract class.

NOTE:-

-100%. Abstraction is not possible by using abstract-class but it is possible by using interface.

Interface:—

- It is an intermediate blw service provider & the consumer.
- Interfau contains only abstract().
- Interfau is declare with interface keyword.
- Interfau & abstract-class acts as a non-primitive datatype.
- Abstract(), abstract cls, interfau are used to achieve Abstraction but by using Interfau we can achieve 100% abstraction.
- We can't creau object for interfau but interfau can refer to the implemetation class object.

Ex:-
```
interface Switch
{
}
```

- interfau is also known as Rules Repository or Coding contract.
- A class can inherit using implements keywor

Ex:-
```
interface Switch
{
}
Class Tubelight implements Switch
{
}
```

- A class which inherits interfau is called implementation class.
- An interfau can inherit another interfau by using extends keyword.

Ex:-
```
interface Inf1
{
}
interface Inf2 extends Inf1
{
}
```

class ——extends——→ class

class ——implements——→ interface

interface ——extends——→ interface

interface ——X——→ class → (Not possible)

28/08/24
Wednesday

Note:-

- A cls can inherit interface by using implements keyword but a interface cannot inherit cls either by using extends keyword/by using implements keyword.

- Interface does not have any constructor.

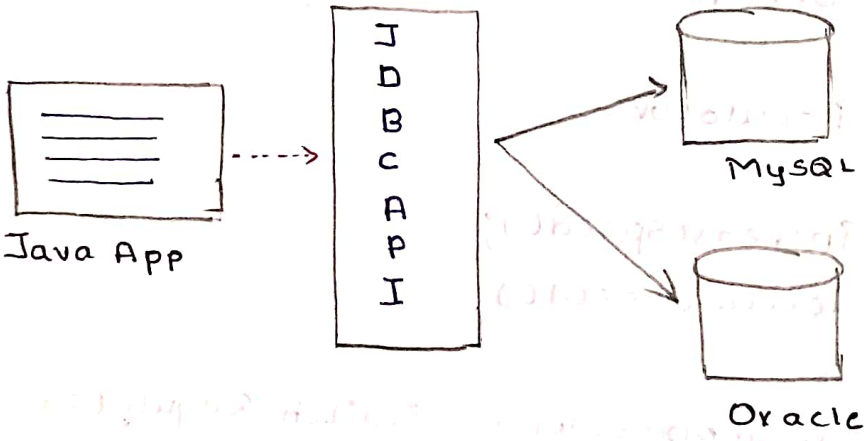
Ex:- Switch is an interface b/w light & user.

Remote is an interface b/w TV & user.

Touch
Mouse is an interface b/w slm & user.

Real-time example in Java.

- JDBC API is an interface b/w javaApplication and database Server.



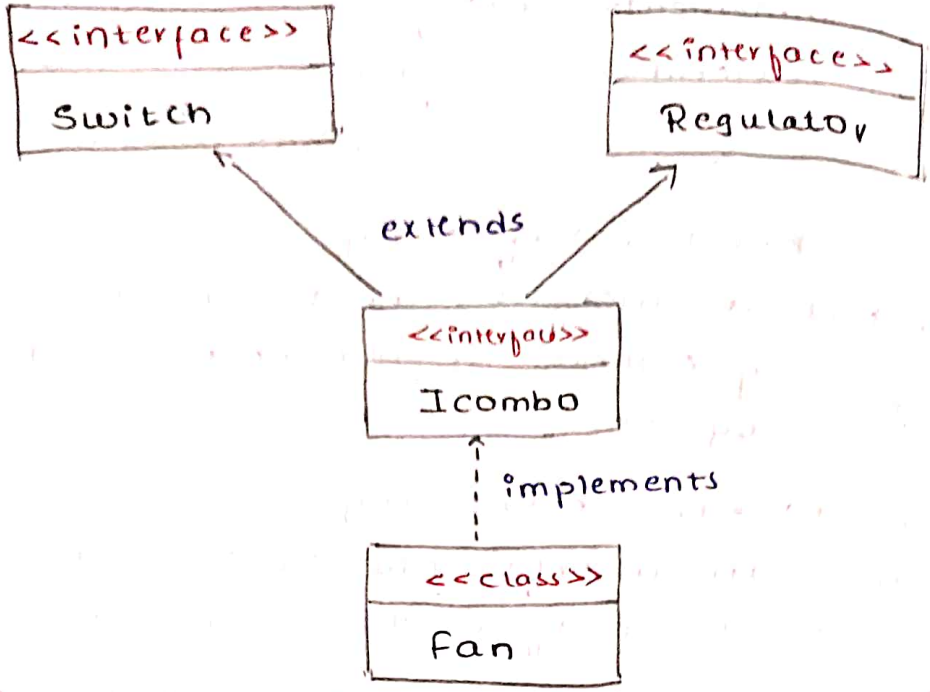
- We can achieve Multiple-Inheritance by using interface.

- If we declare any method in interface & terminated with semicolon then by default it takes public abstract.

Ex:-

```
public interface switch
{
  void sOn();
  void sOff();
}
```

... default public-abstract




```
Public interface Switch
 {
  void SON();
  void SOFF();
 }
interface Regulator
 {
  void increaseSpeed();
  void decreaseSpeed();
  }
interface Icombo extends Switch, Regulator
 {
  }
Public class fan implements Icombo
 {
  @Override
   public void SON()
    {
    sout("fan switch ON");
    }@Override
   Public void SOFF()
    {sout("fan switch OFF");
```

```
@Override
public void increaseSpeed()
 {
sout("Fan speed increased");
 }
@Override
 public void decreaseSpeed()
 {
sout("Fan speed decreased");
 }

public class Testfan
 {
  PSVM()
  {
   ICombo ic = new Fan();
     ic.SON();
     ic.increaseSpeed();
     ic.decreaseSpeed();
     ic.SOFF();
  }
}
```

Output: —

Fan switch on.
Fan Speed increased
Fan speed decreased
Fan switch off.

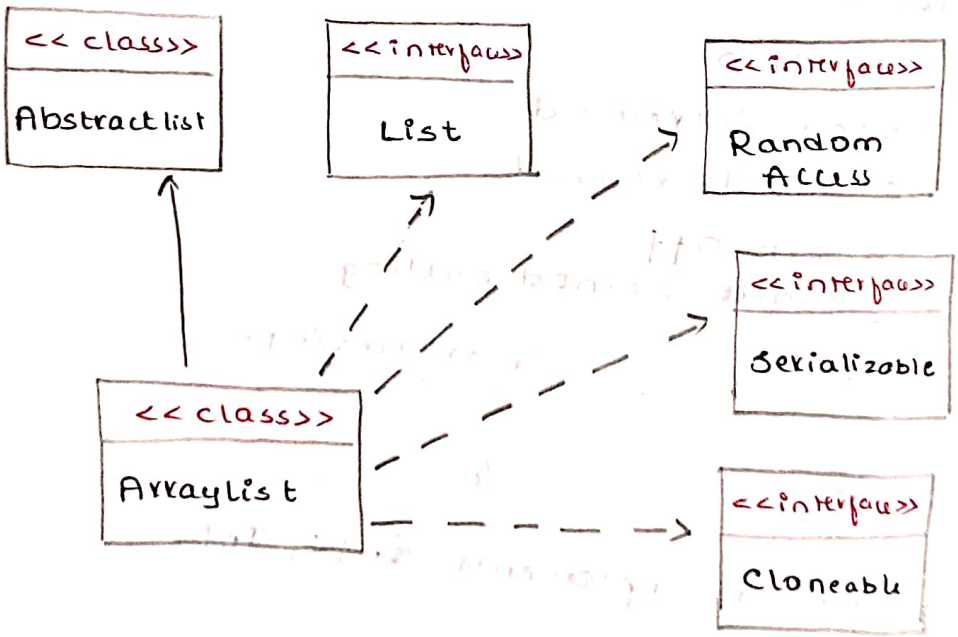
- Ambiguity while method calling

```
interface int1
 {

 }

interface int2
 {

 }

class child implements int1, int2
  {

     child()
     {
      super();
     }
  }
```

- A class can extends to another cls & also it can implements multiple interface.

Ex:-
```
interface Inf1
{

}
interface Inf2
{

}
interface
Class Parent
{

}
Class child extends Parent implements Inf1, Inf
{
}

Class child implements Inf1, Inf2 extends Par
{
    X (Not possible)
}
```

- In real-tym in java there are many classes which extends another-cls & implements multiple interfa



- We can't declare constructor inside interfa
- Interfa does not support var if we declare any var then by default it becomes public static final.

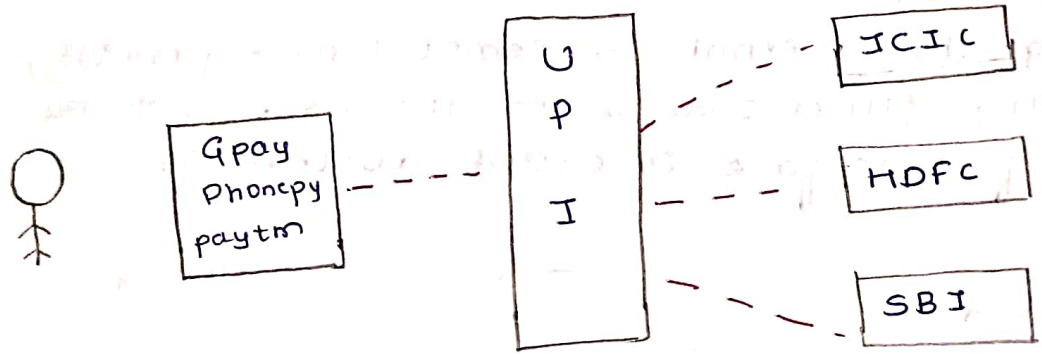
29/04/24
Thursday

Ex:- interface AppConstant
{
int a = 10;
}

public static final int a = 10;

- Interface is an coading contract which gives a contract to implemetation classes to provide the implementation.
- From JDK1.2 onwards we can have static concrete().
- From JDK 1.8 onwards we can have default concrete().

Ex:-
Application specific



*Types of interface:-

(i) Marker interface
(ii) Functional interface
(iii) Regular interface.

- Marker interface:-

- It is an empty interface.
- Marker interface does not have any methods.
- Marker interface is used to indicate JVM about an obj ability
- There are many marker interface are present in jave there are
  (i) RandomAccess
  (ii) Clonable
  (iii) java.io.serializable or serializable

Functional-interface:-

- It is an interfau which contains only one abstract().
- It is used to indicate main bussiness functionality of an obj.

Ex:-

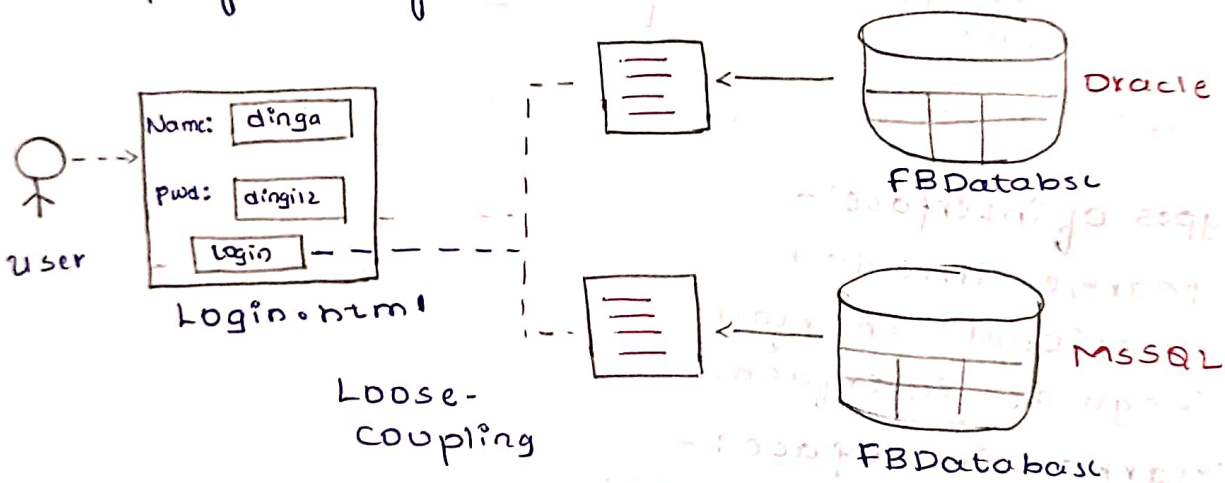
Runnable ---> run();
Comprable ---> compareTo();
Comparator ---> compare();

Regular-interface:-

- It is a normal-interfau which may contain any no of abstract().

Abstraction:-

- Hiding the internal implemetation & providing neussary functionality to the user with the help of interface is called abstraction.



- The output of abstract is Loose-coupling.

→ Loose-coupling.

- Change in implementation which does-not effect the user is called loose-coupling.

- The real-tym example for Abstract is JDBC API
- JDBCAPI is used to achieve loose-coupling blwn Java application & Database server.

Class Loading:-

- It is a process of bringing .cls file to JVM memory.
- To load the class the cls must contain main method.
- we can load the cls in 2 different ways
(i) By calling any of the Java.mem of the cls
(ii) Manually we can load the cls by using forName()

→ forName()
- It is a static() which is present in a class by the name class.

Ex:-
```
class Class
{
  Public static getClass forName(String FQCN)
      throws ClassNotFoundException
  {

  }
}
```

- It take String fullyQualifiedClassName as a parameter
- when ever we use forName() it throws an exception called ClassNotFoundException.

* Syntax:-

```
Public final class Class
```

→ Static block.

- It is used to initialize the static var.
- Static block gets loading & excutated @ the time of class loading.

Ex:-

01.
```
public class Emp
  Static
  $
  s.o.pc" I am a Static block");
  }
```
//output:- main-method
Error saying main-method not found

02.
```
public class Emp
  Static
  $
  s.o.pc" I am a static block");
  }
  psvm()
  $
  s.o.p(" I am a main method");
  }
```
//OUTPUT:-

I am a Static block
I am a main method

- Static blocks are executed only once @ the time of class loading
- Any-configuration has to be executed @ the time of application/cls loading then we have 2 use static-block.

