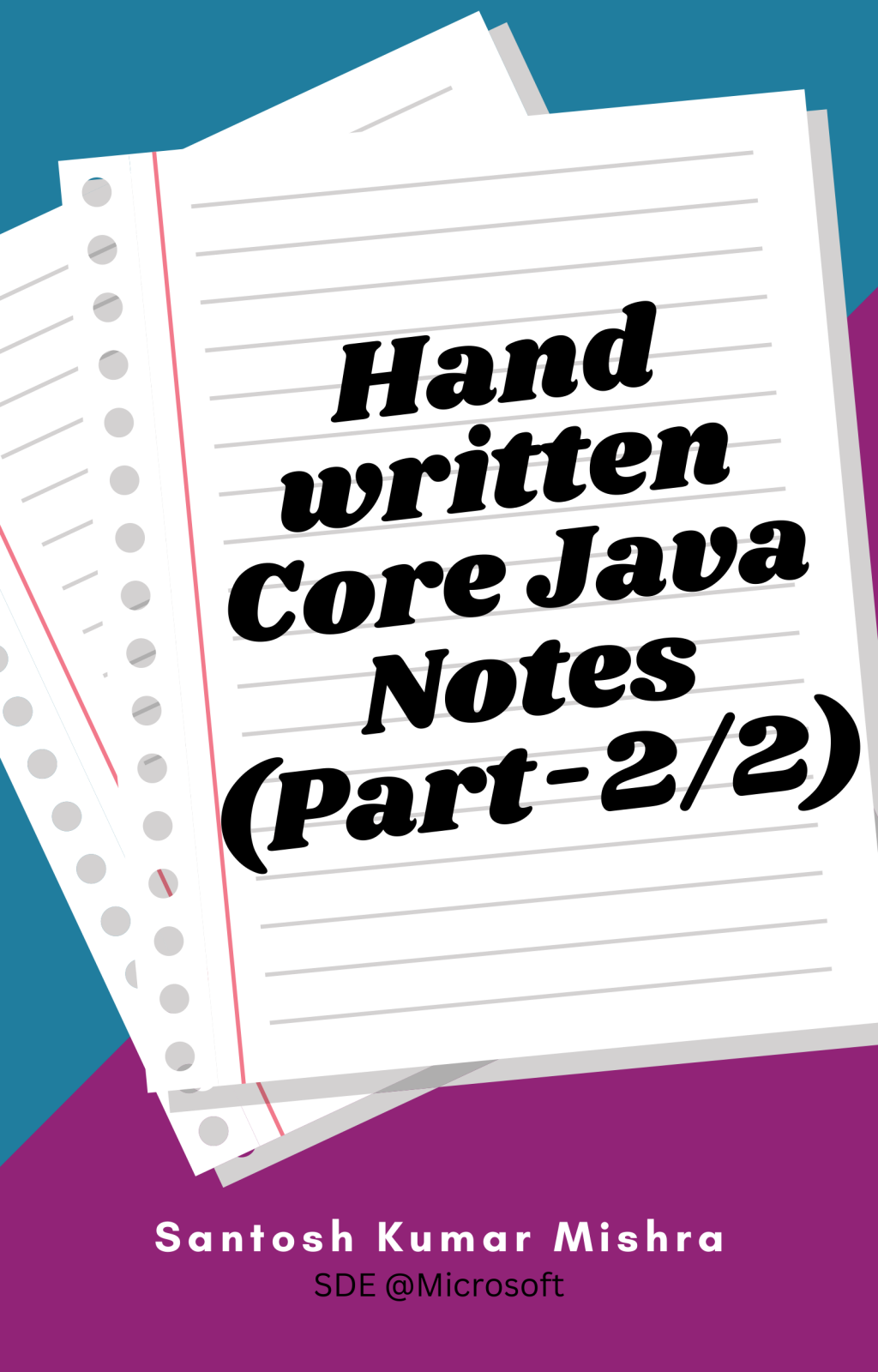


Santosh Kumar Mishra
SDE @Microsoft

* Exception Handling :-

The basic aim of exception Handling is to build robust (strong) applications.

In any programming language, if we write any program then we get some errors.

In s/w Development errors are classified into 3 types they are,

① Compile time errors.
② Logical errors.
③ Runtime errors.

Compile time errors are those which are occuring at compile time due to syntaxes are not followed

compile time errors will be solved by the programmer at development / production level.

Logical errors are those which are occuring at execution / Runtime because of misinterpretation of logic

logical errors always gives wrong output or inconsistent result and it will be solved by programmer

Runtime errors are those which are occuring at runtime due to invalid input entered by

application users, In other words runtime errors will occur during implementation level provided application user enters wrong input.

The languages like C, CPP, pascal, COBOL, etc & whose appln's are treated as weak. because these languages does not contains a facility called exception handling.

The languages / Technologies like java & .Net and their appln's are treated as robust bcoz these languages contains exception handling facility.

* points to be Remembered.

① When the appln user enters invalid input, we get runtime errors.

② Runtime errors always generates system/ technical error messages.

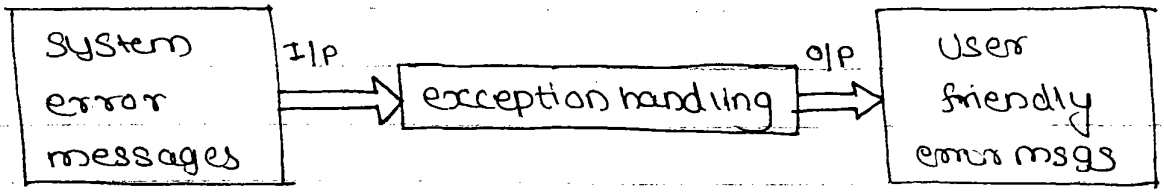
③ Runtime error of java program is known as Exception

④ Exceptions always generates system/technical error messages (Invalid inputs always causes exceptions)

* ⑤ Whenever an exception occurs in the java program, prg execution is abnormally terminated cpu control comes out of prg flow & JVM generates system error messages.

In the s/w development generating system error messages are not recommended but industry is always recommended to generate user friendly error messages.

Defination of Exception Handling :-
The process of converting system error msgs into user friendly error messages is known as exception Handling.



" Each and every action/task/operation must be performed wrt object in exception Handling"

28/8/13

We know that whenever an exception occurs in the Java program, program execution is abnormally terminated, Cpu control comes out of the program flow & JVM generates system error message Generating system error messages is an action/performance & it can be performed wrt an object programmatically generating system error msg. requires object creation. and to create an object we require a class this class is known as exception class. Hence JVM will create an object of appropriate exception and generates system error message provided appn user enters Invalid input.

If this exception class is developed by SUN ms developers then it is known as predefined exception. If the exception class is developed by java programmer then it is called user defined exception

Based on the development of exceptions, exceptions are classified into two types they are ① predefined (Built-in exceptions)
② user / programmer / predefined / built-in operations

predefined exceptions are those which are developed by SUN ms & supplied as a part of java SW (JDK) & they always deals with universal problems.

Some of the universal problems are,
ⓐ Division by zero problems.
ⓑ Invalid number formats.
ⓒ Invalid bounds of the array, etc

Userdefined exceptions are those which are developed by java programmers & supplied as a part of their project & they always used for dealing with common problems.

Some of the common problems are
ⓐ Trying to enter negative salary for an employee.
ⓑ Trying to deposit Invalid amount in the balance of account.

Trying to enter Invalid number of password characters.
Trying to enter Invalid username & password in the logging page, etc

* Types of predefined Exceptions :-

predefined exceptions are classified into two types they are,

① Asyncronous exceptions.
② Syncronous exceptions.

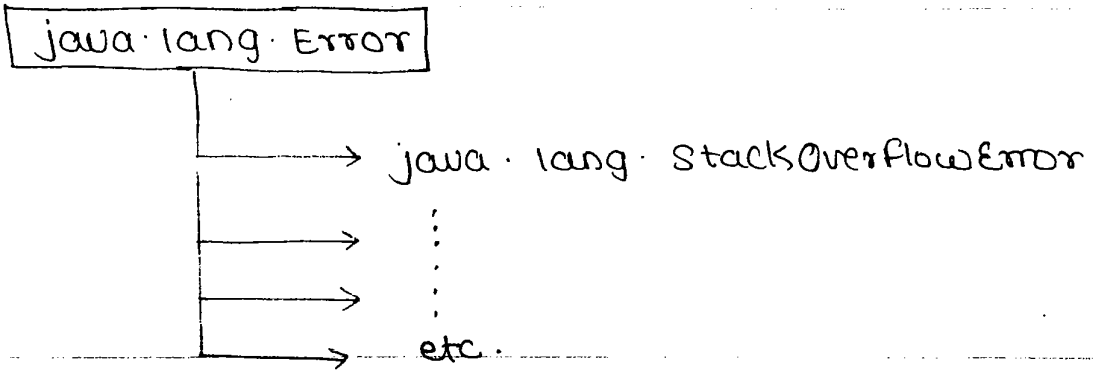
① Asyncronous exceptions :-

These exceptions always deals with Hardware problems and external problems.

e.g: a) mouse / Keyboard / motherBoard failure } HW problems
b) memory problems
c) power failure } external problems

As on today SUN ms developers has completly not developed API for asyncronous exceptions i.e, partially developed. Hence as on today industry is not using asyncronous exceptions.

To deal with all types of asyncronous exceptions we have a predefined class called java.lang.Error in other words java.lang.Error is the superclass for all asyncronous exceptions



② Syncronous Exceptions:-

These exceptions deals with programmatic runtime errors.

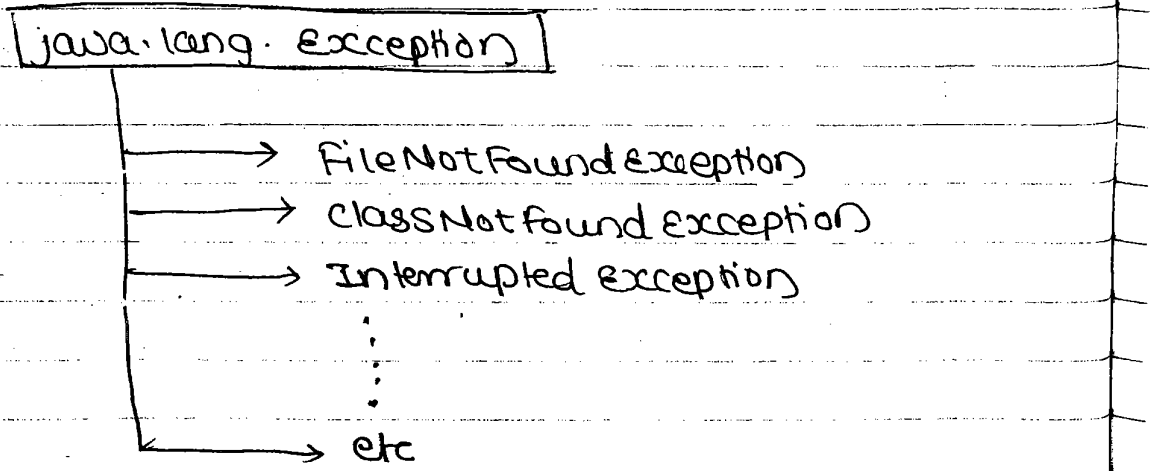
Syncronous exceptions are classified into two types They are,

ⓐ Checked exceptions.