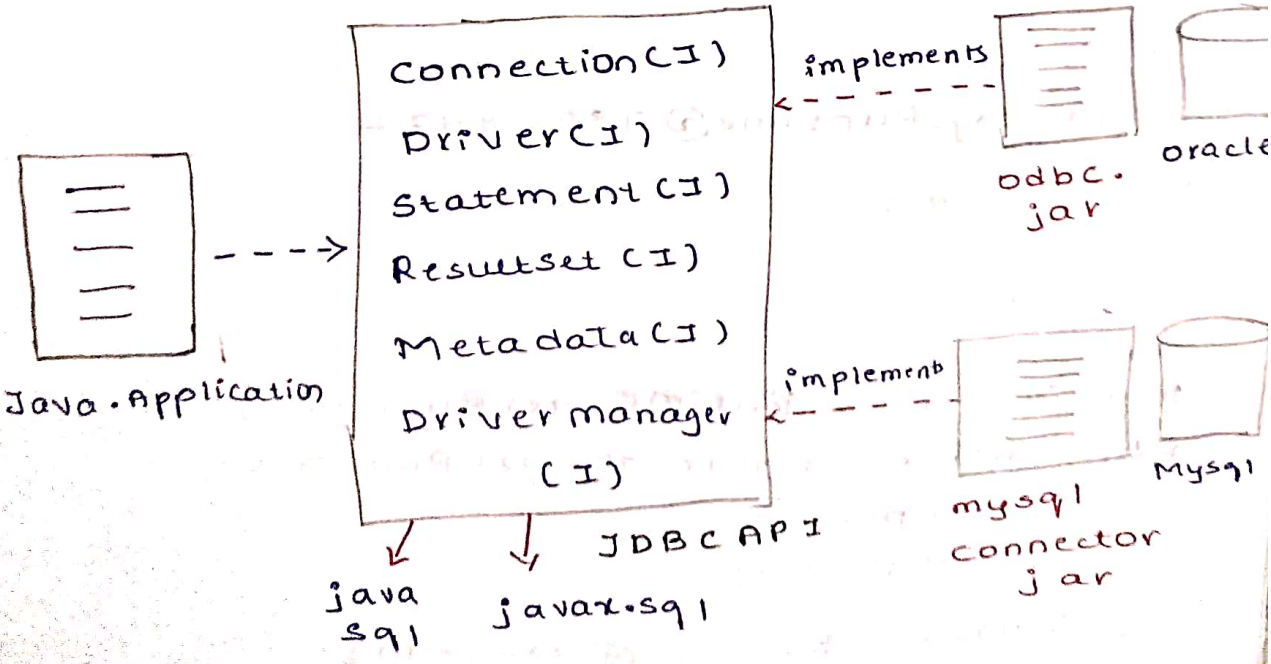
Manual way of loading the class by using public forName().

```
package com.pentagon.empApp;

public class Emp
 {
  String ename = "gunda";
  static
  {
  Emp e1 = new Emp();
 s.o.p("emp name:" + e1.ename);
 s.o.p(" I am static block of emp cls");
  }
}

public class ClassLoadDemo
 {
psvm() throws ClassNotFoundException
 {
  Class.forName("com.pentagon.empApp.Emp");
 }
}
```

//OUTPUT

emp name : gunda

I am a static block of emp cls.



2/9/21
Monday

Design Pattern:-

- It is an optimised solution for commonly re-occuring prblm (design problm / design needs).

Ex:- MVC architecture.

- There are 2 types of design pattern

(I) Factory Design Pattern

(II) Creational Design Pattern.

↳ Ex:- Single tone Design Pattern.

↳ Design this class in such a way that one one obj can be created

if we create 2 obj then try to print reference var then o/p will have same address

Ex:-

```
public class
{
 PSVM()
 {
```

Runtime is present in java.lang package.

→ way to create obj for static var

```
Runtime rt = Runtime.getRuntime();
    S.O.P(rt);
 }
}
```

O/p:-

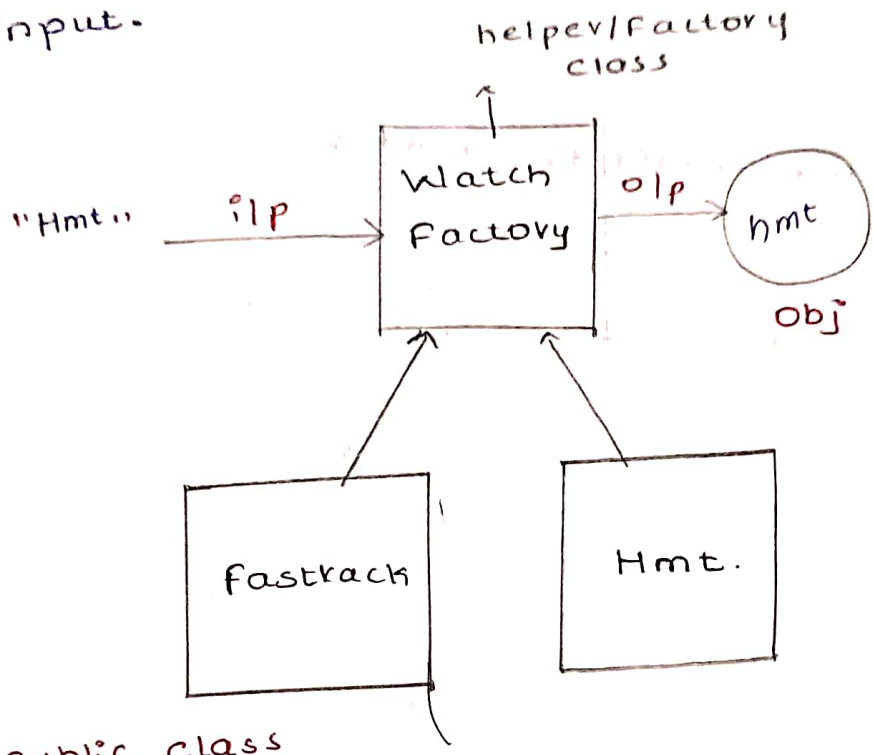
java.lang.Runtime@7852e922

```
Public class
{
  PSVM()
  {
Runtime rt = Rutime.getRuntime();
Runtime rt1 = Rutime.getRuntime();
  i    S.O.P(rt);
       S.O.P(rt1);
 }
}
```

O/p:- java.lang.Runtime@7852e922
java.lang.Runtime@7852e922

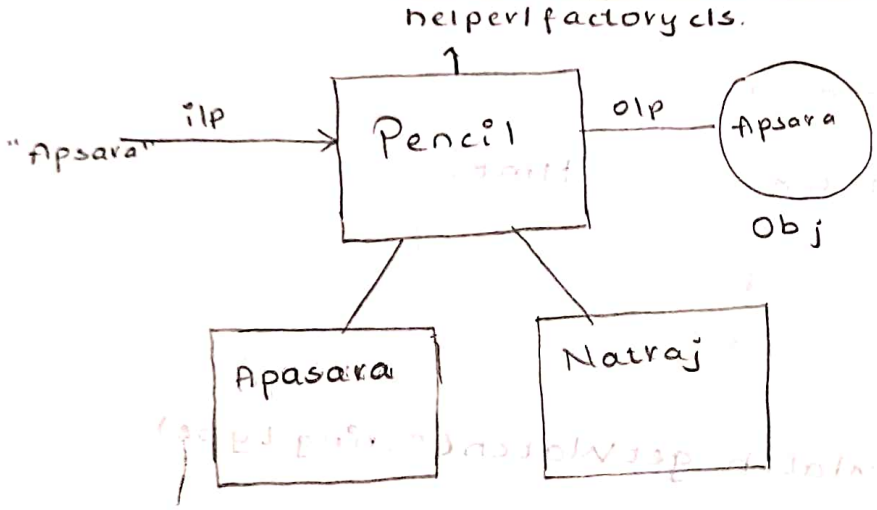
Factory Design Patterns -

- Factory is the one which creates & returns the different obj of same type.
- Factory method always takes String as a input.
-




```
public class
{
 RSHMT()
public static Watch getWatch(String type)
{
if(type.equalsIgnoreCase("Fastrack"))
  {
  return new Fastrack();
  }
 else if(type.equalsIgnoreCase("HMt"))
  {
  return new Hmt();
  }
else
    s.o.p("No Such Watch Found");
    return null;
  }
}
```

```
Class Watch
{
}
class Fastrack extends Watch
{
}
class Hmt extends Watch
{
}
```




```
public class
{
public static Pencil getPencil(String type)
{
if(type.EqualsIgnoreCase("Apasara")
{
return new Apasara();
}
else if(type.EqualIgnoreCase("Natraj")
{
return new Natraj();
}
else
{
System.o.p("not found");
return null;
}
}
```

```
class Pencil
{
}
class Apasara extends Pencil
{
}
class Natraj extends Pencil
{
}
```

- factory design pattern is associated with 3 diff type of logic

1. Implementation logic
2. Object creation logic
3. Consumer/utilization logic

→ Implementation logic:-

- It is a basic found fundamental logic which contains only implementation according to which an implementatic obj has 2 bc created

→ Object creation logic:-

It is used to create an implemtation obj according to implementation logic by using factor/helper method within factory class

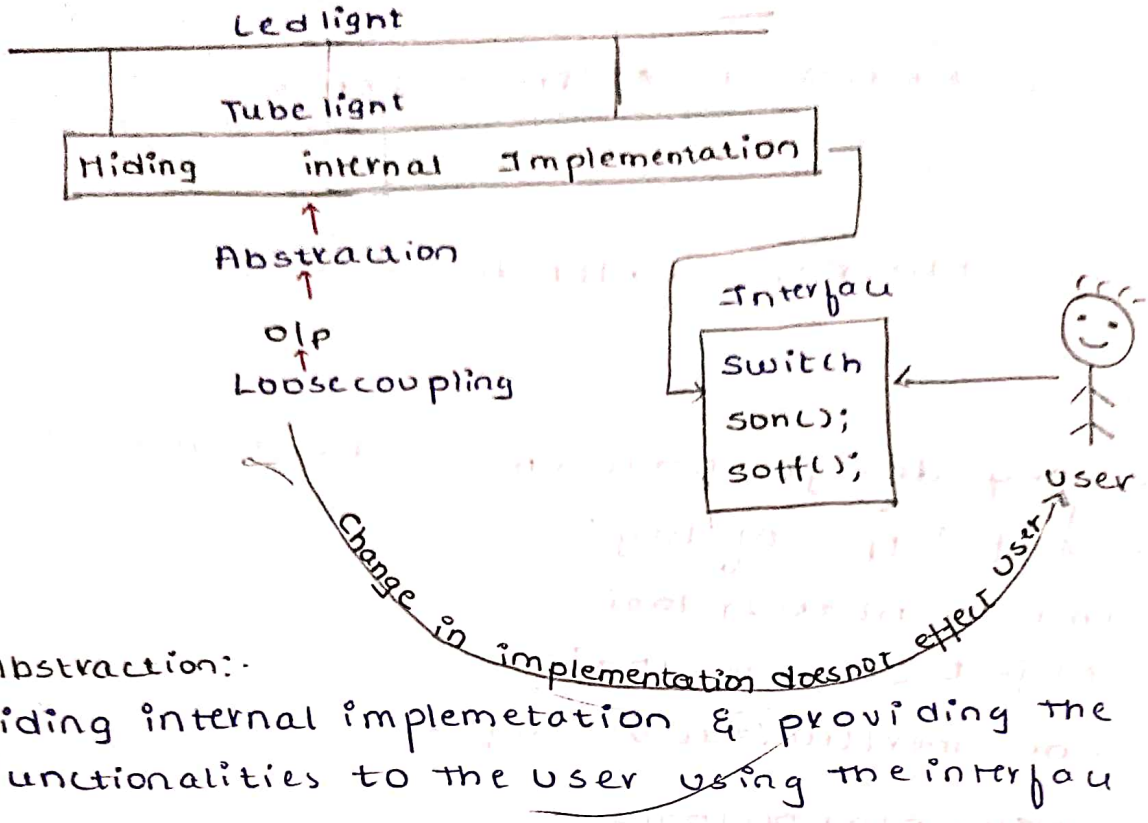
→ Consumer/utilization logic:-

It is the most imp logic which is used to access the functionalies from the implement -ation.

NOTE:-

Without writing consumer/utilization logic non of the implemtation will work.

Abstraction with design pattern:-



Abstraction:-
Hiding internal implemetation & providing the functionalities to the user using the interface

Interface:-
It is the intermediate blw consumer & service provide

The output of abstraction is loose-coupling

Loose-coupling
Change in implementation which does-not effect the user is called loose-coupling

Tight coupling
Change in implementation which effects the user is called tight-coupling

Null-pointer Exception
Pointing towards an obj which is not present throws an exception called Null-pointer Exception

Code for Abstraction using Design pattern
(100% Abstraction code)

03/09/24
Tuesday

```
public interface Switch
{
    void SON();
    void SOFF();
}
```

Seprate file

```
//implementation logic
public class Tubelight implements Switch
{
    @Override
    public void SON()
    {
        s.o.p("Tube Light switch On");
    }
    @Override
    public void SOFF()
    {
        s.o.p("Tube Light switch off");
    }
}
```

Seprate cls file

```
//implementation logic
public class Ledlight implements Switch
{
    @Override
    public void SON()
    {
        s.o.p("Led light switch On");
    }
    @Override
    public void SOFF()
    {
        s.o.p("led light switch off");
    }
}
```

Seprate cls file

```
//Object creation logic
Public class Lightfactory
{
//Factory or helper method
Public static Switch getLight(String type)
  {
   if(type.equalsIgnoreCase("Ledlight"))
     {
      return new Ledlight();
     }
   elseif(type.equalsIgnoreCase("tubelight"))
      {
       return new  tubelight();
      }
   else
     {
      S.O.P("No such light found");
      return null;
     }
  }
}

import java.util.Scanner;
// Consumer or utilization logic
Public class Testlight {
 public static vM() {
   Scanner sc = new Scanner(System.in);
    S.O.P("enter the type of light");
    String input=sc.next();

    Switch sw = LightFactory.getLight(input);
    if(sw!=null)
     {
       sw.SON();
       sw.SOFF();
     }
  }
}
```

if we don't write sw!=null if statement then it throws NullPointerException when we take input as different light other then tubelight// Ledlight

O/P:-

1. enter the type of light
   → Ledlight

   Led light switch on
   Led light switch off

2. enter the type of light
   → tubelight

   tube light switch ON
   tubelight switch off

3. enter the type of light
   Magic light

   No such light found

4. enter the type of light
   Magic light
   No such light found
   throws NullPointerException (to avoid this in Testlight class if(sw! = null) condition has been return

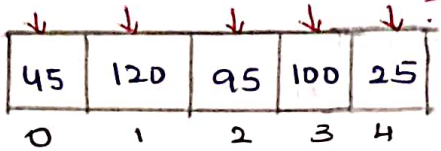
Arrays:-

- Array is a group/collection of similar type of da
- To store multiple values we use array.
- Array can be created in two different ways.

→ 01. `int a[] = {10,40,20,60};`

02. `int arr[] = new int[5];`

arr =



ArrayIndexOutOfBoo
where arr[5] inde
is not prsent

`arr[0] = 45;` `arr[2] = 95;` `arr[4] = 25;`

`arr[1] = 120;` `arr[3] = 100;` `arr[5] = 30;`

↓

Shows error -

- Array is fixed in size
- It stores homogenous data
- Memory is contegious(Element will be stored next to each other)
- array is index based.

01/09/24
Wednesd

Program:-

```
s.o.pln(" Enter the size");
int size = sc.nextInt();
int a[] = new int [size];
S.o.pln(" Enter elements into array: ");
for(int i=0; i<a.length ; i++){
    a[i]=sc.nextInt();
}
S.o.pln(" The array elements are:");
for(int i=0; i<a. length ; i++){
 s.o.pln(" a[i]);
}
```

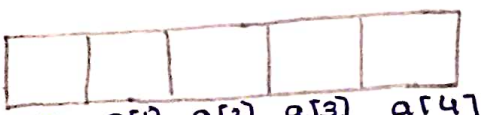
→ its a keyword

```
for(int num:a )
{
s.o.pln(num);
}
```

o/p:-

Enter the size

5



```
Public class Array
{
psvm(){
Scanner sc = new Scanner(System.in);
 int size = sc.nextInt();
 int a[] = new int[size];
 s.o.p("enter the elements: ");
 for(int i=0; i<a.length; i++)
   {
    a[i] = sc.nextInt();
   }
 Arrays.sort(a);
 s.o.pln(" the values of an array:");
 for(int i=0; i<a.length; i++)
   {
   s.o.pln(a[i]);
   }
}
```

NOTE:—

Can we store any array without giving size?

→ We can by using values

Can we store same types of obj in arrays

Arrays is class

→ yes,

```
Class Product
{
 int price = 200;
 String name = "Book";
}
public class ArraysObjects
{
 psvm(){
 Scanner sc = new Scanner(System.in);
  int size = sc.nextInt();
 Product p[] = new Product[size];
 for(int i=0; i<p.length; i++){
}   p[i] = new Product();
```

```
for(int i=0; i<p.length; i++)
    {
     s.o.pln(pri]);
     }
  }
}
```

Advantages of Arrays:-

- Efficient storage
- Can store multiple values

Disadvantages of Array:-

- We can store homogeneous elements only
- Memory wastage
- If we fix the size only that many elemen can be stored.
- No inbuilt methods we have to write logi
- Array does not supp Generic

Disadvantages of Array

- We can store only homogenous elements
- Memory wastage

05/09/24
Thursday

Frameworks:-

- It is an abstraction which contain generic functionalities which can be extended & used also customizable by tional user writen cod
- A framework may contain many inbuit class, interfaces & readymade utility methods.
- A framework is reusable.
- Mainly used for application development
- They are tested
- Uses best practices & neussary design patter
- Different frameworks are available for diff" kinds of application developments

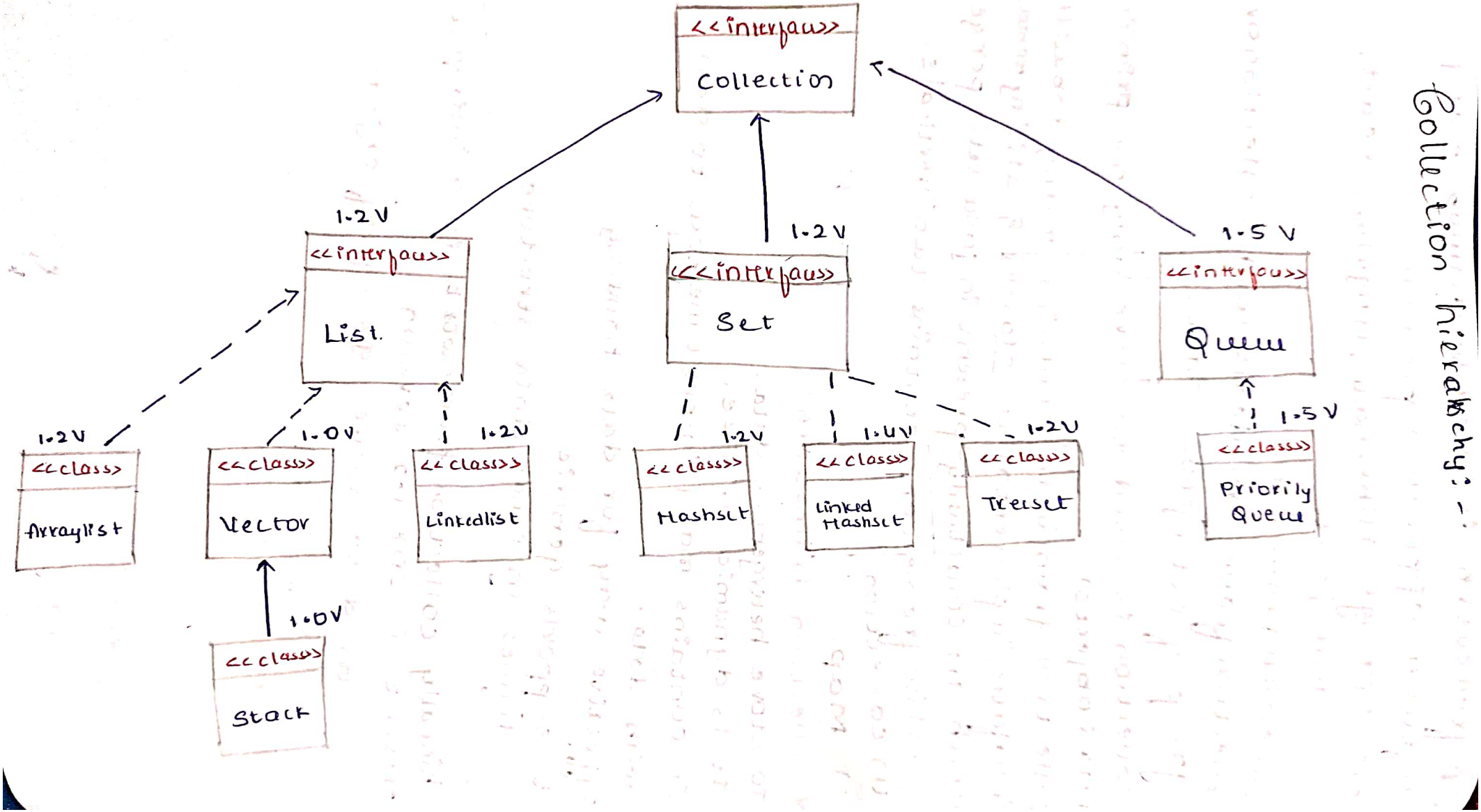
- A framework is used to develop many software application products
Ex:- springs, Hibernate, Angular, React, nodejs etc

Collection Framework:-
- Collection framework deals with collection or group of hetergenous data.
- Collection is used to store only non-primitive data (objects)
- Collection framework contains many inbuilt interfaces, implementation classes & utility methods which are collectively present in java.util pckge.
- Collection framework contains two verticals.
  (i) Collection
  (ii) Map.

Y collection framework?
→ To store hetergenous data.
→ It is dynamic in size,
→ It contains many in-built methods to deal with data.
→ Effective way for data handling
→ It supports generic.
→ It defines internal data structure.

NOTE:-
- Techincally collection framework is a interface present in JDK 1.2 version
→ It is used to store group of objects.

Collection hierarchy:-



- Collection is used to store non-primitive data.
- Collection is an interface which is introduced in 1.2 version present in java.util package
- since collection is an interface we cannot create an obj. but we can use it as a reference which refers to implementation obj.
- Collection interface contains many abstract().

To create collection Object:-

```
Collection c1 = new List();          ---  Both not
Collection c2 = new Collection();    ---  possible

Collection c1 = new HashSet();       ---
Collection c2 = new ArrayList();     ---  Possible
Collection c3 = new LinkedList();    ---
```

- Collection without Generic:-

```
import java.util.ArrayList;
import java.util.Collection;
public class CollectionDemo{
   psvm(){
Collection col = new ArrayList();
   col.add("Dinga");
   col.add("Gunda");
   col.add("Guldu");
   col.add(45);
   col.add(65.34);
   S.O.p(col);
  }
}
```

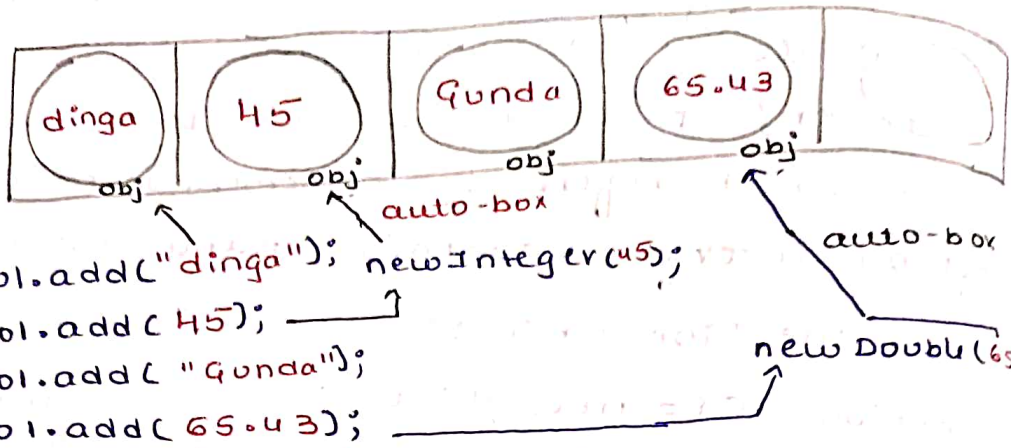
Olp:-

[Dinga, Gunda, Guldu, 45, 65.34]

- In the above example we are trying to store primitive value inside collection but in this case the primitive value will be converted into non-primitive obj (auto-box) & it stores

corresponding obj inside collection

Note:-
Collection never store primitive data.



Generic <E> :- → represents element type

- It is a feature in JDK 1.5 version.
- Generic is used to check the element typ
- Collection with Generic always store homogenous data.

Ex:-

```
List<Student>
Collection<String>
Set<Integer>
```

Collection with Generic (stores similar type of data)

```
import java.util.
import java.util.
Public class
    psvm(){
Collection<String> col = new ArrayList<>();
    col.add("Dinga");
    col.add("Gunda");
    col.add(45); //Error
    s.o.p(col);
}
}
```

Methods of Collection:-

```
int ref sizeC);
boolean int isEmpty();
boolean contains (object a);
object[] toArray();
boolean add(E e);
boolean remove(Object o);
boolean containsAll(Collection c);
boolean addAll(Collection c);
boolean removeAll(Collection c);
boolean retainAll(Collection c);
void clear();
Iterator <E> itrator();
```

Note:-

- Whenever the method parameter is super class then we have to pass the object of that class or the object of sub-class as a argument.
- Whenever the method parameter is interface than we have to pass implementation class object as a argument.

```
Public class CollectionDemo
{
 PSVM(){}
 Collection <String> cscol = new Arraylist<> ();
   cscol.add("Sunil");
   cscol.add("sammed");
   cscol.add("Shraddha");
   cscol.add("Sonal");
 List <String> eccol = new LinkedList<>();
   eccol.add("Subahan");
   eccol.add("Aryadeep");
   eccol.add("Anil");
```

```
Collection<String> mech = new Vector<>();
    mech.add("Ramesh");
    mech.add("suresh");

Collection<String> google = new ArrayList<>();
    google.addAll(cscol);
    google.addAll(ececol);
    google.addAll(mech);
    s.o.pln(google);
// [sonel, Sammed, Shraddha, soneshree, Aryadeep
    Anil, Ramesh, suresh]
    s.o.pln(google.size()); // 9
    s.o.pln(google.isEmpty()); // false
    s.o.pln(google.contains("Anil")); // True
    s.o.pln(google.remove("shradda")); // True
    s.o.pln(google.containsAll(cscol)); // false

        google.removeAll(mech);
        google.retainAll(escol);
        s.o.pln(google);
// [Sunil, Sammed, Sonashree] → Retains only es
    google.clear();
    s.o.pln(google);
// []
}
}
```

List:-

- List is an sub interface @ collection interface present in JDK 1.2V
- List is an index based
- List stores duplicate value.
- List stores any no of null values
- List maintains insertion order

List specific methods:- → object

- void add(int index, E element)
- boolean addAll(int index, collection col)
- E get(int index)
- int indexOf(Object o)
- int lastIndexOf(Object o)
- ListIterator <E> listIterator()
- ListIterator <E> listIterator(int index)
- E remove(int index)
- E set(int index, Elem E element)
- List <t> sublist(int fromIndex, int toIndex)

```
public class ListDemo
{
  PSVM(){
    List<String> lst = new ArrayList<>();
      lst.add("Dhoni");
      lst.add("Virat");
      lst.add("Rohit");
      lst.add("Rahul");
    S.O.p(lst);
  // [Dhoni, Virat, Rohit, Rahul]
    S.O.pln(lst.get(1));
    lst.add(1, "Gill");
    lst.set(4, "Pandya");
    S.O.pln(lst.sublist(0, 4));
  // [Dhoni, Gill, Virat, Rohit]
  }
}
```

A.L
I.B
maintorder
All duplicate
null
G.C
Marker(3)
3 const

Vector
I.B
M.O
All dup
null
1
4 const

LinkedList
Nodes
RandomAccess
NO I.C
2 constructor

Iterator
→ Not Index based
→ Uniqueness
→ hasNext
Next
Remove

ListIterator
→ Not I.B
→ Bi

HashSet
- Not I.B
→ Uniqueness
→ One null
→ Insertion may/may not
→ I.C 16
Ratio 75%
→ hashtable

LinkedSet
→ Maintain I.O
→ LinkedList + HashTable

ArrayList:- (Multi-thread, not synchronized, less secure, more performance)

- It is the implementation cls of list interface present since JDK 1.2V.

Arraylist inherits the all features of List interface

i) Index based
ii) Stores duplicate elements
iii) Stores null value.
iv) Maintains insertion orders

The initial capacity of arraylist is 10.
And the incremental capacity is

$$\text{Incremental capacity} = \left[\frac{\text{Initial capacity} * 3}{2} + 1\right]$$

$$= \frac{10*3}{2} + 1 = \frac{30}{2} + 1$$

$$= 15 + 1$$

$$= 16 \quad (10+6)$$

- There are 3 overloaded constructor for arraylist

(i) Public array list with zero parameter
(or)
Public ArrayList()

(ii) Public ArrayList(int Initial capacity)

(iii) Public Arraylist (Collection → c)

- ArrayList implements marker interface like Random Access, Cloneable, java.io. Serialisation or Serializable.

- Arraylist internally stores data in the form of array

- Arraylist maintains internal data structure called Growable or resizeable array.

Situation to use Arraylist:

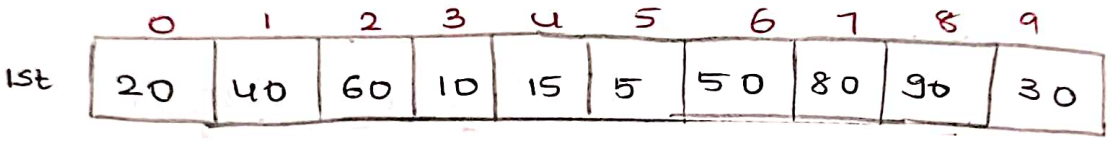
- Arraylist internally uses array hence the search and retrival operation is faster compared to any collection cls.
- It is best suitable for search & retrival ope

Situation to not prefer Arraylist

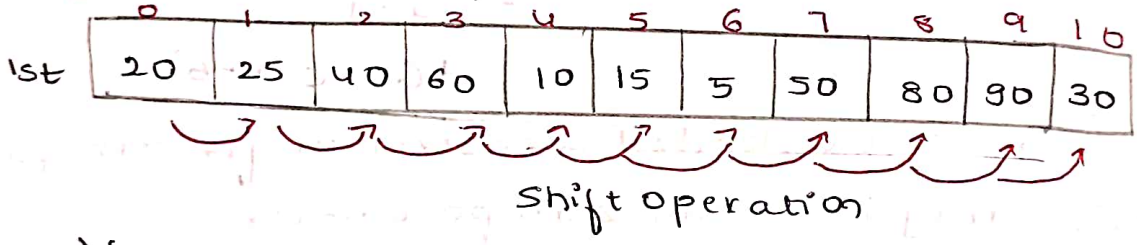
- If there is any frequent modification that is ting or removing the elements in blw then it's not a good choice to use Arraylist coz of shift operation.