ⓑ Unchecked exceptions.

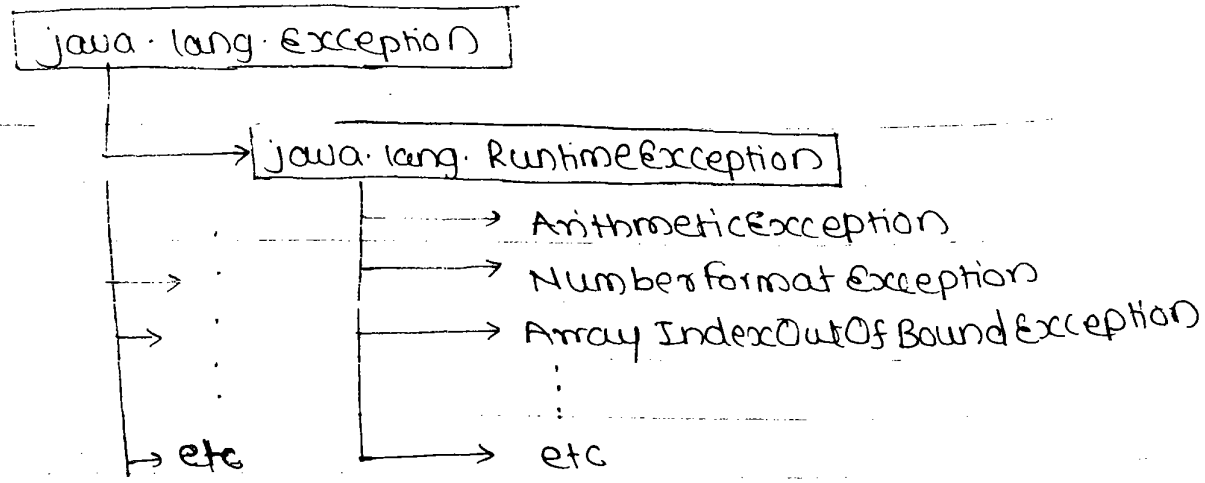
ⓐ Checked Exceptions:-

Checked exceptions are those which are the direct subclasses of java.lang.Exception ~~are~~ (OR) The exceptions which are occuring at runtime will be showing them as ~~errors in~~ compiletime ~~are known as~~ that is compile time regarding exceptions.



(b) Unchecked exceptions:-

Unchecked exceptions are those which are the direct subclasses of java.lang.RuntimeException



29/8/13

Q. What are the three main classes used for dealing with all types of exception?

① Java.lang.Error is used for dealing with all asynchronous exceptions.

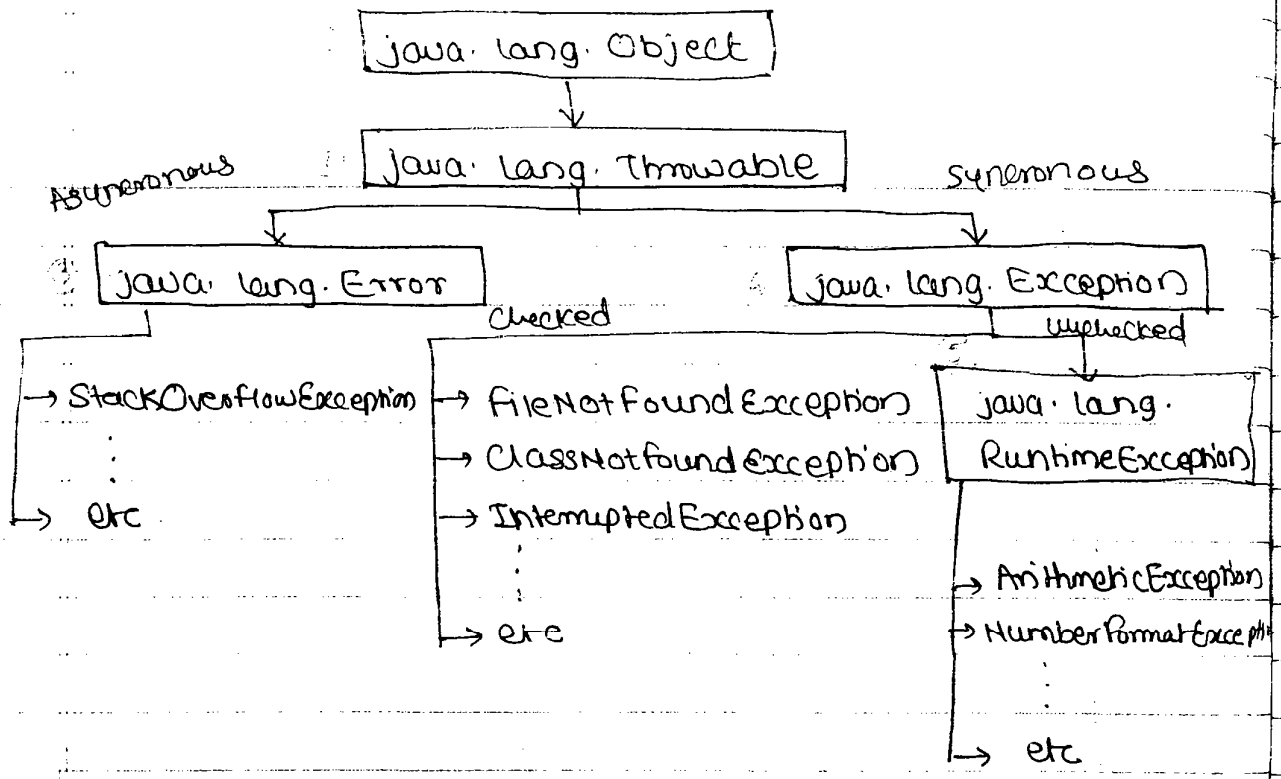
② java.lang.Exception is used for dealing with synchronous checked exceptions.

③ java.lang.RuntimeException is used for dealing with synchronous unchecked exceptions.

✱ Exception Handling Hierarchy Chart:-

Hierarchy charts always makes us to understand how the flow of features inherited from topmost base class to intermediate base classes and from intermediate base classes to bottommost derived classes.

Q. The following diagram gives Hierarchy of classes where flow of features are inherited.



1) In the above hierarchy chart java.lang.Object is the super class for all the classes in Java & it provides garbage collection facility to its subclasses for collecting unused memory space

2) java.lang.throwable is the super class for all the exceptions in Java. and it is used for deciding which types of exceptions occurs in the Java program (either Synchronous or Asynchronous)

3) java.lang.Error is the super class for all the asynchronous exceptions.

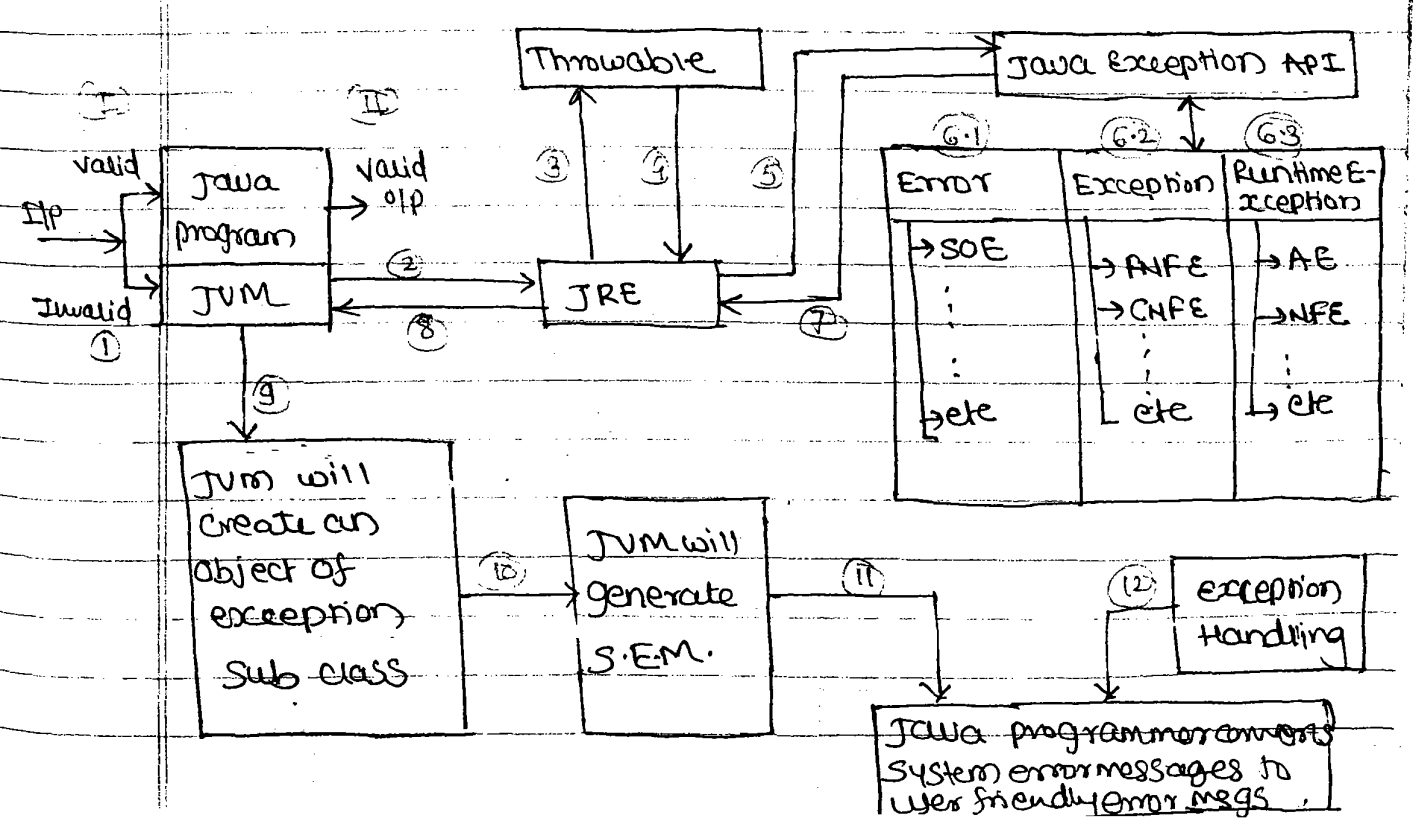
java.lang.Exception is the superclass for all syncronous checked exceptions.

java.lang.RuntimeException is the superclass for all syncronous unchecked exceptions.