Ex:-

```
List lst = new Arraylist();
```




```
lst.add(1,25);
```



Vector:-

- It is the implementation cls of a list interf present since 1.0V .JDK (majority cleses ar single threaded
- Vector is a legacy cls.
- Vector is a single-threaded cls.
- All the methods are synchronised in vector
- The initial capacity of vector is 10, & incremental capacity is double the initial capacity

10+10=20

If we add one elem[illegible] then also

- We can control the incremental capacity of the vector by using overloaded constru
- There are 4 overloaded constructor for the vector

(i) Public Vector()

(ii) Public Vector(int initial capacity)

(iii) Public Vector(int initial capacity, int capacity Increment)

(iv) Public vector(Collection c).

- Vector implements marker interface like Random Access, Cloneable, java.io.Serializable
- Vector internally maintains a data structure called Grow-able or resizable array
- Vector can be prefered when there is a search & retrival operations.
- Vector is not preferd when there is insert or removal operation cuz of shift operation [same as arraylist]
- A list can be ittrated in different ways

(i) for loop

(ii) for-each loop

(iii) using ittrator

(iv) Using List ittrator

more preferred

```
Public class ListDemo {
  Psvm() {
  List <String> lst = new ArrayList< > ();
    lst.add("Tejas");
    lst.add("Dilip");
    lst.add("Darshan");
    lst.add("Sthavan");
    lst.add("Shekar");
   s.o.pln(lst);
   s.o.pln("== iterate using for loop ==");
   for(int i=0; i<lst.size(); i++)
     {
    s.o.pln(list.get(i));
    }
   s.o.pln("== iterate using for-each loop==");
    for(String v:lst)
     {
     s.o.pln(v);
    }
  }
}
```

Olp:-

[Tejas, Dilip, Darshan, Sthavan, Shekar]

== iterate using for loop ==

Tejas
Dilip
Darshan
Sthavan
Shekar

== iterate using fore each loop ==

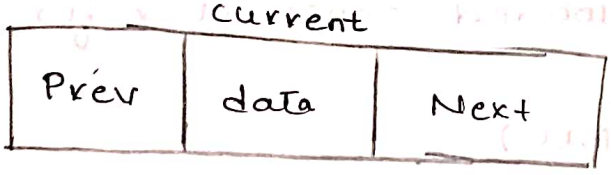
same

ArrayList	Vector
JDK 1.2 V	JDK 1.0V
Its incremental capacity can't be control	can be controlled
It has 3 overloaded constructor	4 overloaded constructor
Incremental capacity will be ted to the initial capacity	Incremental capacity will be double the initial capacity
Multi-threaded cls	Single-threaded cls
Non-Synchronised methods are faster in perfformance	Synchronised methods are slower in performance

LinkedList:- [Doubly LinkedList]

- It is a implementation class of list interface present since JDK 1.2V
- In case of linkedlist, the data will be stored in the form of Nodes.

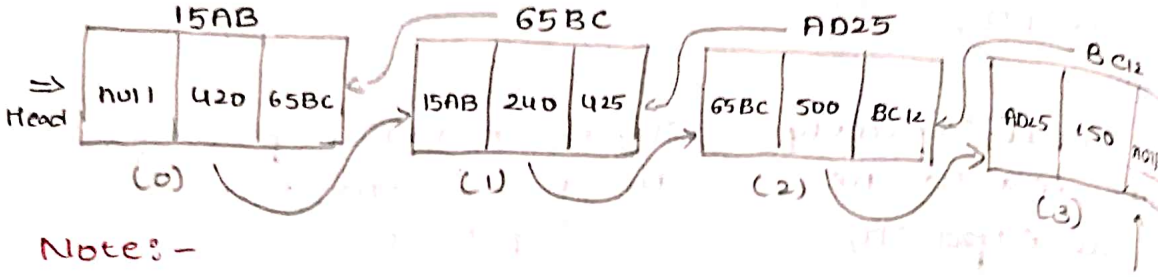
Node:-



- Each nodes store previous node address, next node address, current node address along with the data.
- The first node of LinkedList is called "Head"
- The last node of LinkedList is called as "Tail"

Ex:-

```
LinkedList<Integer> lst = new LinkedList();
    lst.add(420);
    lst.add(240);
    lst.add(500);
    lst.add(150);
```



Note:-

- In case of linkedlist the previous address of first node will be null.
- The last node next address will be null.
- LinkedList internally follows the data structure called Doubly LinkedList.
- In case LinkedList, there is no initial & incremental capacity.
- It implements marker interface such as serializa-ble & clonable but not RandomAccess
- The are 2 overloaded constructor for linkedlist

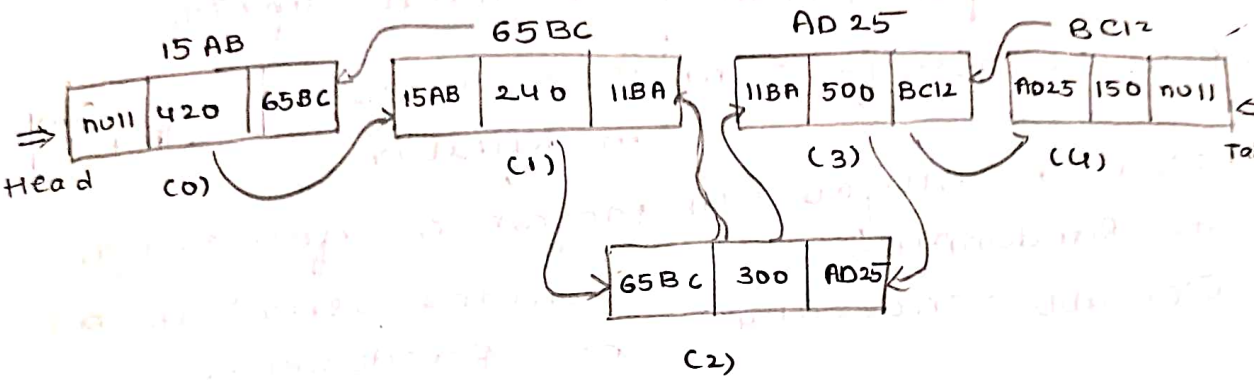
a] public LinkedList()

b] public LinkedList(Collection c)

Situation to prefer LinkedList ?

- Whenever we have frequent modification i,e adding element or removing element from the list then we have to prefer LinkedList

```
LinkedList<Integer> lst = new LinkedList();
    - - -
    - - -
lst.add(2,300);
```



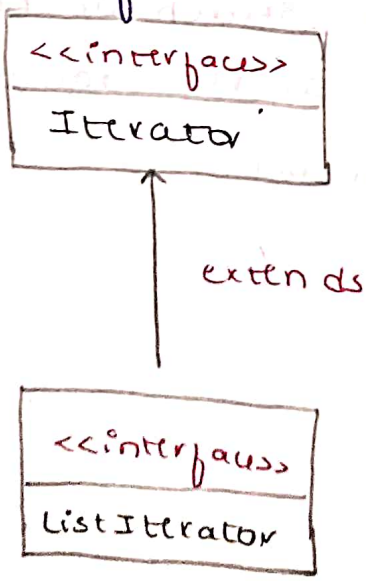
Situation not to prefer LinkedList

- Whenever we have search and retrival operation then it is not a good choice to prefer linkedlist
- Because, when we search or retrieve any elements then it starts tranversing from the head [1st node]

ArrayList	LinkedList
• Data will be stored in the form of Array	Data will be stored in the form of LinkedList
• Internally follows the "Growable or Resizable Array"	Internally follows the Doubly LinkedList
• Initial capacity is 10	No initial capacity
• Best suitable for search	Best suitable for frequ modification
• Marker interfaces such as RandomAcess, Cloneable, Serializable	Marker interfaces such as Cloneable, serializable b not RandomAcess

Iterator and ListIterator:-

- Technically, Iterator & ListIterator are interfaces in java.
- It is also known as cursors of Java.
- Iterator is super interface ListIterator is a sub interface of Iterator.



Iterator:-

- Iterator is not a index based
- It is used to iterate the entire collection
- It is uni-directional
- By using iterator, we can iterate only in forward direction

Methods of Iterator.

(i) boolean hasNext()

(ii) E next() E → objectlelement

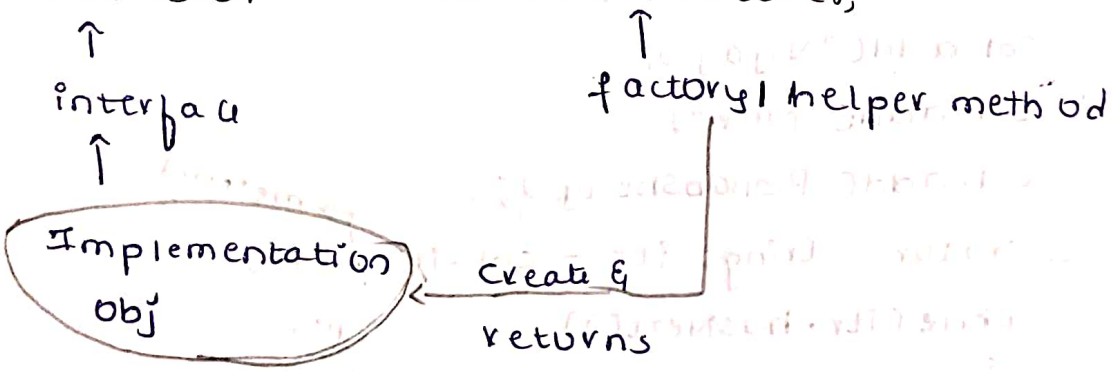
(iii) void remove()

To create implementation object of iterator:-

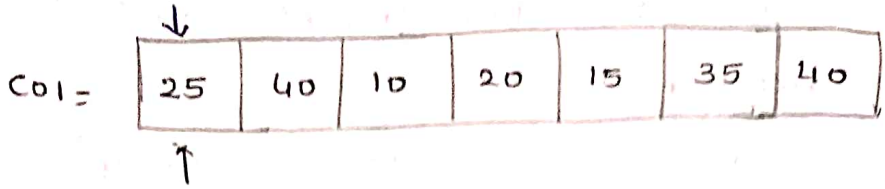
- We can't create object for iterator by using new keyword but we can get the obj by calling Iterator method on collection obj

Ex:-

```
Collection col = new ArrayList();
Iterator itr = col.iterator();
```



(true/false)
↳ itr.hasNext();



itr.next();

(Element)

```
while(itr.hasNext())
{
Integer elc = itr.next();
s.o.pln(elc);
}
```

```
Public class IteratorDemo
{
PSVM(){
Collection <String> col = new Arraylist<>();
col.add("Yash");
col.add("Puneeth");
col.add("Vijay");
col.add("Nter");
col.add("RishabShetty");
Iterator <String> itr = col.iterator();   ← method
while(itr.hasNext())
{
String elc = itr.next();
if(elc.equals("Vijay"))
{
itr.remove();
}
s.o.pln(elc);
}
s.o.pln(col);
}
```

olp:-
Yash
Puneeth
Vijay
Nter
RishabShetty

→ ['Yash', 'Puneeth', 'Nter', 'RishabShetty']

ListIterator:-

- It is the sub-interface iterator interface.
- It is an index based
- Since, it is an index based, it is used to iterate only lists.
- ListIterator is Bidirectional i.e we can iterate both in forward & reverse direction
- It to inherits all the methods of Iterator

ListIterator specific methods

(I) boolean hasPrevious();

(II) E previous();

(III) int nextIndex();

(IV) int previousIndex();

(V) void set(E e);

Ex:-

```
public class ListIteratorDemo
{
PSVM(){
List <String> col = new Vector<>();
  col.add("Aryadeep");
  col.add("Daivik");
  col.add("Amith");
  col.add("Karthik");
  col.add("Murali");
ListIterator <String> lst = col.listIterator();
s.o.pln("=== Iterate in forward direction ===");
while(lst.hasNext()){
  s.o.pln(lst.next());
}
```

```
S.O.pln("=== Iterate in reverse direction ===");
  while(lst.hasPrevious())
  {
  s.o.pln(lst.previous());
  }
}
```

Olp:-

```
=== Iterati in forward direction ===
Aryadeep
Daivik
Amith
Karthik
Murali

=== Iterate in Reverse direction ===
Murali
Karthik
Amith
Daivik
Aryadeep
```

Iterator	ListIterator
- Tranverse element in forward direction only [uni-directional]	Tranverse the elements in both directions [Bi-directional]
- Can be used with List & Set	- Can be used with list only
- Can perform remove operation while tranversing the elements	Can perform add, remove operation while tranversing the elements
- It is not index based	It is index based

16/9/24

Set:-

- It is the sub interface of collection interface present since jdk 1.2V.
- Set doesnt store duplicate value.
- The main purpose of set is to maintain uniqueness
- set is not an index based
- There are three implementation classes for set

```
Set s= new LinkedHashSet();
       HashSet();
       TreeSet();
```

- No objects can be created in the set as it is interface.
- Hashset & linked hashset is an hash based collection.
- Hash based collection:- Instead of storing obj directly it stores hashcode of an obj.
- Hash-code:- It is an integer representation of an object
- We can get hash-code of an object by using hashcode method.
- Hash-code method is present in super most class called object class
- Each & every object comes with unique hash-code.

Ex:-

```
public class Emp
    psvm(){
    Emp e1 = new Emp();
    s.o.pln(e1);          // Emp@7852e922
    s.o.pln(e1.hashCode()); // 2018699554
    s.o.pln("%x", e1.hashCode()); // 7852e922
}
```

Note:-

We can override hashcode based on hash code contract

- Since String cls & all the wrapper classes are overriden hashcode method based on hash code contract.

Hash-code contract:-

01. Two different object of same type with the same content must have same hash

Ex:-

```
psvm(){
    String s1 = new String("Guldu");
    String s2 = new String("Guldu");
    s.o.pln(s1.hashCode() + "==" + s2.hashCode());
}
```

o/p → 69162543==69162543

02. Two different object of same type with different content must have different hash-code.

Ex:- psvm()$

Double d1 = new Double(65.45);

Double d2 = new Double(24.24);

s.o.pln(d1.hashCode() + "!=" + d2.hashCode());

}

O/p:- 1935896755 != 815488311

Ex:-3. Two different object of different type must have different hashcode

→ psvm()$

String s = new String("Gunda");

StringBuffer sb = new StringBuffer("Gunda");

s.o.pln(s.hashcode() + "!=" + sb.hascode());

}

o/p:- 69164445 != 2018699554

Hash-Set:—

- It is the implementation cls of set interface present since JDK 1.2v
- Hash-set is an hash based collection
- where an hash code of an object will be stored
- Hash set inherits all the feature of set such as:-
- Maintain uniqueness, no duplicate values & not an indix based.
- Hash set may or may not follow insertion order
- Insertion order depends on hashcode of an object
- Hash set internally maintains a data-struct -ure called hash-table

- The initial capacity of hash set is 16.
- And the filled ratio/load factor is 0.75/75%.
- There are 4 overloaded constructor present for hashset

(i) hash set (); (OR) Hashset ();

(ii) Hashset (int initial capacity)

(iii) Hashset (int initial capacity; float load Factor)

(iv) Hashset (Collection c).

- Hashset stores only one null value
- Hashset implements marker interface like serializable & cloneable.

```
public class Demo{
    PSVM (){
Hashset <string> hs = new Hashset <> ();
    hs.add ("Virat");
    hs.add ("Dhoni");
    hs.add ("rohit");
    hs.add (null);
    hs.add ("vikat");
    hs.add (null);
    s.o.pln (hs);
}
```

⇒ [null, virat, dhoni, rohit]

* Here the hasset is not following the insertion order

Linked hash-set :-

- Implementation class of set interface present since jdk 1.4v
- It is an hash based collection
- Linked hash set maintains insertion order
- The interval data structure of linked hash set is hash table & linked list
- The increment capacity/ filled ratio/ load factor is 0.75
- The initial capacity is 16.
- Remaining features are same as a hashset.

```
import java.util.*;
public class Demo{
  Psvm(){
Linked Hashset<String> hs = new Linked Hash
                            set<>();
   hs.add("virat");
   hs.add("dhoni");
   hs.add("rohit");
   hs.add(null);
   S.o.pln(hs);
}
```

olp:- [virat, dhoni, rohit, null]

List	Set
- It is index based - It stores duplicate values - It follows insertion order	It is not index base It will not stor duplicate values . It may or may not follow insertio order

Hashset	Linked hashset
- It will not follow insertion order (or) may follow - It will maintain uniqueness • JDK 1.2v • Data structure is hash tabl	It will follow the insertion order It will not maintain uniquenss JDK 1.4V Data structure is hash tabl & linkedlist

Tree-Set:-

17/9/24

Tree

- Reset is an implementation cls of set present in JDK 1.2V.
- Tree-set is mainly used for sorting.
- Tree-set internally use comparable interface for sorting.
- Tree-set doesnot store heterogeneous data, it stores only homogeneous data.
- Tree-set stores only homogenous data coz the main purpose of Treeset is sorting internally it compares the element & sort the element, while comparing the element the element should be of same type if the element is different type then it cannot compare the elements.
- When we try to store heterogeneous data in Treeset it throws exception called Class Cast Exception.
- Tree-Set doesnot store even a single Null-value.
- When we try to store Null value inside Tree-set it throws an exception called Null-pointer exception.
- The internal data-structure of tree-set is Balanced-Tree.
- There is no initial & incremental capacity for Tree-set.

- There are 4 overloaded constructor present for Treeset

(i) Treeset()

(ii) Treeset(comparator ctr)

(iii) Treeset(Sorted set)

(iv) Treeset(collection anotherCol)

Treeset implements like Serializable & Clonable.

Ex:-

```
import java.util.Treeset;
public class TreeSetDemo {
   psvm(){
Treeset<String> ts = new Treeset<>();
  ts.add("Sudeep");
  ts.add("Darshan");
  ts.add("Yash");
  ts.add("Ganesh");
   s.o.pln(ts);
// [Darshan, Ganesh, Sudeep, Yash]
```

To print reverse

```
import java.util.Treeset;
  public class
Treeset<String> ts = new Treeset<>(Collections.
                                   reverseOrder());
  ts.add("Sudeep");
  ts.add("Darshan");
  ts.add("Yash");
  ts.add("Ganesh");
   s.o.pln(ts);
// [Yash, Sudeep, Ganesh, Darshan]
```

```
import java.util.Treeset;
Public class TreesetDemo{
 PSVM(){
 Treeset ts = new Treeset<>();
  ts.add("sudeep");
  ts.add("Darshan");
  ts.add("Yash");
  S.O.pln(ts);
 }
}
```

NOTE:-

- When we try to add user-defined inside Tree-set it throws an exception called ClassCastException.
- We get the above Exception coz the user defined data i,e class has to implement comparable interface.
- String class & all the wrapper-cls implements comparable interface hence it stores String Object & all the wrapper-cls obj.

```
Public class TreeSetDemo{
  PSVM(){
  Treeset<Customer> ts = new Treeset<>();
Customer c1=new Customer("shekar", 112, 9900887811);
Customer c2=new Customer("bilip", 113, 8800766551);
Customer c3 = new "Darshan", 114, 7788990);
  ts.add(c1);
  ts.add(c2);
  ts.add(c3);
  S.O.pln(ts);
 }
}
```

NOTE:-

-When we try to store user-defined data inside Set then it doesnot maintain any uniqueness. cuz in java each & every obj comes with unique hash-code hence, if the obj is having same content & same type then also the hashcode will be different (unique). hence it consider each & every obj as a unique object.

Ex:-

- To overcome this prblm we have to override equals & hashcode method.
  ↳ to check content

```
import java.util.Hashset;
  "   java.util.Set;
public class TestEmp{
   PSVM(){
   Emp e1 = new Emp("Dinga", 1120);
   Emp e2 = new Emp("Dinga", 1120);
   Emp e3 = new Emp("Gunda", 1121);
   Set<Emp> s = new HashSet<>();
     s.add(e1);
     s.add(e2);
     s.add(e3);
      S.O.pln(s);
```

→ Empl cls

[Dinga 1120, Dinga 1120, Gunda 1121]

```
Public clas Emp{
   String ename;
   int empid;
Public Emp(String ename, int empid)
    {
     this.ename = ename;
     this.empid = empid;
   }

  @Overrid
   public String toString(){
    return this.ename+" "+this.empid;
   }

@Override
  public int hashCode(){
  int val = empid + ename.hashCode();
   return val;
}

public boolean equals(Object obj){
  Emp e = (Emp) obj; → Downcast
    if (this == e)
       {
        return true;
       }
    else if(this.ename.equals(e.ename) &&
      this.empid == e.empid)
      {
        return true;
      }
    else
        return false;
    }
```

```
@Override
public int hashcode(){
int val = empid + ename.hashcode();
  return val;

}

import java.util.*;
  public class TestEmp{
    psvm(){
Emp e1 = new Emp("Dinga", 1120);
Emp e2 = new Emp("Dinga", 1120);
Emp e3 = new Emp("Gunda", 1121);
Set<Emp> s = new Hashset<>();
  s.add(e1);
  s.add(e2);
  s.add(e3);
   s.o.pln(s);
// [Gunda, 1121, Ding 1120]
```

18/09/24
Wednesday.

Sorting:-

Ex:-

```
  import java.util.*;
  public class sortDemo{
    Psvm(){
  List<string> lst = new Arraylist<>();
    lst.add("Dhoni");
    lst.add("Virat");
    lst.add("Rohit");
    lst.add("Rahul");
    Collections.sort(lst);
    sout(lst);
}
// [Dhoni, Rahul, Rohit, Virat]
```

```
PSVM() $
List<String> lst = new ArrayList<>();
    lst.add("Dhoni");
    lst.add("Virat");
    lst.add("Rohit");
    lst. add ("Rahul")
Collections.sort(lst, Collections.reverseOrder());
  s.o.pln(lst)
// [Virat, Rohit, Rahul, Dhoni]
```

output is * comparable

Collections:-

- It is a cls present in java.util pckg
- Collections class is mainly used for sorting the list
- Sort method internally use comparable for sorting.
- Collections.sort method internally uses ~~sort~~ Comparable interface for sorting.
- Public static void sort(List<T> list) → used only on list.
- Sort method used only on List

NOTE:-

- In all the wrapper classes & in string cls they have overridden compareTo method by implementing comparable interface hence String data & all wrapper cls data can be sorted by using Collections.sort and by using TreeSet.

Comparable Interface:-

- It is an functional interface present in java.util package.
- comparable is used for Sorting.
- comparable contains only one method.

amrd → public int compareTo(T o);

Ex:-

(I) comparable < String >
    + int compareTo (String o)

(II) comparable < Emp >
    + int compareTo (Emp e)

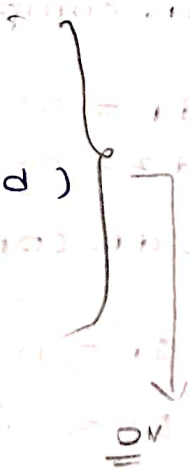
(III) Comparable < product >
    + int compareTo (product p)

NOTE:-
Return type of compareTo is Int.

Ex:-

```
Public class SortDemo{
  psvm(){
Integer i1 = new Integer(40);
Integer i2 = new Integer(30);
 s.o.pln(i1.compareTo(i2));   // 1

 Double d1 = new Double(55.25);
 Double d2 = new Double(100.01);
 s.o.pln(d1.compareTo(d2));   // -1

 String s1 = new String("Gunda");
 String s2 = new String("Gunda");
  s.o.pln(s1.compareTo(s2));   // 0
  }
}
```

```
Public class customer implements Comparable<
{
String cname;
int cid;
long contact;
Public customer(String cname, int cid,
        long contact)
{
this.cname = cname;
this.cid = cid;
this.contact = contact;
}
@Override
public String toString() {
return this.cname + " " + this.cid + " "
        + this.contact;
}
@Override.
if(+t
public int compareTo(Customer o){
if(this.cid > o.cid)
    return 1;
else if (this.cid < o.cid)
    return -1;
else
    return 0;
}
```



```
@Override
Public int compareTo(Co
{
Integer i = this.cid;
return i.compareTo(o.c
}
```

```
public class TreeSetDemo {
  psvm() {
Treeset<Customer> ts = new Treeset<>();
Customer c1 = new Customer("Shekar",
        112, 9900881166L);
Customer c2 = new Customer("Dilip",
        115, 8800776655L);
Customer c3 = new Customer("Darshan",
        111, 7788990066L);
    ts.add(c2);
    ts.add(c3);
    ts.add(c1);
   s.o.pln(ts);
   }
}
```

Olp:-

```
[Darshan  111  7788990066,
 Shekar   112  9900881166,
 Dilip    115  8800776655]
```

- To maintain sorted order based on string

```
@Overvride
Public int compareTo(Customer emp o) {
  String s = this.ename;
  return s.compareTo(o.ename);
  }
```

To maintain sorting in descending order:-

Change in

```
@Override
public int compareTo(Costomer o){
    if(this.cid > o.cid)
        return -1;
    else if(this.cid < o.cid)
        return 1;
    else
        return 0;
}
```

OR

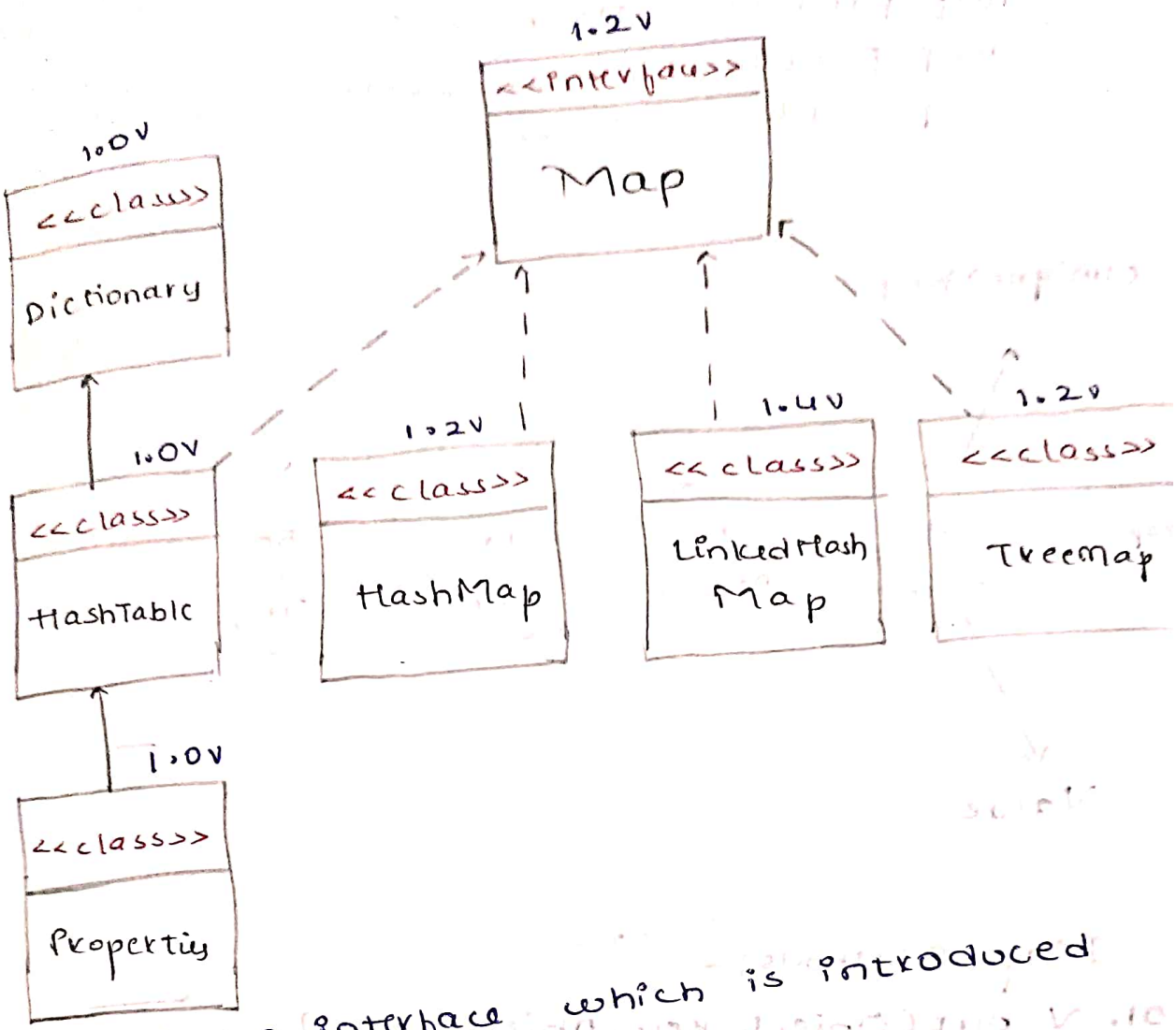
```
@override
public int compareTo(Costomer o){
    Integer i = this.cid;
    return i.compareTo(o.cid)*-1;
```

Olp:-

```
[Dilip    115   8800776655,
 Shekar   112   9900881166,
 Darshan  111   7788990066]
```

Map: —

Thursday



- Map is an interface which is introduced in the version 1.2V.
- Map is present in java.util package
- Map is a seprate vertical of collection Framework.
- Map is a collection of Entries.