NOTE:- Even though there exists multiple exceptions in a single java program, at any point of time JVM can handle / process only one exception at a time but not multiple exceptions.

* Internal flow Of Exception occurance:-

Whenever we write a java program there is a possibility of occuring many exceptions. The following diagram gives sequence of ~~steps for~~ of internal occurance of exception



(i) Application user enters valid input to the java program

(ii) JVM can process the java program with valid input & gives valid output.

(1) Appln user enters Invalid input to the java prg

(2) JVM can not process the java program with Invalid input and JVM contacts to JRE (JVM + Java API) for obtaining Exceptions of subclass.

(3) JRE inturns contacts to java.lang.throwable for obtaining which type of exception occurs in the program

(4) java.lang.throwable is a superclass for all the exceptions in java & it decides type of exception & gives same message backs to JRE.

(5) JRE further contacts to java Exception API. for obtaining Exception subclass

(6.1) (6.2) (6.3) Java Exception API executes either 6.1 (java.lang.Error for Asynchronous Exceptions) or 6.2 (java.lang.Exception for syncronous checked exceptions) or 6.3 (java.lang.RuntimeException for syncronous unchecked exceptions).

(f) Java exception API gives appropriate exception subclass to JRE.

(g) JRE gives Exception subclass back to JVM.

(h) JVM will create an object of appropriate exception subclass

e.g: for the division of two numbers prg If we enter 10 & 0 as an input (Invalid) then internally JVM will create an object of ArithmeticException programatically as shown below,

ArithmeticException ae = new ArithmeticException("/by zero");

(i) JVM will generate by default system error msg after the object creation of appropriate exception subclass.

e.g: The system error message for the exception which is occuring in eg of step(g) is, shown below,

```
exception in thread main: ---------
java.lang.ArithmeticException: / by zero
---------------------------
```

↑ system error message

In the s/w development it is not recommended to generate system error msgs & recommended to generate user friendly error msgs.

Java programmer converts system error msgs into user friendly error msgs by making use of exception Handling concept.

e.g: The Userfriendly message for the system error message which is occuring in the e.g. of step(10) is shown below,

Don't enter zero for Denominator

User friendly error msg.

30/8/13

* Handling the Exceptions:-

We know that whenever an exception occurs in the java program, JVM generates System error msg Generating system error msgs are not recommended in real world & industry is highly recommended to generate userfriendly error msgs by making use of exception handling concept.

Handling the exceptions are nothing but converting system error msgs into userfriendly error msgs to do this process as a part of exception handling we are having 5 keywords They are,

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

Syntax for Handling the exceptions:-

```
try
{
    Block of Statements ()
    . Causes problems at runtime;
}
```

```
catch(Type of exception 1 obj1)
{
   Block of stmts() provides
   user friendly error msgs;
}

catch (Type of exception2 obj2)
{
   Block of stmts() provides
   user friendly error msgs;
}
  :
  :
  :
catch (Type of exception n objn)
{
   Block of stmts() provides user
     friendly error msgs;
}

 finally
{
   Block of stmts()
   which will execute
     Compulsary;
}
```

(1) Try block :-

① In this block we always write block of stmts which causes problems at runtime or what are all the block of stmts generating exceptions, such block of stmts must be always return inside the try block

② When the exception occurs in the try block program flow is abnormally terminated, JVM control comes out of try block and executes appropriate catch block.

③ After executing appropriate catch block JVM control never goes to try block again to execute the rest of the stmts even we use return stmt in catch block.

④ Each & every try block must be immediately followed by catch block i.e. no intermediate stmts are allowed between try & catch block

⑤ for each & every try block we must write atleast one catch block. It is highly recommended to the java programmer to write multiple catch blocks for generating multiple user friendly error msgs

⑥ one try block can contain another try block i.e. nested / inner try blocks can be possible

(2) catch block :-

(1) It is the block in which we write block of statements and they provides user friendly error msgs. In other words catch block will supress System error msgs & generates user friendly error msgs.

(2) Catch block will execute provided exceptions occurs in try block.

(3) It is highly recommended to the programmer to write multiple catch blocks to generate multiple user friendly error msgs.

(4) At any point of time JVM can execute only one catch block among multiple catch blocks.

(5) Programmatically in the catch block declaration of appropriate exception objects done by Java programmer & internally exception will be referenced by JVM.

e.g:

```
catch (ArithmaticException ae = new ArithmaticException("/ by zero");
{ ------
  ------
}
```

Declaration done by Java programmer

referencing done by JVM

(6) one can also write try & catch block as a part of catch block.

31/8/13

③ Finally Blocks-

① In this block we write the block of stmts which will relinquish (release/close/terminate) the resources (files, databases)

② finally block will execute compulsorily.

③ Writing finally block is optional.

④ It is highly recommended to the Java programmer to write one finally block ~~for~~ per Java prg.

⑤ In finally block one can also write try & catch blocks.

* Case 1:- If exception occurs then,
A part of try block will execute.
An appropriate catch block will execute
A finally block will execute (If we write).

case 2:- If no exception occurs then,
A complete try block will execute.
finally blocks will execute (If we write).

Q. Write a Java prg which illustrate the concept of try block, catch block & finally block.

```
// Div.java
Class Div
{
  psvm (-)
  {
    try
    { String s1 = args[0];
```

```
        String s2 = args[1];
        int x = Integer.parseInt(s1);
        int y = Integer.parseInt(s2);
        int z = x/y;
        Sop(" Division is :" +z);
    }
catch( ArithmeticException ae)
{
   Sop(" Don't Enter zero at denominator");
}
 catch ( NumberFormatException nfe )
 {
    Sop (" Don't enters alphanumeric values");
 }
 catch( ArrayIndexOutOfBoundException ab)
 {
    Sop (" plz enter two values from cmd ");
 }
 finally
{
    Sop (" I am from finally Block");
}
} // main
} // class
```

21/9/13

Throws :-

Throws is a keyword which will express or describes the types of exception occuring in the body of common methods (OR).

Throws is a keyword which gives an indication to the specific method to place common exception method under try & catch block for generating user friendly error msgs.

According to Industry standards specific methods should always contain try & catch blocks for handling the exceptions & generates user friendly specific error msgs & it should not contain throws keyword.

Common methods of Java must always contains throws keyword for expressing the type of exceptions occuring as a part of common method body and it is not recommended to contain try & catch blocks.

The place of using throws keyword is always as a part of method heading.

Syntax:-

```
return type ... method name ( list of formal params if any ) throws {<clsname1, clsname2, .... clsnamen}
{
    Block of Stmts;
}
```

In the above syntax clsname1, clsname2, ... clsnamen represents name of the exception classes & they may belongs to either userdefined or predefined or both.

e.g:

java.lang.Integer

public static int parseInt(String) throws java.lang.NumberFormatException

ParseInt(-) method heading contains throws keyword sothat it is one of the common Exception method & it must be always placed within try & catch block.

Q Write a java prg which illustrate the concept of throws keyword which is used as a part of predefined methods

```
Class Ex2
{
    psvm(-)
    {
        try
        {
            int x = Integer.parseInt("10a3b");
            Sop("val of x:" +x);
        }
        Catch NumberFormatException
        {   Sop("Don't Enter Alphanumeric Values");
        }
```

Q. Write a java prg which illustrate the concept of ~~throw~~ ~~throwcat~~ throws which is used as a part of user defined common methods.

```
package SP;
public class Sai
{
    public void division (String S1, String S2) throws
     ArithmeticException & NumberFormatException
    {
       int x = Integer.parseInt ( S1 );
       int y = Integer.parseInt(S2);
        int z = x/y;
      sop(" div in sai:" +z))
    }
}
```

Compile the above program

Cmds javac -d . Sai.java

In Sp.Sai class division method is one of the user defined, common exception method & it must be always return within try & catch block

write a main program which is making use of Sp.sai class.

```
import Sp.Sai;
Class SaiStudent
{ psum (->
  {
```

```
try
{   sai so = new sai();
        so.division(args[0], args[1])
}
Catch (ArithmeticException ae)
    system.ers.println(" dont enter zero for
                        denominator");

Catch (NumberFormatException anfe)
    System.ers.pln(" dont enter alphanumeric
                    values");

finally
{
    system.ers.pln(" I am from finally block");
}
}
}
```

* Rethrowing the Exception:-

First Method is throwing the Exception to second method & second method is again throwing the first method Exception to third method with respect to second method first method related exception is known as Rethrowing the exception.

e.g: In the above program parseInt (first method) is throwing NumberFormatException to Division method (second method). The division method inturns throws NumberFormatException of parseInt

method to main method (third method). w.r.t the division method NumberFormatException of parseInt method is known as re-throwing the Exception.

* Dealing with details of unknown Exception :-

Whenever we write a java prg there is a possibility of number of exceptions in the Java prg & all these exceptions may not be aware of the type of exceptions occuring in java program. If the java programmer is not knowing about specific type of exceptions then the java programmer may not be able to generate specific usefriendly error msgs.

It is highly desirable to know about unknown exceptions. In java programming we have 3 approaches for finding the details about unknown exceptions they are,

1. By using an object of java.lang.Exception
2. By using printStackTrace() method. by
3. By using getMessage() method.

(1) By using an object of java.lang.Exception:-

We know that java.lang.Exception is the superclass for all synchronous Exception & according to the concept of Dynamic Binding, an object of java.lang.Exception is knowing

about all its subclasses.
An object of java.lang.Exception always gives following details.

@ Name of the unknown exception.
(b) Nature of the message.

e.g:

```
try
{ .....
  .....
  int c = 10/0;
  .....
} catch (Exception e) = new ArithmeticException("/ by zero");
  {     written by Java programmer.
    Sep(e); // java.lang.ArithmeticException: / by zero
  }
```

written by JVM

Name of exception | name of msg

3/9/13

② By using printStackTrace() :-

It is one of the predefined method present in java.lang.Throwable class & it is further inherited into both java.lang.Error and java.lang.Exception.

By using printStackTrace() method we can find the following details.

@ Name of the unknown exception
(b) Nature of the message
(c) Line number where the exception occured.

The prototype of printStackTrace method is shown below,

java.lang.Throwable

```
public void printStackTrace ()
```

Since this method is returning void as a return type, it should not be called as a part of ~~try~~ & cat sop or sep.

```
e.g: sop( e.printStackTrace());   } Invalid
     sep ( e.printStackTrace());  }
     e.printStackTrace ();          Valid
```

In java programming if any method returning void as a return type, it should not be called as a part of sop or sep

```
class Ex1
{ .....
  .....
   try
   {
5:   .....
6:   int c = 10/0;
7:   .....
   } catch( Exception e)          Name of Exception
     {
       e.printStackTrace(); // java.lang.ArithmaticException:
     }                          / by zero at Ex1.java:6
                                nature of msg   Line no
```

When the exception occurs in the java program before the program execution is abnormally terminated, the exception details will be pushed into the stack

Stack Memory.