→ Entries

- It is a data in the form of key & value pairs.
- In case of Entries key must be always unique values can be anything

NOTE:-

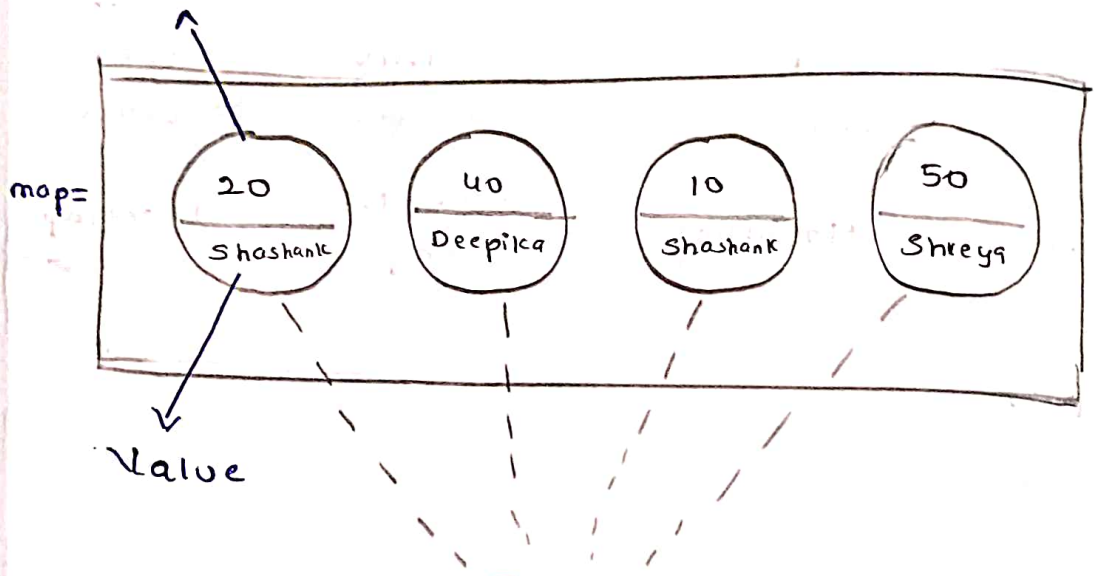
Map doesnot share the methods of Collection interface. [Map doesnot inherit collections
collection " " Map]

```
Map<Integer, String> map = new HashMap<>();
map.put(20, "Shashank");
map.put(40, "Deepika");
map.put(10, "Shashank");
map.put(50, "Shreya");
```

Value can be same

↓ (Key's must be unique)

(unique) Key



Map Methods:-

01. V put(Object key, Object value)
02. void putAll(Map map)
03. V remove(Object key)
04. boolean remove(Object key, Object value)
05. boolean containsValue(Object value)
06. boolean containsKey(Object key)
07. V get(Object key)
08. boolean isEmpty()
09. void clear()
10. Collection values()
11. Set keySet()
12. Set<Map.Entry<K, V>> entrySet()

Imp

Interface Interface nested interface

map =

20	40	10	50
shashank	Deepika	shashank	Shreya

map.values();

Collection → | shashank | Deepika | Shashank | Shreya |

Here the values of map is stored in collection cuz Map can store both key and value, not only value so it is stored in collection

map.keyset();

set → | 20 | 40 | 10 | 50 |

Here the key is stored in set cuz key must be unique so it cannot store in collection & map. hence cuz set is unique key is stored in set.

→ Public interface Map<K,V>
{
interface Entry<K,V> {
K getKey();
V getValue();
V setValue();
}
}

Entry is nested interface which is present in Map interface

Entry can store data in the form of key and value.

- Map cannot be iterated directly so the written type is set in

  Set<Map.Entry<K,V>> entrySet()

  OR

EntrySet returns data in form of key & value which can be stored inside map.Entry & converted into Set.

The written type of Entry Set is Set interface coz the map cannot be iterate directly so hence we convert it into Set and we perform itteration.

```
import java.util.Hashmap;
import java.util.Map;
import java.util.Set;

public class MapDemo{
  PSVM(){}
```

HashMap:-

- It is a implementation class of map interface present in JDK 1.2 V.
- HashMap is an Hash based collection.
- In case of map maintaning uniqueness, sorting, Searching hashcode etc is based on key but not based on value.
- The operations like searching, sorting insertion order depend on key but not value
- Hashmap internally uses Data structure called hash table.
- Hashmap may or may not follow an insertion order.
- The initial capacity of hasmap is 16 & Load factor is 0.75
- It implements marker interface like serializable & Clonable.

- It stores only one null key.
- There are 4 overloaded constructor for HashMap

(I) Hashmap()

(II) Hashmap(int initial capacity)

(III) Hashmap(int initial capacity, float load factor)

(IV) Hashmap(ColMap m)

Linked-HashMap:—

20/09/24 Friday

- It is an implementation cls of map interface present since JDK 1.4V.
- Linked hash map follows an insertion orde
- The internal data-structure of linked hashmap is Hashtable & Linked list.

(Everything is same as map):

TreeMap:—

- Implementation cls of Map interface present since JDK 1.2V.
- Tree-map is mainly used for sorting & the storting takes place based on key.
- Tree-map doesnot store even a single Null-key.
- Tree-map stores homogenous data (key).
- The internal data-structure is Balanced Tree.

Ex:- import java.util.TreeMap;

```
public class TreeDemo{
psvm(){
TreeMap<Integer, String> m = new Treemap<>();
m.put(10, "Dinga");
m.put(30, "Gunda");
m.put(5, "Guldu");
m.put(8, "Dingi");
s.o.pln(m);
}
}
```

{5 = Guldu, 8 = Dingi, 10 = Dinga, 30 = Gunda}

Iteration of Map:-

```
public class MapDemo1{
psvm(){
HashMap<Integer, String> m = new HashMap<>();
m.put(2, "Dilip");
m.put(5, "Shekhar");
m.put(1, "Darshan");
m.put(3, "Suresh");
Set<Integer> s = m.keySet();
for(Intger key: s){
s.o.pln(key + "=" + m.get(key));
}
s.o.pln("=== using ittrator ===");
Set<Map.Entry<Integer, String>> es = m.entrySet();
Iterator<Map.Entry<Integer, String>> itr = es.iter
                                          -ator();
while(itr.hasNext())
{
s.o.pln(itr.next());
}
}
```

Exception:-

21/09/24
Saturday.

- An unwanted or unexpected event which distrub the normal flow of the program is called Exception.
- It is a Run-time interption which makes the program to stop from execution.
- Exception occurs only during Run-time and when it occurs the program gets terminated
- Exception will never occurs @ compile-Time
- Exception is always thrown by JVM.

Exception hierachy:-

NOTE:-

Exception can be handled Error's can't be handled

```
a = a+b;
b = a-b;
a = a-b;
```

```
a = a^b;
b = a^b;
a = a^b;
```

```
temp=a
a=b
b=c
```

```
Class fs
psvm
{
int a=0.b=1;
for(i=1; i<=n; i++){
    sout(a);
    int c=a+b;
      a=b
      b=c;
    }
    sout(b)
```

```
Amst
int num=137;
int n=num;
for(int i<1;;ik
  while(n>0)
    {
rem = n%.10;
result+= math.pow(rem,3);
  n=n/10;
}
sout(result)
```

```
int count=0
for(i=1; i<=n;i++){
  if(n%.i==0)
    {
      count++
    }
  if(count==2)
{
  fact=1;
  for(
    f=f*i;
  }
  sout(fact;
```

```
while(n>0)
{
for
  rem= n%.10;
  rev= rev*10+rem;
  n=n/10;
  sout(rev);
```

```
a+b = a^b
a-b = a^b
a-b = a^b.
```

```
steps=3
a=b
b=c
```

```
for(i=0;i<=10;i++){
c=a+b
a=b
b=c;
}
```

```
num=371
int n=num;
for while(n>0)
{
rem=num%10;
rev=math.pow(rem,3)
n=n/10;
}
sout(rev);
```

```
while(num!=0)
{
rem=n%10;
rev=rev*10+rem
n=n/10;
}
```

```
for(i=0;i<=n;i++){
if(n%i==0)
{
co++;
}
if(co==2)
```

```
for(i=1;i<=n;i++)
{
fact=0;
for(j=1;j<=i;j++)
{
fact++;
}
if(f==2)
{
co
}
}
```

```
fact=1
for(i=1;i<=n;i++)
{
f=fact*i
pf(fact)
```

o/p = 101

```
int a = 101;
int b = 010;
int res = a+b;
sop(res);
```

o/p = ∞

```
by a=1
while(a<30)
sop(a)
a++;
```

```
console.log('017'==015)  → false
          ('018'==018)  → true
```

15/07/24
Monday

Operators.

1. Arithmetic → +, -, x, %, /
2. Logical → &&, ||
3. Relational → >, <, <=, >=, ==, !=
4. unary → inc, dec
5. Ternary → : ?
6. Assignment
7. Bitwise
8. Shift.

Inc
- Post: a++
- Pre: ++a

Dec
- Postdec: a--
- Predec: --a

Ex:- a = 99

```
sout(--a);   // 98
sout(a--);   // 98 → Prith then dec 97
sout(a++);   // 97 → print then inc 98
sout(++a);   // 99.
```

Greater of 3 no:-

```
if(a>b){
 if(a>c){
   sout(a)
  }
 else{
     sout(c);
   }
 else{
     if(b>c)
       {
         sout(b);
      }
     else{
         sout(c);
   }  }
```

(condition) ? (Exp1) : (Exp2);

~~int m = (a>b) ? (a>c)) ? a : c~~

```
int m = (a>b) ? ((a>c) ? a : c)
              : ((b>c) ? b : c);
    }
    sout(m);
}
```

```
if(a<b)
 {
  if(a<c)
   {
    sout(a)
   }
  else {
    sout(c)
   }
else
   if(b<c){
     sout(b);
   }
   else
     sout(c)
   }
}
```

```
int s = (a<b) ? (a<c) ? a : c)
              : (b<c) ? b : c);
 }
 sout(s);
 }
}
```

Tuesday
16/07/24

Different btwn if & else if else, nested if

```
if( )
if( )
if( )
```

```
int per=90;
 if( per >=90)
 {
   sout("Dest");
 }
 if(per >=60) {
   sout(" Firstcls");
 }
 if(per >= 50) {
   sout(" Second cls");
 }
```

```
if(per<= 35)
  }
  sout("fail");
  4
```

o/p:- Dest
first cls
second cls
fail

```
if(per >= 85){
sout("Dest");
  4
else if(per >= 60){
  sout("firstcls");
else if( per >= 40){
    sout("second cls");
  else
    sout("fail");
```

```
int per=85
//output
   Dest
 int per 40
//   second cls
```

Switch:-

```
int day=4;
Switch (day){
case1;
sout('its monday');
break;
Case2:
 sout('its Tuesday');
break;
Case3;
  Sout(its wed');
break;
Case4:
  sout('its Thursday');
break;
Case5:
  sout('its friday');
default:
  sout('no day');
break;
```

Note:-
If we didnt write break statement & try to execute
- If input is 4
Output:
Thursday
friday.

If we use break statement & take input as 4.
Output:
Thursday

```
byte i=1;
while(i<=300){
    sout(i);
    i++;
}
}
```

→ It will print ∞ loop from byte range i.e -128 to 127 (bcoz neg & pos value print)

-128+1 = -127

-127+1 = -126

⋮

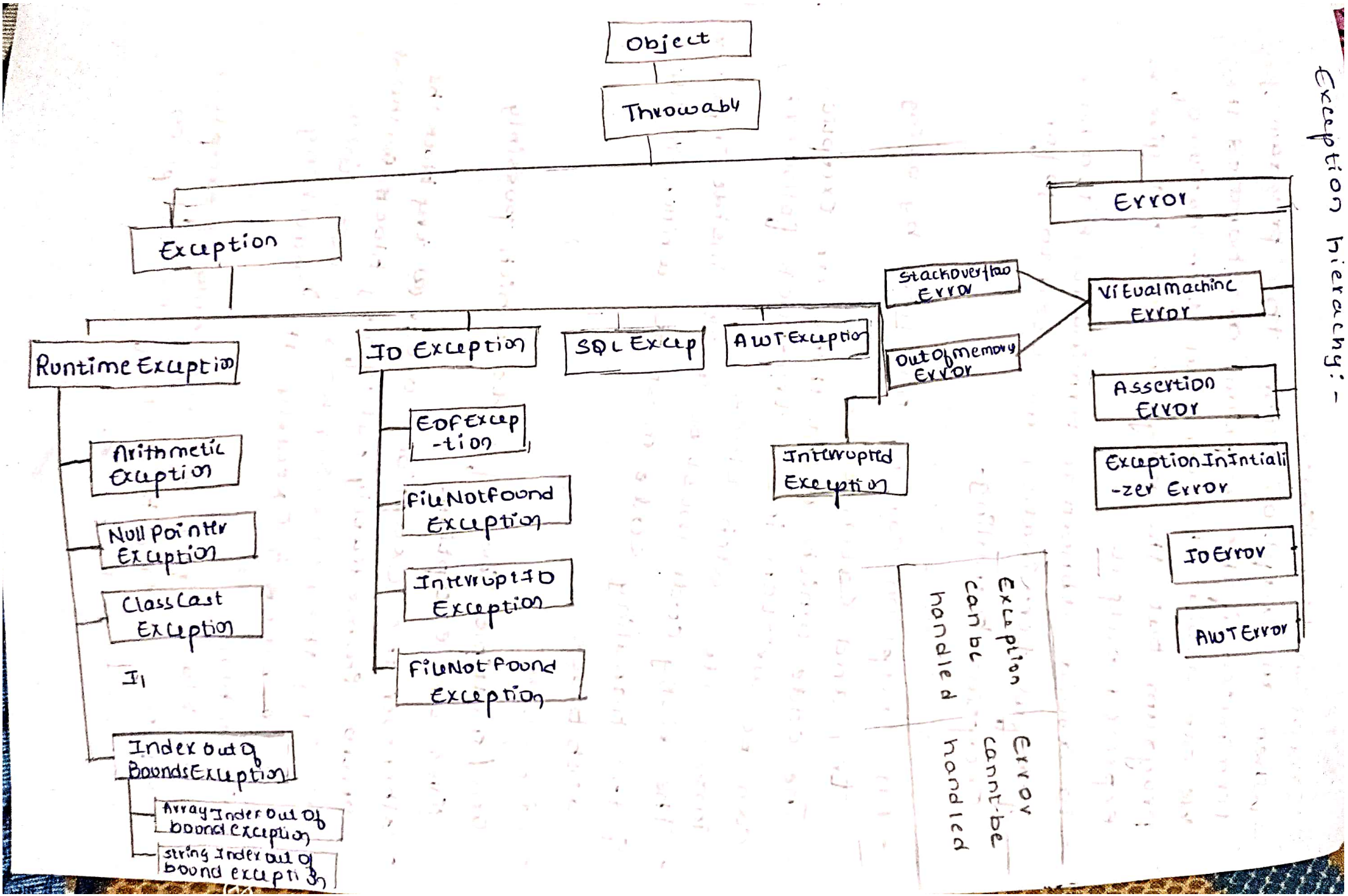
It become +ve

whe -1+1 = 0

0+1 = 1

1+1 = 2

Exception hierarchy:-



NOTE:-

Exception Handling:-

- When an Exception occurs the program gets terminated if the exception's are handled then we can avoid the abnormal termination of program but the program may not give the expected output.
- To avoid abnormal termination & to achieve graceful termination we used to go for Exception handling.

How to handle Exception?

- An exception can be handled by using try and catch block.
- The code which might throw an Exception can be written inside try-block followed by immediate catch-block.
- The risky code must be written inside Try block & the handling code must be written inside Catch block.

→ Risky code:-

- One/multiple lines of code which possibly gives an exception.
- Whenever the Exceptions occur @ try block the control immediately out of try block without executing the remaining lines of code within the try block & Executes the matching catch block.
- Catch block gets executed only if exception happens in try block & if the exception matches catch block.

Ex:-
01.

Risky code
- Math.Operation...
- File.Operation .... <— FileNotFoundException
- Data Base Operation ... x
- String. Operation x

02.

```
try {
    Math.Operation ---
    File.Operation ---<— FNFE
    Data Base Operation --- x
    String Operation --- x
}
Catch(Exception e)
{
    Handling code---
}
```

will not execute coz file.operation has FNFException

- When exception occurs in try block control comes out from try block & search for matching catch block if match block is not found then program gets terminated (Abnormal termination)
- Once control comes out of try block it will never goes back to try block.

Ex03:-
```
try {                    → If there is any Exception in try block then next line of code will never execute
    1. Hii...
    2. Helo... <— Exception
    3. Welcme... x
    4. to PentagonSpace... x
}
Catch(Exception e)
{
    5. sorry you get Exception
}
```

Never get execute

NOTE:-
To execute next line of code after exception had occured we have to use Nested try & catch block

Olp:-

Hi---
Hello---
Sorry you got Exception

Ex:- 04. No error/Exception in try block then catch block wont Execute

```
try
{
01. Hi---
02. Hello----
03. Welcome---
04. to pentagonSpace---
}
Catch(Exception e
{
5. sorry you got Exception
}
```

Olp:- Hi---
Hello---
Welcome---
to pentagon Space.---

NOTE:-

** There must be not be any executable code in blw Try & catch

- Exception means JVM creates object of appropriate Exception cls. & that obj is thrown @ run-time.
- The reference type used in a catch block must belong to Exception catagory i,e it must be directly or indirectly a sub-cls of Throwable Class.

Monday

```
import java.util.Scanner;
public class Demo1 {
  PSVM(){
  Scanner sc = new Scanner(System.in);
    s.o.pln(" Enter value of a");
  int a = sc.nextInt();
  s.o.pln(" Enter value of b");
  int b= sc.nextInt();
  try
  {
int c= a/b;
    s.o.pln(c)
  }
  Catch
  {
  s.o.pln(" Exception handled");
  }
  s.o.pln("main method executed normally");
  }
}
```

Arithmetic Exception (obj)

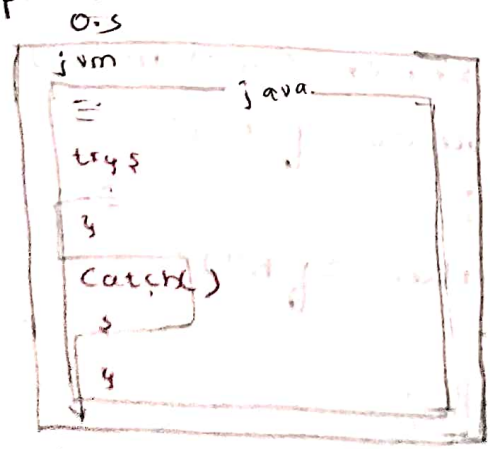
Mistake information

1000 (address)

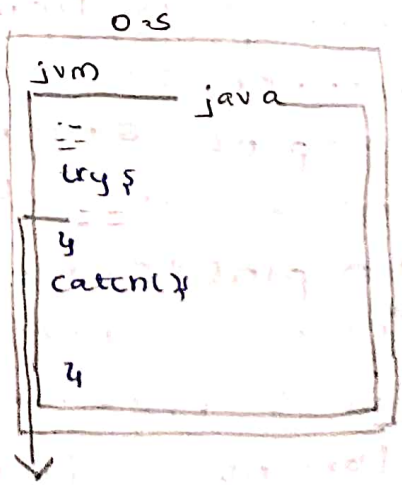
10
0

- Sometymes try block

Exception:-

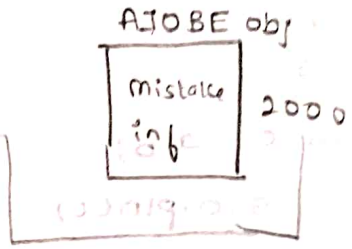


Error:-



ArrayOutOfBound Exception

```
Class Demo{
 psvm(){
 int [] arr = new int[5];
  arr[0]=1;     arr[3] = 4;
  arr[1]=2;     arr[4] = 5;
  arr[2] =3;
 try{
 s.o.pln(arr[10];
 }
Catch(AIOBE pentagon)
  {
  s.o.pln("Exception handled");
  }
}
S.O.pln();
}
```



Nested try - Catch: —

```
import java.util.Scanner;
public class Demo3 {
        psvm(){
Scanner sc = new Scanner(System.in);
 s.o.pln(" Enter the value of a");
 int a = sc.nextInt();
 s.o.pln("Enter the value of b");
 int b = sc.nextInt();
 int[] arr = new int[5];
 for(int i=0;i<5;i++)
  {
   arr[i] =i;
 }
```

```
try{
    int c = a/b;
    s.o.println("a/b gives:" + c);
    s.o.println("====");

  try{
    for(int i=0; i<10; i++)
    {
     s.o.println(arr[i]);
    }
  }
  Catch(ArrayIndexOutofBounds Exception e)
    {
     s.o.println("catch1 is executed");
    }

}
    catch(Arithmetic Exception e)
    {
     s.o.println("catch 2 is executed");

    }
    s.o.println("Execute normally");
}

}
```

o/p:-

Enter the value of a

10

Enter the value of b

2

a/b gives: 5

====

0

1

2

3

4

Catch1 is executed

Execute normally

24/09/24
Tuesday

Multiple-Catch blocks:—

```
try {
    DB operation.
    maths operation.
    sql oper.
}
Catch (SqlException e)
{
}
Catch(ArithmeticException e)
{
}
Catch(Exception e)
{
}
```

sub-cls

sub-cls

super class

NOTE:-

- Order of exception should be from sub-cls to super-cls.
- If we are gng to maintain oraer of multiple from super-cls to sub-cls then we will result in unreachable code

↳ Ex:-

```
try {
    DB
    sql
    math
}
Catch(Exception, e) {
}
catch(AI. e)
{
}
```

super-cl,

sub-cls

→ Error saying unreable code

```
public class Demo
 {
  psvm()
  {
  int[] arr = new int[5];
  for(int i=0; i<5; i++)
    {
     arr[i]=i;
    }
  int a,b,c;
   a=10;
   b=5;
try{
    c=a/b;
    s.o.pln(arr[i]);
  }
catch(AE|OBE e)
  {
   s.o.pln("Array index out of bound Exception");
  }
catch(AE ae)
  {
  s.o.pln("Arithmetic exp");
  }
cath(Exception e)
   {
    s.o.pln("throws an exception");
   }
 }
  s.o.pln("main method executed");
  }
}
```

o/p:- Arithmetic exp
main method excuted

if a=10
b=5
o/p:- Array index out of bound Exception
main method executed

finally :-

```
try {
    validate card operation...
    validate operation
    validate withdraw Amount...
    Deposit...
    }
catch(Exception e){
    }
finally{
    return Atm card;
    }
```

Q:- can we write try without catch finally?

```
finally {
    try {
    }
    catched()
    {
    }
}
```

→ Yes, possible

- finally block is used to exception handling
- It is always gets excepted irrespective of exception occurs or not.
- If any costly resources are present inside the prgm then we need to close it mandatorily so to close the costly resource we use finally block.
- We can have try block & finally block without catch block,

Q:- Can we write single try block

→ Yes, we can write but we have to use finally.

```
try {
    ==
    }
finally{
    ==
}
```

Q:- only finaly block can we write

→ Not possible

```
finally{
}
```

Types of Exception:—

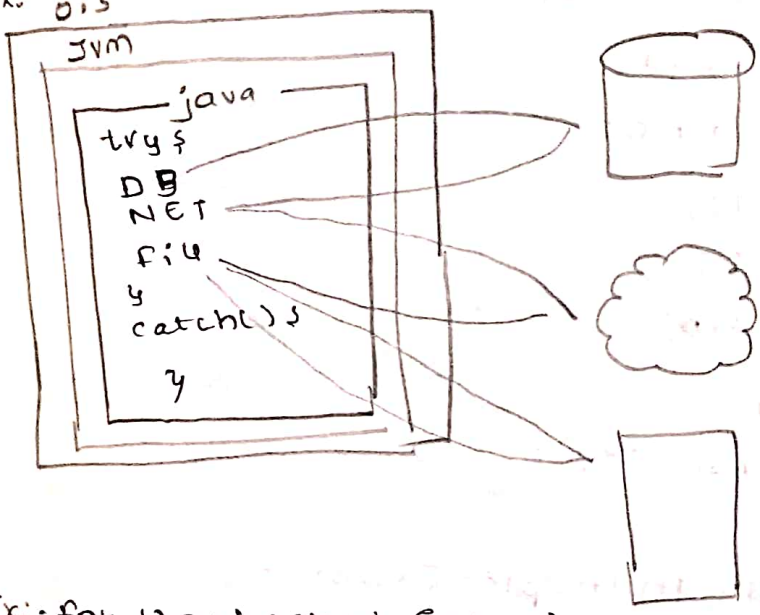
(i) Checked - Exception
(ii) Un-checked - Exception

(i) Checked Exception:- [It occurs @ run-time but compiler will worry about the exception & say to use try & catch]
In checked exception compiler will force for the developer to handle the exception.
Ex:- SQL Exception, AWT Exception, FileNotFound Exception, ClassCast Exception, Interrupted Exception etc

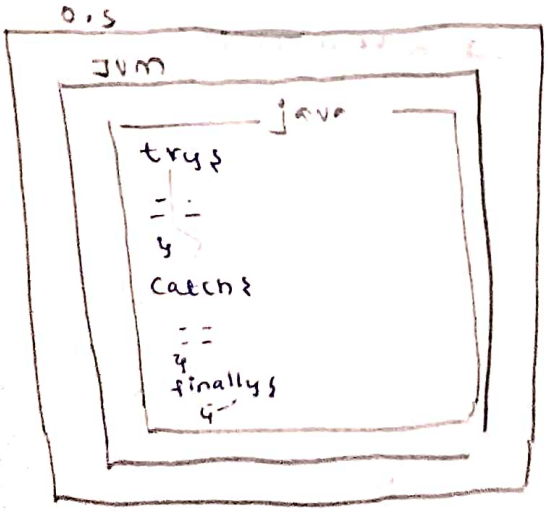
(ii) Un-checked exception: Here compiler will not force the developer to check the exception @ runtime JVM will handle it
Ex:- Arithmetic Exception, AIOBE, Nullpointer Exception, String Index Out of Bound Exception etc.

- It occurs @ Run-time.

Ex:- for Checked exception



Ex:- for Unchecked Exception



Example for Checked Exception.

```
public class Demo;
    Psvm();
  Int a.b.c;
   a=10;
   b=5;
   c=a/b
   s.o.pln(c);
  Thread.sleep(2000);
}
```

→ At this point we will get compiler will force us to use try & catch block.

so. we have to use try & catch block

```
public class Demo;
     Psvm();
     Int a.b.c;
      a=10;
      b=5;
      c=a/b;
    try;
    Thread.sleep(2000);
     }
    Catch (InterruptedException e)
      {
       e.printStackTrace();
    }
}
```

Throwing Exception Manually:-

26/07/24 Thursday.

Throw:- (is used to throw exception Manually)

```
class Demo{
  psvm(){
scanner sc = new Scanner(System.in);
s.o.pln("Enter value of a");
int a = sc.nextInt();
s.o.pln("Enter value of b");
 int b = sc.nextInt();
 try{
  if(a>b){
  int c = a-b
  }
  else{
  throw new Exception();
   }
  }
   catch(Exception e){
    s.o.pln(" A lesser then B");
    }
  }
}
```

→ or else throw → Exception object

e.printStackTrace()
↓
To know where exception has accured

Olp:-

10

20

a less then B

Ex1:-

```
Class Throwww Demo{
    Psvm();
    int a=sc.newInt();
    int b= sc.newInt();
    try{
      if(a>b){                    → 18th Line
        c=a-b;
        s.o.pln(c);
      }
    else{
      throw new AlessthenBException();
    }
  }
  catch(Exception e)
    {
    e.printStackTrace(); // shows in which line exception is occured
    s.o.pln                points to
    }
  }
}
Class AlessthenBException extends Exception{
    AlessthenBException()
      {
      s.o.pln("Aless then B");
      }
  }
```

here customer obj we are creating we must have obj class for it

we have to extend coz it is custom so we have to say Exception cls then it is also custom Exception

→ a = 10
b = 20

A less then B

at throw.main(Throww.java:18)

@ 18th line we have Exception

Error:- When try & catch cannot handle the Exception then it is called Error

Ex:-

```
public cls Demo(){
  psvm(){
try{int [] arr = new int[99999999];
  }
catch(AIOBE e)
  {
  s.o.pln("Array index out of bound");
  }
}
s.o.pln("main method executed");
}
}
```

Olp:- Main method executed.

Ex2:-

```
class RTOApplication{
  psvm(){
Scanner sc=new Scanner(System.in);
s.o.pln("Enter ur Age");
  int Age = sc.nextInt();
try{
if(Age<18 || Age>60){
  throw new InvalidAgeException();
  }
else{
  s.o.pln("DL is issued");
  }
}
Catch(InvalidAgeException e)
  {
  s.o.pln(e.getMessage());
  }
}
}
```

```
class InvalidException extends Exception
  {
  @override
  public String getmessage();
    return "age is not valid";
    }
  }
```

Olp:-

```
Enter the age
  18
→ Issue the DL
→ DL is issued

Enter the age
  12
   Age is not valid
```

Throws:-

```
public class Throws{
  PSVM() throws AE, AIOBE {
    int [] arr = new int[5];
    for(int i=0; i<5; i++)
      {
       arr[i]=i;
      }
    int a,b,c;
     a=10;
     b=0;
    try
    {
      c= a/b;
      S.O.pln(arr[0]);
    }
   finally{
     S.O.pln("main method executed");
  }}
```

Olp:-

```
main method executed
Then
java.lang.ArithemticException
  /by zero
    @ throws.main
      (Throws.java:14)
```

→ first execute whole Pgm then throw Error

Multi-Tasking:- 20/09/24 Friday

Performing several task simultanously is called Multi-Tasking.

There are 2 types of Multi-tasking

(i) Process-based Multi-Tasking

(ii) Thread based Multi-Tasking

Process-based Multi-Tasking

- Performing serial task simultanously where easy task is a seprate independent process
- Process based Multi-tasking is best suitable for O.S level

Ex:- Listening to Music & browse the internet @ the same-time.

Thread-based Multi-Tasking (Multi-Threading)

- Performing several task simultanously where each task is a seprate independent part of same program.
- When we need to excute 2 or more prgm simultanously then go for Thread-based MT.
- Thread-based MT is best suitable for Programtic-level

Ex:- Using browser we can navigate throw web Pge @ same tym we can download a file.