Name of Exception
Nature of message
Line number

③ By using getMessage() :-

It is one of the predefined method present in java.lang.Throwable class & it is further inherited into both java.lang.Error & java.lang.Exception.

With getMessage() method we can obtain only nature of the message.

The prototype of getMessage() method is shown,

java.lang.Throwable

```
public String getMessage()
```

example:-

```
try
{ .....
  int c = 10/0;
  .....
} catch (Exception e)
{
  Sop(e.getMessage()); // prints: / by zero
}
```

* User | Secondary | Custom | Programmer Defined Exceptions :-

Defination :- User defined exceptions are those which are developed by java programmer & supplied as a part of their project & they always deals with common problems

Some of the common problems are,

@ Trying to enter Invalid salary to an employee.
⑥ Trying to deposite negative values in the balance of account.
© Trying to enter Invalid number of password characters.
@ Trying to enter Invalid username & password etc.

When the application user enters Invalid inputs to any java application, program execution is abnormally terminated by creating an object of appropriate exception subclass. If these exception subclasses developed by sun ms then they are called predefined exceptions. If these exception subclasses developed by java programmers then they are called userdefined exceptions.

What are all guidelines are followed by SUN ms developers for development of predefined exceptions, the same guidelines will be followed by the java programmer for development of user defined Exceptions.

* Steps | Guidelines for Development of User defined Exceptions :-

SUN ms developers has prescribed the following guidelines for development of user defined exceptions

(1) Choose an appropriate package for placing user defined exception subclasses for common access and ensure package statement must be the first executable statement.

(2) Choose an appropriate user defined class & ensure whose modifier must be public.

(3) The class which is selected in Step 2 must extends either java.lang.Exception or java.lang.RuntimeException for inheriting features of Exceptions. to stop the program execution abnormally when appln user enters Invalid input. Hence Step 2 class is known as userdefined Exception subclass.

(4) Each & every userdefined exception subclass contains parameterized constructor must take String as parameter. Here String parameter represents nature of the message.

(5) Each & every userdefined exception subclass parameterized constructor must call either parameterized constructor of java.lang.Exception or java.lang.RuntimeException by making use of super(msg). The reason for writing super(msg) is that if any specific exception occurs in the

java program & if such specific exceptions are not handled by java programmer then such specific exceptions will be handled by either java.lang.Exception or java.lang.RuntimeException.

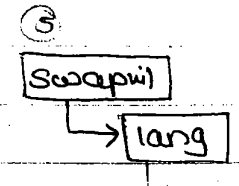
⑥ Whatever the userdefined exception subclass we are placing in the package such classname must be given as filename with an extension .java.

4/9/13

Q. Write a java program for developing user defined exception subclass for dealing with negative salary of an employee.

NSalException.java

```
package swapnil.lang;
public class NSalException extends Exception
{
    public NSalException (String s)
    {
        super (s);
    }
}
```



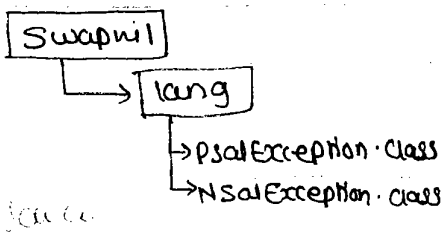
cmd> javac -d . NSalException.java

Q. Develop user defined exception subclass for dealing with positive salary of an employee

PsalException.java

```
package swapnil.lang;
public class PsalException extends RuntimeException
{
```

```
Public PsalException (String s)
{
    Super (s);
}
}
```



Here NSalException & PsalException are comes under userdefined exception subclasses.

In Slw development as a part of user defined exception development we develop 3 types of applications in 3 phases

① Development of programmer / userdefined exception subclasses

② Development of programmer / user defined common classes / general classes.

③ Development of programmer / user defined specific class.

① Development of programmer / userdefined exception subclasses :-

example:- NSalException.java, PsalException.java (see previous pages) Are comes under programmer / userdefined exception subclasses.

② Development of programmer / userdefined common classes :-

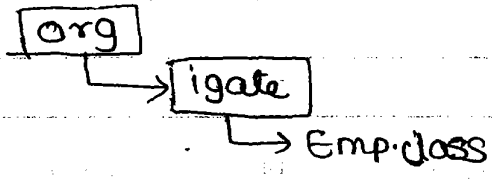
This phase of classes always deals with common requirements in the project. These classes

makes use of user defined exception subclasses.

example:- Write a programmer defined common class for dealing with salary of all the employees

```
package org.igate;
import swapnil.lang.NSalException;
import swapnil.lang.PsalException;
public class Emp
{
    public void decidesal (String s) throws NSalException, PsalException, NumberFormatException.
    {
        int sal = Integer.parseInt(s);
        if(sal<=0)
        {
            NSalException no = new NSalException ("Invalid Salary");
            throw(no);
        }
        else
        {
            PsalException Po = new PsalException("valid sal");
            throw (Po);
        }
    }
} // Emp ⇒ Programmer defined common class.
```

cmd> javac -d . Emp.java



③ Development of programmer defined specific class :-

These classes are always dealing with specific / individual requirements & these classes makes use of common classes which are developed in second phase & also makes use of exception subclasses developed in first phase.

Write a java program which will check salary of employee.

```
import javaorg.lgate.Emp;
import swapnil.lang.NSalException;
import swapnil.lang.PSalException;
class SwapEmp
{
    psvm (-)
    {
      try
      {
        String sal = args[0];
        Emp eo = new Emp();
        eo.decidesal (sal);
      }
      catch ( NSalException no)
       sep(" dont enter zero/-ve salary of emp");
      catch (PSalException po)
       sep (" valid salary");
      catch (NumberFormatException nfe)
       sep(" Don't enter alphanumeric salary");
      Catch (ArrayIndexOutOfBoundsException ab)
```

```
        finally
        {
            sop("I am from finally block");
        }
    }
}
```

(5) Throw :-

Throw keyword is always used for hitting / generating / raising the exception which is occuring as a part of method body. (Or) :-
throw is a keyword used for carrying the created exception as a part of method body & handover to throws keyword.

The place of using throws keyword is always as a part of method body.

When we use throw keyword as a part of method body, it is mandatory to the java programmer to use throws keyword as a part of method heading

Syntax1 :-

```
<clsname> objname = new <clsname>(msg);
        throw(objname);
```

Here <clsname> represents name of exception subclass

```
e.g: NoSalException no = new NoSalException("InvalidSal");
        throw(no);
```

Syntax 2:- (Throw kwd along with throws kwd)

```
Return type  method name ( list of formal params if any ) throws <clsname1>,<clsname2>
                                                            ..... <clsnamen>
{
    if (Test condn)
    {
      <clsname1> objt = new <clsname1> (msg);
                throw (obj1);
    }
    .
    .
    .
    if(Test condn)
    {
      <clsnamen> objn = new <clsnamen> (msg);
                throw (objn);
    }
}
```

In the above syntax <clsname1> ..... <clsnamen> represents either user defined exception subclasses or predefined exception subclasses or both

Example:- See decidesal (-) method of org.igate.Emp class (Emp.java).

Q. What are differences between throw & throws.

Throw	Throws
Throw is a keyword used for hitting or raising the exception which is occuring as a part of method body.	Throws is a keyword used for giving indication to the specific method to place common exception method under try & catch block

The place of using throw keyword is always as a part of method body	Throws is a keyword which is always used as a part of method heading
When we use throw keyword as a part of method body, it is mandatory to the java programmer to use throws keyword as a part of method heading	When we use throws keyword as a part of method heading it is optional to the java programmer to use throw keyword as part of method body

5/9/13

* Multithreading :-

Multithreading is one of the distinct feature of java programming.

The aim of multithreading is to provide concurrent execution (If we perform multiple operations simultaneously or parallelly then such type of execution is known as concurrent execution).

A flow of control is known as thread.

The purpose of thread is to perform or to execute the logic of the java program which is written in the form of userdefined methods.

If a java prg. is containing multiple flow of controls then it is known as multithreaded.

In s/w development we have two types of languages they are,
(a) Single threaded modeling languages.
(b) Multithreaded modeling languages.

The languages like C, C++, pascal, COBOL, etc are treated as single threaded modelling languages because their execution environment provides only single flow of control, provides sequential execution, takes more execution time. and these languages does not contain any library for development of multithreading based applications

The languages/Technologies like Java, .Net are treated as multithreaded modelling languages because their execution environment provides multiple flow of controls, provides concurrent execution, takes less execution time and these languages contains an effective API. for development of multithreading based applications.

In Java programming for the development of multithreading applications we use following API,
(a) java.lang.Thread (class)
(b) java.lang.Runnable (Interface).

When we write any java program there exists two types of threads they are,
(a) foreground/child Thread.
(b) Background/Parent Thread.

A foreground thread is one which is always used for executing the logic of a java prg. which is written in the form of user defined methods.

A background thread is one which is always monitoring the execution status of foreground threads.

By default every java program contains single foreground thread & single background thread programmatically a java prg may contain multiple foreground threads to execute multiple user defined methods concurrently & recommended to have one background thread for monitoring the execution status of foreground threads

The real time implementation of multithreading concept is that to develop real world server softwares such as Tomcat (Apache jakartha slw foundation), Weblogic (BEA systems), Websphere (IBM) In general most of the third party slw vendors (except microsoft) developed their server slw's in java language by making use of multithreading concept.

Hence multithreading of Java is one of the specialised form of multitasking of operating system.

Q. How do you justify "Each & every java prg is multithreaded" ?

ANS: When we execute the java prg internally the logic of the java program is executed by one of the thread known as foreground thread.
TO monitor the execution status of foreground thread, internally there exist one more type of thread known as background thread so a java program is by default contains two threads ~~be~~
Hence every java prg is multithreaded.

Q. What are the differences between program & process ?

Program	process.
Set of optimized instruc-tions is known as program	A program is under execution known as process.
Program exists in the secondary memory for long time.	Process resides in main memory for limitted span of time

6/9/13

In real world as an application developer we develop two types of applications they are.
① process based / Non multitasking appln.
② Thread based / multitasking appln.

process based	Thread based
In the execution env of process based appls there exists single flow of control.	In the execution env of thread based appln there exists multiple flow of controls.

All the appl^s of single threaded modeling lang. (C, CPP, Pascal, etc) are comes under process based applications.	All the applications of multithreaded modeling lang (Java & .net) are comes under thread based appl^s
In process based appl^s we get only sequential execution but we are unable to get concurrent execution.	In thread based appl^s we can get both sequential and concurrent executions
In process based appl^s Context switch is more	In thread based appl^s context switch is less.
In process based appl^s for each & every sub prg there exist a seperate address space	In thread based appl^s, irrespective of number of sub programs there exist single address space.
All the process based appl^s are treated as heavy weight bcoz whose processing time is more.	All the thread based appl^s are treated as light weight component becoz whose processing time is less.

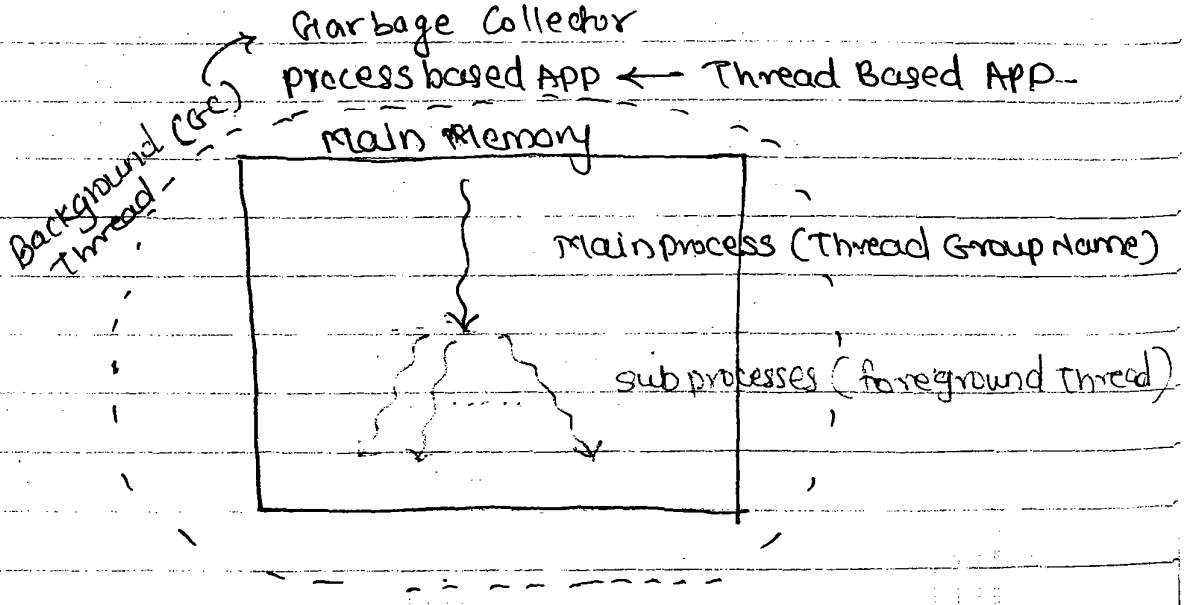
NOTE:- The amount of memory space created by operating systems on the stack memory for the temporary execution of the method is known as address space

Context switch:- The mechanism of switching the control of CPU from one address space to another address space is known as Context switch.

for the best appl's context switch must be less.

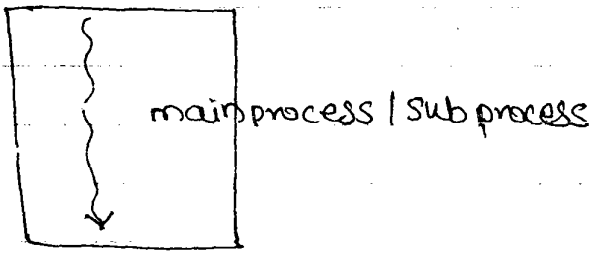
A Heavy weight component is one whose processing time is more. a light weight comp. is one whose processing time is less.

* Each and every thread based application is one of the process based appln and in which one main process will be created and the main process inturns creates number of subprocesses. In Java programming point of view the main process is known as thread groupname and subprocesses are known as foreground threads



A process based application is not treated as thread based appln bcoz it doesn't have the nature of creating multiple flow of controls

Process based App ← process based App



7/9/13

* States / Lifecycle of a Thread :-

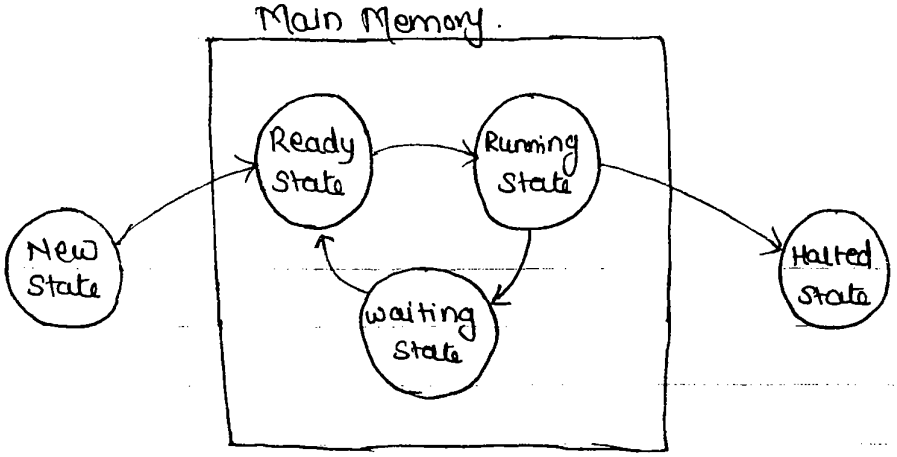
When we write any multithreading program there is a possibility of containing many number of threads. During these threads execution, threads will undergoes various stages & these stages are known as states. In otherwords the different breaking points occuring in the context of threads execution are known as states / lifecycle of a thread. The states of the thread are classified into 5 types they are,

1. New State – out memory State
2. Ready State
3. Running state
4. Waiting state.

(2, 3, 4) – In-memory states

5. Halted State. – out memory State

Statechart Diagram :-



① New State :- In new state thread is created and about to enter into main memory.

② Ready State :- In Ready State thread is entered into main memory, memory is allocated for the thread & first time waiting for the CPU.

③ Running State :- In Running State the thread is under the control of CPU. In other words is the thread is under the execution process of the CPU then the state of thread is called Running

④ Halted State :- In Halted state thread is completed its total execution.

CPU Burst Time :-

Def^n : The amount of time required by the thread from CPU for complete execution of user defined method is known as CPU burst time.

Wating state:- A thread is said to be in waiting state iff any of the following factors are satisfied,

(a) Threads are coming to the waiting state for the remaining CPU burst time (CPU burst time decided by OSP but not by programmer).

(b) suspending the currently executing Thread

(c) Making the currently executing thread to sleep. for some amount of time to be specified in terms of milliseconds (1sec = 1000 ms)

(d) Making the currently executing thread to wait with some amount of time in terms of msec's

(e) Making the currently executing thread to wait without specifying any waiting time

(f) Joining the thread.

Hence Ready, Running & waiting states are called in-memory states & whose execution status is "true"

New & Halted states are called out-memory states & whose execution status is "false"

Q. How do you justify threads of Java executing concurrently?

ANS: When threads are executing hour by hour, minute by minute

second by second then such threads are executing sequentially & whose execution time is more because the programmer is able to differentiate hour by hour, minute by minute, second by second.

- If threads of java executing with the difference of milliseconds of time then it is considered as all at once executing or concurrently executing & whose execution time is less/fast. because we are unable to differentiate the difference of milliseconds of time.

Therefore Java threads are designed in such a way that they are executing concurrently with the difference of milliseconds of time by following Round-Robin Algorithm.

10/9/13

* Creating Threads :-

In Java programming we have two approaches to create a flow of control or thread they are,

1. By using java.lang.Thread class.
2. By using java.lang.Runnable Interface.

1. By using java.lang.Thread class :-

In Java programming by using java.lang.Thread class we can create its object in 3 ways they are,

@ Directly by using new operator :-

```
Thread t1 = new Thread();
```

(b) By using factory method:-

```
Thread t2 = Thread.currentThread();
```

(c) By An object of subclass of java.lang.Thread class is an object of java.lang.Thread class.

```
class Th1 extends Thread
{ ......
  ......
  ......
}
Th1 t1 = new Th1();
```

Here t1 is an object of Th1 class and Th1 is a subclass of java.lang.Thread class & indirectly t1 is an object of java.lang.Thread class

Profile of java.lang.Thread class:-

public	static final	int	MAX_PRIORITY (=10)
public	static final	int	NORM-PRIORITY (=5)
public	Static final	int	MIN-PRIORITY (=1)

These data members are called thread priority modifiers Thread priority modifier are always used for setting priorities to thread. priorities always makes CPU to execute which one to execute first, which one to execute second & which one to execute last. Default priority of thread is norm priority NORM-PRIORITY

Constructors :-

① Thread ()

This Constructor is used for creating an object of thread class without giving userfriendly name to the thread.

e.g: Thread t_1 = new Thread ();

Here t_1 is called object name & the default name of thread is thread-0.

In general if we create n objects by using this default constructor then the default names of the thredes starts from Thread_0, ..., Thread_n-1. Giving user friendly names are more recommended compare to default names.

② Thread (String)

This constructor is used for creating an object of string class by giving user friendly name to the thread.

ex. Thread t1 = new Thread ("sairam");

Here t1 is called object & sairam is called user friendly name to the thread.

③ Thread (Runnable)

This constructor is used for converting Runnable Interface obj

④ Thread (Runnable, String) :- The functionality of this constructor is exactly similar to the above constructor but with this constructor we are able to give user friendly name to the foreground thread.

```
ex: Runnable r = new Th1();
    // r.setName("java"); //Invalid
    Thread t1 = new Thread(r);
    t1.setName("java");
    Sop("Name of FGT=" +t1.getName());
```

OR

```
Thread t = new Thread(r, "java");
Sop("Name of FGT=" + g etName());
```

* **Instance Methods:-**

① public final void setName(String)
② public final String getName().

This methods are used for setting the user friendly names to the thread & getting the name of the thread.

```
ex: Thread t1 = new Thread();
                  └──→ Thread-0  By default.
    String tname = t1.getName();
    Sop(tname);
    t1.setName("sairam");
    tname = t1.getName();
    Sop(tname);
```

Q. Define Mutators & Inspector Methods.

Mutators are the methods used for changing the values of the object & whose general notation is setXxx(String).

Inspectors are the methods used for obtaining the values from the object & whose general notation is getXxx().

② public final void setPriority(int)
④ public final int getPriority().

This methods are used for setting the priority to the thread & getting the priority of the thread.
exc.

```
Thread t1 = new Thread("sairam");
int PV = t1.getPriority();
    sop(PV);
t1.setPriority(Thread.MAX_PRIORITY);
PV = t1.getPriority();
    sop(PV);
```

⑤ Public final boolean isAlive()

This method is used for checking the execution status of thread. This method returns true, provided the thread is in Ready Running, & waiting states. This method returns false provided the thread is in New & Halted states.

11/9/13

⑥ Public final void ~~start~~ start()

This method is used for making the thread to move from new state to Ready state, provides the Internal services of the multithreading (concurrency, synchronization, Interthread comm^n, etc) and automatically calling ~~Run~~ run() method

Ex:-

```
Thread t1 = new Thread();
```

```
boolean b = t1.isAlive();
    sop(b);                    // false
t1.start();
b = t1.isAlive();
 sop(b);
```

⑦ public void run().