Multi-Threading:-

- It is a process of execuling multiple threads simultanously.
- MThreading is mostly used in gaming, Animations & to develop servlets server etc

Thread:-

- Thread is a light weight sub-process & it is a smallest unit of processing
- Multi-processing & multi-threading both are used achieve Multi-Tasking.
- Threads are seprate flow of Execution.
- Each threads are independent.
- If the exception occurs in one thread does not affect another thread.
- Each threads are excuted in seprate run-time Stack memory.

Multi-processing:-

- In case of Multi-processing each process is associal allocated with seprate memory area
- A process is heavy weight.
- Cost of communication btw the process is high.
- Context switching takes more time i.e switching from one process to another process require some-time for saving, loading register's, memory map's & updating list etc.
- There are two-ways to create thread in java.
  - (i) By extending thread cls

    class Demo extends Thread

  (ii) by implementing Runnable interface

    class Demo ex implements Runnable

Ex:- 1st way of creating Thread-obj

```
class MyThread extends Thread
{
  @Override
   public void run()
   {
   s.o.p(" Child - Thread Running");
   }
  Public static void main(String [] arg)
  {
  MyThread t1 = new MyThread();
  t1.start();
   s.o.p(" Main - Thread Running");
  }
}
```

t1 (child - thread)

main thread

Q. Can we invoke run() instead of start method?

→ Yes, we can invoke run method will be executed as normal method & seprate flow of execution will not be created.

→ Instead t1.start() → t1.run()
& after execution o/p = Chid - Thread Running
Main - " "

NOTE:-
Run() will be invoked by the thread
t1.start() → means start() contain internally run, so many methods in this way it is run mthod invoked

Co

o/p:-
Main - Thread Running Child - Thread Running
Chid - Thread " OR main " "

Can we override Stack()?

→ Yes, we can override stack() but in this case we are not giving chance for super-cls (Thread-cls) start method to execute so that the seprate flow will not be created.

NOTE:-

if the thread has to considered as a valid thread then it is mandatory Thread cls start() either directly or in-directly on that perr particular thread.

28/09/24
Saturday

Thread-Class Constructors:-

(i) public Thread()

(ii) public Thread(String name)

(iii) public Thread(Runnable r)

(iv) public Thread(Runnable r, String name)

2nd Way of Creating Thread object:-

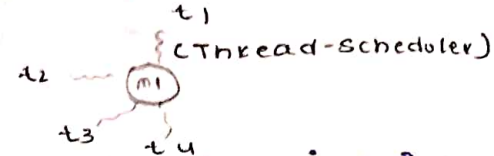
```
class MyThread implements Runnable
{
    @Override
    public void run()
    {
        s.o.p("Chid-cls Thread Running");
    }
    psvm() {
    MyThread mt = new MyThread();
    Thread t1 = new Thread(mt);
    t1.start();
    t1.start(); → Exception occur we cannot reborn-thread.
    s.o.p("main-Thread-Running");
}
}
```

Olp:- Here to olp cannot be predicted child-Thread may execute or Chid-Thread coz it depend on JVM to JVM

01. In two ways which one have more advantage.

- By Implementing runnable interface coz in this case we can use the benefits of inheritance.
- In case of extending to Thread cls we miss the benefits of inheritance.

Thread-Scheduler:-



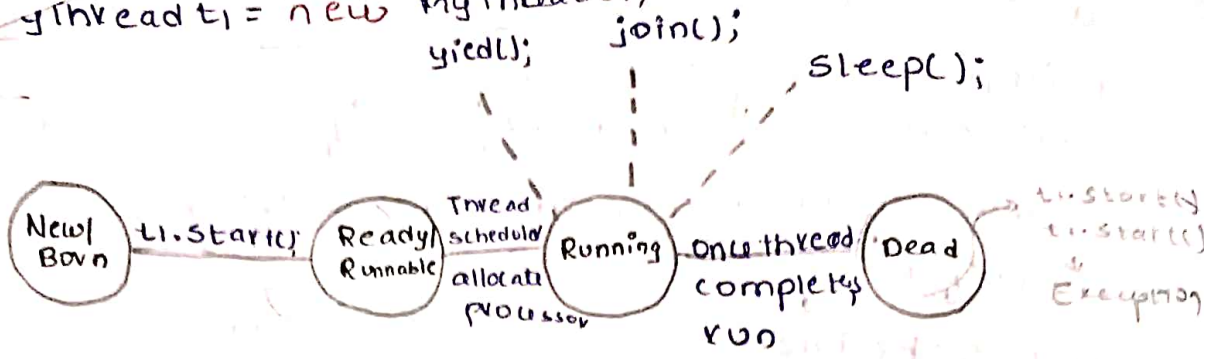
- It is the part of JVM.
- It is responsible to schedule threads i,e if multiple threads are waiting to get the chance of execution then in which order threads will be executed is decided by thread scheduler.
- We can't except exact algorithm followed by thread scheduler it is varied from JVM to JVM hence we can't expect threads execution order & exact o/p.
- Hence, whenever situation comes to multi-thread there is no gurantee for exact o/p but we can provide several possible o/ps.

Methods of Thread:-

Modifier & Type	Method	Description
void	start()	It is used to start the execution of the thread
Void	run()	It is used to do an task/job for a thread
static void	sleep()	It sleeps a thread for the specified amout of time
static void	yield()	It causes the currently executing thread obj to pause & allow other threads to execute temporarily
Void	join()	It waits for a thread to die

Thread - Life cycle :-

MyThread t1 = new MyThread();



New/Born :- Whenever the thread obj is created it is always in the new state or born state & allocated with the memory resource

- The thread in new state code has been not yet run & thread has not began its execution

Ready/Runnable :-

- In this phase thread cls start() is invoked
- once the start() invoke thread enters into ready or runnable phase
- In this phase the thread is ready to run @ any given instance of tym.
- know its a responsibity of a thread scheduler to allocate the processor

Running phase :-

- once the thread scheduler allocates the processor to thread gets the CPU & moves from Runnable to running state.
- Generally in this phase thread enters into running state & actively executing the code & perform its task.

Dead (Terminated) :-

- A Thread reaches to terminated state cuz of two reasons

(1) Coz when a thread finished its task then it enters into terminated state / dead state.

ii) In case of abnormal termination [illegible] when unusually event such as @unhanded exception occurs then due to that unhanded exception it enters to dead state

- A terminated thread means a thread is no more in the stm i.e a thread is dead & there is no way to re-born the dead thread.

- When we try to restart the thred it throws an exception called illegalThreadStateException.

Thread Priority:—

- Each thread has a priority.

Priorities are represented by a no blw 1 & 10.

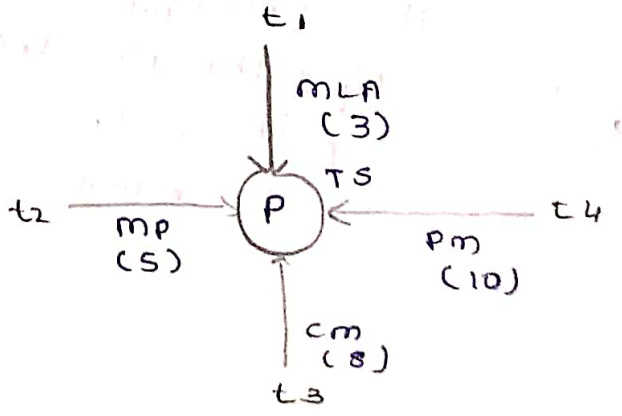
In most cases, the thread scheduler schedules the threads according to their priority.

- Jvm assigns priorities for Threads.

Note that not only Jvm a Java programmer can also assign the priorities of a thread explicitly in a Java program.

- The max priority is 10 & min priority is 1 & normal/default priority is 5.
- When both the thread having same priority then the Thread scheduler will decide which thread must be excluted first based on FCFS, Round robic etc (Algorithm).

MIN_PRIORITY;
NORM_PRIORITY;
MIN_PRIORITY;



Ex:-

```
Public class MyThread extends Thread {
    PSVM(){
    MyThread t1 = new MyThread();
        t1.start();
    S.O.pln("max:" + Thread.MAX-PRIORITY);
    S.O.pln("Norm:" + Thread.NORM-PRIORITY);
    S.O.pln("MIN:" + Thread.MIN-PRIORITY);
    }
    }
```

Olp:- max: 10
Norm: 5
min : 1

Methods to prevent the Thread from the Execution: —

(i) yield()
(ii) Join()
(iii) Sleep()

yield():-

- yield() is used to pause the current executing thread to give the chance for waiting thread of a same priority: ⊙, if the waiting threads have low priority then the same thread will continue the execution.
- If the multiple threads are waiting with the same priority then Threadscheduler will decide based on a algorithm.

```
public class MyThread extends Thread {
@Override
public void run() {
for(int i=0; i<10; i++) {
 s.o.pln("Child - Thread Running");
}
Thread.yield();
}
psvm() {
MyThread t1 = new MyThread();
  t1.start();
 for(int i=0; i<10; i++) {
s.o.pln("main - Thread Running");
 }
 }
}
```

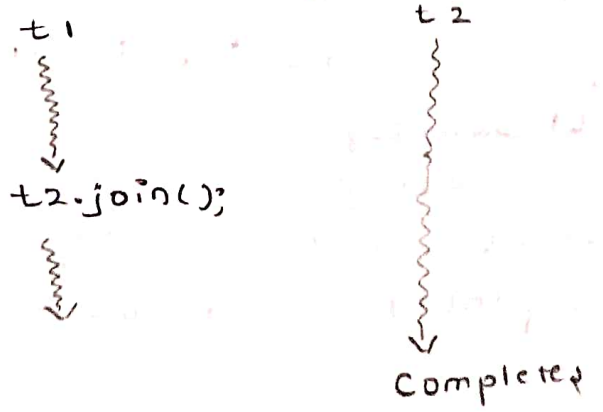
Try → Thread.yield()

NOTE:-

Even we use yield sometimes o/p cannot be predicted

Join():-

- If a thread needs to wait for another thread to complete the execution then we need to use join()
- Once we invoke join() on thread t1 then t2 thread enters into waiting stage (blocking for joint) and it will be waiting state until the thread t2 completes the execution.



- There are 3 variants of join()

(i) public final void join() throws InterruptedExceptio

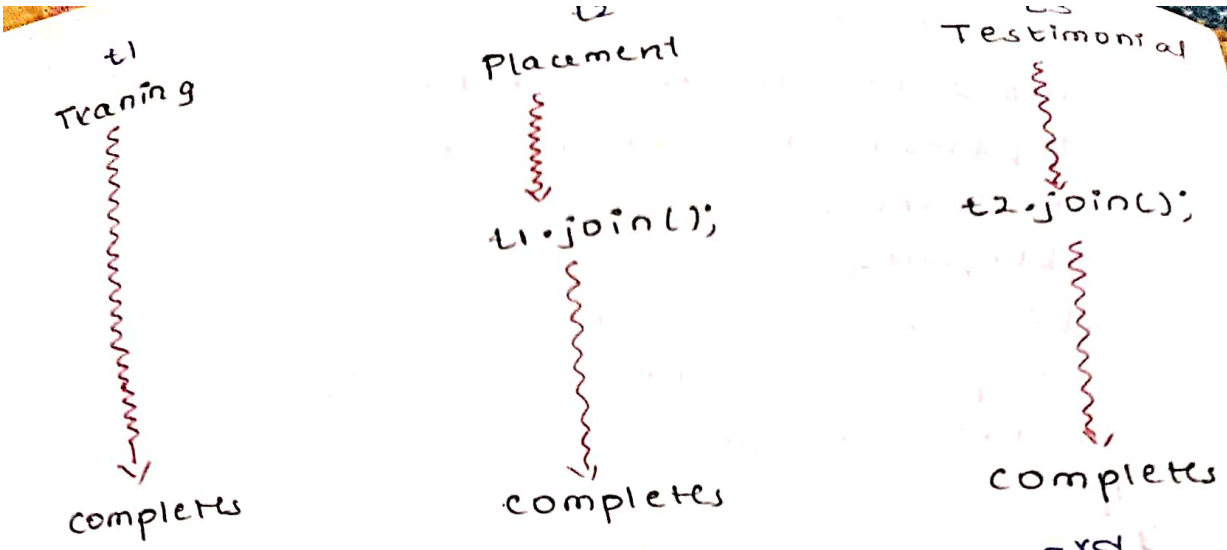
(ii) public final synchronized void join(long mls) throws InterruptedException

(iii) public final synchronized void join(long mls, int nanos) throws InterruptedException.

Whenever we use this 3 overloaded variants it throws an Exception called InterruptedException.

Ex:-

```
public class MyThread extends Thread {
    @Override
    public void run() {
    for(int i=0; i<10; i++) {
    System.out.println("Child - Thread Running");
    try {
        Thread.sleep(2000);
    }
    catch(InterruptException e) {
        e.printStackTrace();
    }
}

    PSVM() throws InterruptedException {
Thread t = Thread.currentThread();
MyThread t1 = new MyThread();
    t1.start();
    t1.join();
for(int i=0; i<10; i++) {
System.out.println("main - Thread Running");
}
}
```



1st 2nd 3rd

copy same code as deadlock but in main() don't write t1.join()

- DeadLock: -
- Two threads waits for each where waiting never ends.


```
public clas MyThread extends Thread$
  static Thread mt;
  public void run()$
   try$
      mt.join(); → Waiting for main method to execute
   }
   catch(InterruptedException e)$
      e.printStackTrace();
    }
   for(int i=0; i<10;i++)$
   S.O.pln("Child-Thread Running");
  }
}
```

```
Psvm() throws InterruptedException {
    mt = Thread.currentThread();
    MyThread t1 = new MyThread();
    t1.start();
    t1.join(); → Waiting for child class to execute
    for(int i=0; i<10; i++) {
    s.o.pln("Main - Thread Running");
     Thread.sleep(2000);
  }
}
```

Olp:- Blank.

To join same thread / To achieve deadlock in single line

copy same code & chang in main method

```
Psvm() throws InterruptException {
    mt = Thread.currentThread().join();
  }
```

or

```
Psvm() throws IE {
    mt = Thread.currentThread();
      mt.join();
  }
```

sleep():-

- It is a method used to pause the current executing Thread for particular amt of tym.
- There are two overloaded varients of sleep()

(i) public static void sleep(long mls) throws InterruptedException

(ii) public static void sleep(long mls, int n) throws IE.

- Whenever we use any of this overloded varients it throws an exception called IE.

Ex:-

```
public class MyThread extends Thread;
  @Override
  public void run();
  for(int i=1; i<=10; i++);
  s.o.pln("Slide-" + i);
  try{
    Thread.sleep(2000);
  }
  Catch(InterruptedException e);
    e.printStackTrace();
  }
}

Psvm()
 {
  MyThread t1 = new MyThread();
    t1.start();
 }
}
```

o/p:-

Slide - 1

:

Slide - 10

Interrupt:—

- Interrupt() is used to interrupt sleeping thread.

same as previous

```
        catch(IE e){
        s.o.println(" i got interrupted");
    }
}

psvm(){
MyThread t1=new MyThread()
  t1.start();
  t1.interrupt();
  }
```