This method is always used for providing the logic of a thread. Originally this method is defined in Thread class with null body. As a java programmer we provide logic to the thread by overridding run() method.

Programmatically choose an appropriate user defined class, it must extends java.lang.Thread class and override the null body run() method

ex: (Skeleton e.g)

```
class Th1 extends Thread
{
    public void run()
    {
        ....          // overridden
        ....          // logic of a Thread
    }
}

Th1 t1 = new Th1();   // new state
boolean b = t1.isAlive();
    sop(b);           // false
    t1.start();       // Enters into Ready state
    b = t1.isAlive();
    sop(b);           // True
```

If we want to develop any thread based appln then we must always create an object of subclass of java.lang.Thread class.

* It is not recommended to the java programmer to call the run method directly because threads will not get services of multithreading & they are provided by start() method.

⑧ public final void suspend().

This method is used for suspending the currently executing thread. When we call suspend method upon Thread class object then it performs following operations internally

ⓐ It stops current Thread execution.

ⓑ It stores temporary execution details into PCB (process control Block).

ⓒ It make the thread to move from running state to waiting state & it will be in waiting state for a long time until we call back to ready state.

eg: t_1.suspend();

PCB:- (Process control Block)

PCB is an buffer created by operating system for each & every Thread when it enters first time into ready state

⑨ Public final void resume()
This method is used for Transferring the Thread from waiting state to ready state. When the Thread is resumed (Restarted) then it starts its execution or continue its execution where it left off before its suspension by retriving the temporary execution details from PCB.
e.g: t1.resume();

⑩ Public final void stop()
This method is used for ~~suspending~~ stopping or terminating currently executing Thread. When we call stop() method upon the thread class object, it stops its execution, it gives Temporary results to the programmer & the thread sending to the halted state. When the stopped Threads starts their execution, They starts their execution from the beginning irrespective of where it stops.

12/9/13

⑪ Public final void join() throws java.lang.InterruptedException.
This method is used for joining all the threads which are completed their execution as a single unit & collectively handovering all the threads at a time to garbage collector.

The default environment of multithreading says collects the threads individually after their completion of execution & individually

handovering those threads to the garbage collector which is one of the time consuming process.

When we create multiple threads in the java prg it is highly recommended to the java programmer to join all the threads as a single unit & handover to the garbage collector. This process improves performance of multithreading based applications.

When InterruptedException occurs

InterruptedException never occurs in Standalone (MIC) but it always occurs in busy environment like client-server env. because of mismemory management of server side operating system.

Let us Assume there exists n number of threads are created, started their execution & n-1 threads are completed their execution & joined & still nth thread is under execution. Due to mismemory mgmt of server side operating system ThreadGroup Name is attempting to collect all the threads even though nth thread is under execution. At this point of time with respect to nth thread JVM generates a predefined exception called java.lang.InterruptedException.

InterruptedException is one of the checked exception & it must be always ~~return~~ written within try & catch block.

Ex:

```
try
{
   t1.join();
   t2.join();
    :
   tn.join();
}
catch (InterruptedException ie)
{
  sop(" Problem in thread Execution");
}
```

* Static Methods :-

① Public static final Thread currentThread()

It is one of the factory method & it is used for obtaining the name of the threads which are currently running.

e.g: Thread t = Thread.currentthread();

sop(t); // Thread[main, 5, main]

PG-T, priority, TGN

t.setName("Sai");

Sop(t); // Thread[Sai, 5, main]

PG-T, pr, TGN

② public static final int ActiveCount().
This method is used for finding the number of threads which are under running in other words this method gives number of ~~the~~ thread which are present in In memory states.

```
int ac= Thread.activeCount();
        sop(ac);
```

③ public static final void sleep(long msec) throws java.lang.InterruptedException.
This method is used for making the currently executing ~~thread~~ to ~~sl~~ sleep for some amount of time in the form of milliseconds (1sec=1000ms) Once the sleep time will over automatically the thread will be entered into ready state from waiting state. When the thread is under sleep further stmts can not be completed until the sleep time is will be Completed.

ds

When the InterruptedException occurs.
Let us assume first program threads are sleeping in some location of the Server ~~under~~ & at the same time second prg threads are trying to sleep where first prg threads are sleeping due to mismemory mgmt of Server side operating system. Wrt first program threads JVM generates predefined exception called java.lang.InterruptedException.

e.g:

```
try
{ .....
  .....
  Thread.sleep (10000);
  .....
}
Catch (InterruptedException ie)
{
   Sop(" problem in thread execution");
}
```

Q. Write a Java prg which will print or generate priliminary information about the thread such as The name of threads which are by default running, execution status of those threads which are running, default name of the foreground thread, execution status of the foreground thread, Thread priority modifier values, etc

ThDemo1.java

```
Class ThDemo1
{
   PS VM(-)
   {
      Thread t1 = Thread.CurrentThread();
      Sop(" Default thread: " + t1); Thread[main,5,main]
      t1.setName(" sai");
      Sop(" Default Thread after modifying:"+t1);
                                  // Thread[sai,5,main]
      Sop (" Execution status of t1: " t1.isAlive()); true
      Thread t2 = New Thread (); user defined FGT
      Sop(" Default Name Of FGT:" + t2.getName());
```

// Thread-0

```
SOP("Execution Status of t2:"+t2.isAlive());
SOP("val. of Max priority:"+Thread.MAX_PRIORITY);
SOP("val. of Norm priority:"+Thread.NORM_PRIORITY);
SOP("val. of Min priority:"+Thread.MIN_PRIORITY);
    }
}
```

13/9/13

* Internal Flow of Thread :-

When we write multithreading program by creating multiple flow of control & when they are under execution, the following sequence of steps will takes place.

1. Java program starts executing
2. JVM creates Thread Group Name (TGN) (TGN always resides in main method)
3. TGN creates forground threads & they are always meant for executing run() methods (Because run() methods contains logic of the thread)
4. TGN dispatches (start) the foreground thread(s) To execute respective run() method.
5. foreground thread(s) executes respective run() method & gives result back to TGN. either one by one or all at once
6. TGN receives result from foreground threads either all at once or one by one.
7. TGN gives result to java programmer / Appl° user either all at once or one by one
8. TGN collects foreground thread(s) either all at one or one by one & hand over to background Thread (Garbage collector)

⑨ JVM collects TGN & handovers to background thread (Garbage collector).

⑩ Java program stops its execution.

Q. Write a thread based appln which will print one to ten numbers after each & every second.

ANS:

```
class Th1 extends Thread
{
    public void run()
    {
        try
        {
            for( i=0; i<10; i++)
            {
                sop(" val of i = " +i);
                Thread.sleep(1000);
            }
        } catch (InterruptedException ie)
        {
            sep(" problem in thread exec");
        }
    }

class Thdemo2
{
    psvm(-)
    {   Th1 t1= new Th1();
        sop(" excel. status of t1 before start:" +
                                    t1.isAlive());
        t1.start();
```

```
SOP(" exec. status of t1 after start: " + t1.isALive());
try
{
   Thread.sleep(5000);
} catch (InterruptedException ie)
   {
      sep (" problem in thread execution");
   }
SOP("execution status of t1 during exec:" +t1.isAlive())
try
{
   Thread.sleep(50001); CR : t1.join();
} catch (InterruptedException ie)
   {
      sep (" problem in thread execution");
   }
SOP(" execution status of t1 after completion:"
        + t1.isAlive());
   }
}
```

Q. How do you print name of foreground thread during run() method execution?

ANS: One can print name of the foreground thread by using two approaches They are,

Approach 1:-

Call currentThread() method & apply getName() in the run() method.

```
class Thi extends Thread
{
    public void run()
    {
        Thread t= Thread.currentThread();
        Sop("Name of ~~currentT~~ fGT:"+t.getName());
    }
}
```

Approach 2:-
Apply directly getName() method in the context of run() method

```
class Thi extends Thread
{
    public void run()
    {
        Sop("Name of fGT:" getName());
    }
}
```

14/9/13

* java.lang.Runnable (I)

Runnable is one of the predefined Interface present in java.lang.* package.

Runnable interface is one of the alternative mechanism for development of multithreading applications instead of using java.lang.Thread Class. The reason of introducing Runnable interface is that java programming does not support multiple inheritance through

the concept of classes but it can be supported by the concept of interfaces.

ex1: Class Th1 extends x, Thread //Invalid
{
.....
.....
}

ex2:- Class Th1 extends x implements Runnable
{ //valid.
.....
.....
}

Runnable interface contains only one abstract method & whose prototype is given below,

public abstract void run().

The run() method of thread class contains null body & the run() method of Runnable interface is abstract.

Industry is highly recommended to override run() method of Runnable Interface instead of overridding null body run() method of Thread class.

Hence in order to develop a multithreadi

appln by using Runnable interface we choose a class, it must implements Runnable (I) & overide run() which is inherited from Runnable (I)

```
class Thi implements Runnable
{
    public void run()
    {
        .....  } logic of Thread
        .....
    }
}
```

Q. Write a Java prg which will generate 1 to 10 numbers after each & every second by using Runnable (I).

```
class Thi implements Runnable
{
    public void run()
    {
        try
        {
            for(int i=1; i<=10; i++)
            {
                Sop("val of i=" +i);
                Thread.sleep(1000);
            }
        } catch (InterruptedException ie)
        {Sop(" problem in thread execution");}
    }
}
```

```
class Thdemo3
{
   psvm(-)
  {Runnable t1 = new Th1();
   Thread t2 = new Thread(t1, "sai");
   Converting Runnable interface obj into Thread
   class obj for entering into run() method by
   making use of start() of Thread class &
   giving user friendly name to T&T
   SOP("name of T&T" + t2.getName());
   SOP("exec. status of t2 before start"
                        + t2.isAlive());
      t2.start();
   SOP("exec. status of t2 after start="
                        + t2.isAlive());
  }
}

class Th1 implements Runnable
{
   Thread t1;
    Th1()
    {
       t2 = new Thread(this, "sai");
    }  t2.start();
    public void run()
    { SOP("name of T&T in run()="
        + t1.getName());
      try {....} catch {....}
}
```

```
Class Thdemo4
{
    psum(-)
    {
       Th1 t1 = new Th1();
    }
}
```

16/9/13

Q Write a java program by using Threads for Computing sum & subtraction of two numbers

```
Class Sum extends Thread
{
    int a,b,c;
    void set (int a, int b)
    {
        this.a= a;
        this.b = b;
    }
    public void run ()
    {
        c= a+b;
        Sop(" ~~val~~ sum in th1 =" + c);
    }
}

Class Sub implements Runnable
{
    int a,b,c;
    void set( int a, int b)
    { this.a=a;
      this.b=b;
```

```
        public void run()
        {
            c=a-b;
            SOP("Sub in thd = " + c);
        }
}

Class Thdemo3
{
    PSVM(-)
    {   if (args.length != 2)
            SOP("enter two values");
        else
        {
            int x = Integer.parseInt(args[0]);
            int y = Integer.parseInt(args[1]);
        // Create threads & set values
        Sum so1 = new Sum();
            so1.set(x, y);
        Sub so2 = new Sub();
            so2.set(x, y);
        // Start threads for computing sum & sub
            so1.start();
        // so2.start(); Invalid bcoz start() doesn't
            exists in Runnable(I) & Sub(C).
            Thread tso = new Thread(so2);
                tso.start();
        // join the threads
            try
            { so1.join();
              tso.join();
```

```
            catch (InterruptedException ie)
            {
               sop(" problem in thread execution");
            }
        sop("exec status of ts1 after join:" +s1.isAlive());
        sop("exec status of ts0 after join:" +ts0.isAlive());
    }
  }
}
```

NOTE :- If we have n number of independent operations & TO execute those operations concurrently then n number of independent operations related logics must be written in n number of Independent run() methods in the independent classes by extending thread class or implementing Runnable (I)

If we have multiple threads in our java program for executing independent operations concurrently & after they complete their execution then it is highly desirable to join those multiple threads as a single unit for collectively collecting all the threads & handovering at once to the garbage collector

Q. Write a java program which will implement banner Animation (OR)

Q. Write a Thread based application which will implement scrolling the given message.

```
import java.awt.*;
import java.applet.*;
/*<applet code = "Ani" height = 200
width = 200></applet> */
public class Ani extends Applet implements Runnable
{
  String msg = " OM SAI RAM";
    public void init()
    {
      setBackground (Color.BLUE);
      setForeground (Color.RED);
    }
    public void start()
    {
      Thread t = new Thread (this);
```

This refers to current class Obj, which implements Runnable interface & converts into thread class Obj

```
      t.start();
    }
    public void run()
    {   try
        {
          while (true)
          {
            Char ch = msg.CharAt(0);
```

```
            msg = msg.substring(1,msg.length());
            msg = msg + ch;
            Thread.sleep(1000);
            repaint();
          } // while
        } catch(InterruptedException ie)
          { sop("problem in thread exec"); }

      } // run
    public void paint(Graphics g)
    {
       Font f = new Font("arial",Font.BOLD,60);
          g.setFont(f);
          g.drawString(msg, 100, 100);
    }
} // Ani
```

```
cmd> javac Ani.java
cmd> appletviewer Ani.java
```

Q. Write a thread based appl^{n} for automatic incrementation of numbers

```
import java.awt.*;
import java.applet.*;
/* <applet code = "AutoInc" height = 300
        width = 300" >
        </applet> */
public class AutoInc extends Applet implements
Runnable
{  int i = 0;
```

```
);
    public void init()
    {
        setBackground(Color.yellow);
        setForeground(Color.BLUE);
    }

    public void start()
    {
      Thread t= New Thread(this);
        t.start();
    }
    public void run()
    {   try
        {
            while(true)
            {
                ++i;
                Thread.sleep(1000);
                repaint();
            }
        } catch(InterruptedException ie)
          { sop(" problem in thread exec"); }
    }
    public void paint(Graphics g)
    {  Font f= new Font("arial", Font.BOLD, 60);
        g.setFont(f);
        g.drawString(String.valueOf(i), 100, 100);
    }
}

5
```

17/9/13

* Syncronization :-

Syncronization is one of the OS based concept & implemented in java language for building consistent applications.

If we have multiple threads & if they decided to start & making use of same resource then the single resource will generate inconsistent results (wrong / deadlock results).

To avoid these inconsistence results we applied concept of synnonization.

Defination:-
The mechanism of allowing only one thread among multiple threads into sharable area to perform read & write operations & generates consistence result by eliminating inconsistent result is known as synnonization.

Need of syncronization:-
Let us assume there excists a sharable variable balance & whose initial value is zero. Let us assume there exists two threads t_1 & t_2 decided to deposite rupees 10 & 20 in the balance variable. Both the threads started their execution & completing their execution by depositing

10 & 20 rupees in the balance variable. After the threads completion the final variable value of balance variable is either to 10 or 20 but not 30 which is inconsistent results.

To avoid this inconsistent results we apply the concept of syncronization.

Syncronization concept is Applied on the above problem :-

Let us assume there exist 2 threads t1 & t2 decided to deposite rupees 10 & 20 in the balance variable. Both Threads started their execution for depositing rupees 10 & 20 in the balance variable (t1 started first & t2 started later with difference of milliseconds of time) At this point of time JVM gives value of balance (zero) to t1 & at the same time thread t2 is trying to access the balance variable value but JVM makes thread t2 to wait & until thread t1 completed its execution.

Thread t1 completed its execution the balance variable will be unlocked by JVM & the balance variable value 10 to Thread t2 given by JVM & once again JVM will lock the balance variable. Thread t2 also completed its execution, JVM will unlock value of balance & final value of balance variable is 30 which is

Hence in the syncronization concept, the JVM continues the locking & unlocking operation until all the threads completed their execution.

Thread Syncronization Techniques :-

In multithreading of Java we have two Thread Syncronization techniques they are,

① Syncronized methods
② Syncronized BLOCKS.

① Syncronized methods :-
If any ordinary method accessed by multiple threads concurrently then the ordinary method gives inconsistent results. To avoid this inconsistent results the ordinary method defn must be made as syncronized by using Syncronized keyword.

Based on the place we write Synoronized keyword before the method, they are classified into two types.

ⓐ Syncronized Instance methods
ⓑ Syncronized Static methods

ⓐ Syncronized Instance methods :- If an ordinary instance method accessed by multiple threads concurrently then the ordinary instance method generates inconsistent results. To eliminate these Inconsistent results, an ordinary

Instance method defination must be made as syncronized by using syncronized keyword.

Syntax:-

```
syncronized returntype methodname(list of formal params if any)
  {
     block of stmts();
  }
```

example:-

```
Class Account
{
    int balance=0;
    syncronized void deposit (int amt)
    {
       balance = balance+amt;
       sop(" Balance = "+balance);
    }
}
```

As long as one thread is executing syncronized Instance deposit method of Account class ther JVM will lock an object of Account class.

In general if one thread is executing syncronized Instance method then JVM will locks an object of comesponding class.

(b) Syncronized Static method:-

If an ordinary static method accessed by multiple threads then the ordinary static method generates inconsistent results. To avoid these inconsistent results, the declaration of ordinary static method must be made as syncronized by using syncronized kwd

Syntax:-

```
Syncronized Static Return type method name (List of formal params if any)
{
    block of stmts ();
}
```

example:-

```
class Account
{
    int bal = 0;
    Syncronized static void deposit (int amt)
    {
        bal = bal + amt;
        sop(" current balance: " + bal);
    }
}
```

Once the Thread is executing the above Syncronized Static deposit method, JVM will lock account class. In general once the thread is executing syncronized Static method then JVM will locks corresponding class.

Therefore in the syncronization concept of multithreading the JVM will lock either object or class depends on type of method (syncronized instance or syncronized static method) thread created

NOTE:-

① Once JVM locks an object then it is not possible to the java programmer to call non Syncronized Instance methods but it is always possible to call non syncronized static methods (bcoz class is free)

② Once class is locked by JVM then it is not possible to the java programmer to call any type of method of the class bcoz class is locked & obj creation is also not possible

(2) Syncronized blocks :-

Syncronized block is an alternative thread syncronization technique for archieving consistance results inste

Importance of Syncronized Block :-
If a derived class is Inheriting non syncronized Instance method either from base interface or base class & if the inherited non syncronized instance method accessed by multiple threads concurrently then Inherited non syncronized instance method generates inconsistent results. To avoid these inconsistent results, as a derived class programmer we are attempting to write "syncronized" keyword before ~~A~~ an inherited non syncronized instance method for making as syncronized method which is not possible. bcoz as a derived class programmer we should not change the prototype of either base class methods or base Interface methods. So. At this context we are unable to eliminate the Inconsistent results wrt. Syncronized methods hence at any point of time to get the Consistent results in all the circumstances we must use Syncronized blocks.

Syntax :-

```
Syncronized ( obj of currentclass)
{
   Block of Stmts ();
}
```

Syncronized blocks always written inside of non Syncronized instance methods but not in non Syncronized Static methods

Synchronized blocks always locks object of current class.

eg:-

```
package ap;
public interface Account
{
  void deposit(int);
}
```

non synchronized Instance method

Cmd> javac -d . Account.java

ap → Account.class

```
Class SavingsAccount implements Ap.Account
{
  int bal=0;
public void deposit(int amt)
{
   synchronize(this);
    {
      bal = bal+amt;
      sop("current balance:"+bal);
    }
}
```

non Synchronized Inherited Instance method

Synchronized block

Q. Write a java program which illustrate the concept of synchronization (problem stmt: let us assume there exist an account with a variable balance and whose initial value is zero.

Choose the appropriate methods for depositing the amount in the balance of account & retrieving the balance from account

Let us assume there exists n number of customers (Threads) who decided to deposite RS:10 for each in the balance of account Concurrently. Ensure during this concurrent execution consistent results must present in the balance of Account)

// Sync.java

```
Class Account
{
    private int bal=0;
    syncronized void deposit(int amt)
    {
        bal = bal+amt;
        sop(" current balance="+bal);
    }
    int getBal()
    {
        return bal;
    }
}

Class Cust extends Thread
{
    Account ac;
    cust( Account ac)
    {
        this.ac = ac;
    }
```

```
    public void run()
    {
        ac.deposit(10);
    }
}

class Sync
{
    psvm(-)
    { // Create single obj of Account class
        Account ac = new Account();
        // Create array of cust class objs (ie. threads)
        Cust cu = new Cust[5];
        //give the single obj Account class to array of cust
        for(int i=0; i<5; i++)                    Class obj
        {
            cu[i] = new Cust(ac);
        }
    // dispatch the array of cust to deposit Rs: 10/-
    for each in bal of Account class
        for(int i=0; i<5; i++)
        {
            cu[i].start();
        }
    // join the array of cust objs after their completion
        try
        {
            for(int i=0; i<5; i++)
            {
                cu[i].join();
            }
```

```
        }
        catch (InterruptedException e)
        {
            sop(" problem in thread execution");
        }
    sop(" total bal=" + ac.getBal());
    }
}
```

NOTE:-

① If we execute any synchronization based application (Sync.java) on mono/single user operating systems (DOS 3.1, windows 95) then it is mandatory to the Java programmer to write synchronized keyword wherever it is required otherwise we get inconsistent results bcoz even though synchronization concept is available in OS, which is not implemented in mono OS'es.

② If we execute any synchronization based application (Sync.java) on multiuser/multithreaded OS's (windows xp, NT, Unix, etc) then it is optional to the Java programmer to write synchronized keyword bcoz OS takes care about synchronization concept and all multithreaded OS's are containing default implementation for synchronization concept of OS. If we are not writing synchronized keyword then our Java appln depends on OS based synchronization &

it always gives final result as a consistent result but the result related to order of updation may or may not be consistent. which is not a recommended process.

(3) Industry is highly recommended to write synchronized keyword wherever it is required. irrespective of type of os we use If we write syncronized keyword then the java multithreading appln makes use of java based synchronization & it always gives consistent results in both final result & order of updation.

Q. How do you give an object of faculty class to student class?

One class obj can be given to another class obj by writing object parameterized constructor

```
class faculty
{
    int fno;
    string fname;
    faculty (int fno, string fname)
    {
        this.fno = fno;
        this.fname = fname;
    }
    void disp ()
    { sop ("facultyno:" + fno);
      sop ("facultyname:" + fname);
```

```
class Student
{
    Faculty fo;
    Student (faculty fo)
    {
       this.fo = fo;
    }
    void disp ()
    {
       sop(" student : disp ()");
          fo.disp();
    }
}

class SfDemo
{
    psvm (-)
    {
        faculty fo = new faculty (1, "scoop");
        Student so = new Student (fo);
              so.disp();
    }
}
```

21/9/13

* Interthread Communication (I.T.C) :-

Interthread communication of Java is one of the specialized form of interprocess communication of OS. In real world interthread communication appln's are very fastest appln's compared to any of the appln's in real world.

Defination:-

The mechanism of exchanging the data between multiple threads is known as interthread comm^n

- (OR) -

If the output of a first thread is given as input to second thread, o/p of second thread is given as i/p to third thread, etc then communication between first, second, third threads is known as interthread communication.

Real Time Implementations of ITC :-

In real world we can use ITC concept for the following implementations.

1. Development of universal protocols.
2. Development of Real world Servers.

In Java programming to develop ITC based applications by using ITC concepts we use some predefined methods which are present in java.lang.Object & these methods are known as ITC methods.

* ITC methods in java.lang.Object :-

① public final void wait (long msec) } Throws java.lang.InterruptedException
② public final void wait ()
③ public final void notify ()
④ public final void notifyAll().

Method ① is used for making the thread to wait for some amount of time in the form of milliseconds
for ex:- If first thread is completing by taking 10 msec of time, the second thread will be made wait 11 msec. But the CPU ~~burs~~ burst of threads decicled by OS when it is about to execute. Since the programmer is not able to decide the CPU burst time of the first thread, obviously it is quite impossible to make second thread to wait on the basis of the time. Hence practically this method is not advisable to use.

Method ② is used for making the thread to wait without specifying any amount of time for example:- with our programming technique we can find when the first thread completes its execution, at that point of time we can make second thread to be ready to continue its further execution. This method is quite advisible to make second thread to wait without time

Method ① & ② throws a predefined exception called java.lang.InterruptedException. (for more explanation see explanation of sleep() method related InterruptedException).

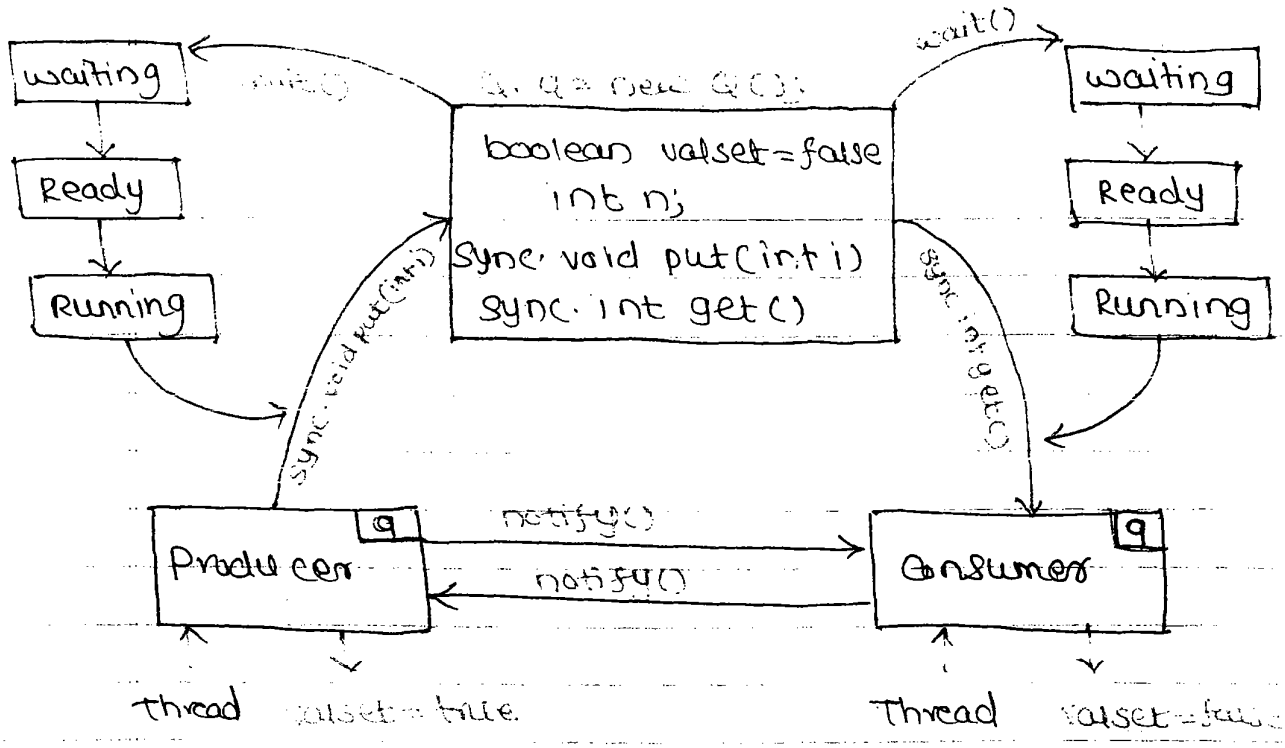
Method ③ is used for transfering one Thread at a time from waiting state to ready state

Method ④ is used for transfering all the threads from waiting state to ready state.

* The classical Examples of ITC are,
① Producer - consumer problem
② Dinning philosophers problem
③ Reader writers problem
④ Barber shop problem

① Implementation of producer-consumer Problem:-
Producer Consumer problem statement:-
Producer is one of the thread which will produce only one item which is suppose to be consumed by consumer. At any point of time producer is not allowed to produce multiple items.
Consumer is a thread which will consume only one item which is produced by producer. At any point of time ~~produ~~consumer is allowed to consume only one item but not multiple items to consume.

The following diagram gives implementation of producer consumer problem.



Q. Write a Java program which will implement producer consumer problem

```
// pcdemo.java
class Q
{
    int n;
    boolean valset = false;
    synchronized void put (int i)
    {   try
        {   if(valset)
            {
                wait ();
            }
        }
```

```
        catch(InterruptedException ie)
      { sop (" problem in thread execution");
      }

      n=i;
      SOP(" put:" +n);
      valset = true;
      notify();
   } // put()
synchronized int get()
{
   try
   { if (! valset)
      {
         wait();
      }
   } catch (InterruptedException ie)
   { sop (" problem in Thread execution"); }

   sop(" got:" +n);
   valset= false;
   notify();
   return (n);
} // get()
} // Q

class prod implements Runnable
{
   Q q;
   prod(Q q)
   { this.q = q;
```

```
        Thread t = new Thread (this);
          t.start();
      }

    public void run()
    {   int i=0;
        while (true)
        {
            q.put(++i);
        }
    }
}

Class Cons implements Runnable
{   Q q;
    Cons (Q q)
    {   this.q=q;
        Thread t = new Thread (this);
        t.start();
    }
     public void run()
     {
         while (true)
         {
             int x= q.get();
         }
     }
}

Class Pcdemo
{   psvm (-)
 {   Q q= new Q();
    Prod Po = new Prod(q); Cons co=new Cons(q);
}}
```

out and err are the objects of printStream class. err, out objects created as a static data members in system class. The out object is used for displaying the soft outputs of the java program on the console. err object is always recommended to use to display user friendly error msgs which are occuring as a part of exception handling in the catch block.

$ in is an object of InputStream class created as a static data member in system class. The purpose of in object is that it provides an environment to read data from keyboard.

* Reading Data from keyboard:-

In java programming to read the data from the keyboard we have a predefined class called Scanner which is present in java.util.* package. This class is available from jdk 1.5 version onwards.

Profile of scanner class:-

constructor

① Scanner (InputStream)

Instance methods

② public xxx nextXxx()

③ public String nextLine().

constructor ① is used for creating an object of Scanner class by passing an object of InputStream. An Object of InputStream called "in" created as a static data member in the System class

provides an env. to read the data

Example:- Scanner sc = new Scanner (System.in);

With Scanner class object one can read only one value but not possible to read multiple values.

In method ② XXX represents any fundamental data type.
method ② is used for reading any fundamental values from keyboard.

Method ③ is used for reading any type of value in the form of String type

Q. Write a java prg which will accept two numbers from the keyboard & add them

```
import java.util.Scanner;
Class DataRead
{   psvm (-)
    {   Scanner Sc= new Scanner (System.in);
        SOP("Enter first no: ");
        String s1= Sc.nextLine();
        SOP(" Enter second no: ");
        String s2= sc.nextLine();
```

```
        int res = Integer.parseInt(s1) +
                  Integer.parseInt(s2);
        sop("sum:" +res);
    }
}
```

24/9/13

* Data Conversion Techniques :-

In Java programming we have 6 data conversion techniques they are,

① Converting numerical string value into numerical or fundamental value :-

In order to convert numerical string values into numerical / fundamental values we use the following generalised method which is present in each & every wrapper class.

Wrapper class →

```
public static xxx parseXxx (String)
```

Here xxx - represents any fundamental datatype except char datatype

e.g:
```
String s1 = "100";
int x = Integer.parseInt(s1);

String s2 = "20.8";
double y = Double.parseDouble(s2);
```

② Converting numerical / fundamental values into string type value :-

In order to convert numerical / fundamental values into string type values we use the following predefined generalized overloaded static factory method. and it is present in java.lang.String class.

java.lang.String →

public static String valueOf(xxx)

Here xxx- represents any fundamental Datatype

e.g: int x = 30000;
String is = String.valueOf(x);
sop(is); → "30000"

float y = 23.87f;
String fs = String.valueOf(y);
sop(y); → "23.87"

③ converting fundamental values into Object type data :-

In order to convert fundamental values into its equivalent object type we use the following predefined generalized parameterized Constructor by taking fundamental datatype as parameter & it presents in each & every wrapper class-

Wrapper class name

Wrapper class name (xxx)

Here xxx- represents any fundamental data type.

ex: int a=10; fundamental value — Integer

Integer io = new Integer(a); — Integer(int)

Object type value

double d1 = 10.85; fundamental val — Double

Double do1 = new Double(d1); — Double(double)

Obj type val

In this example we explicitly converted fundamental values into equivalent obj type is known as boxing (In JdK 1.4 & lowers)

In JdK 1.5 version onwards we have a concept called autoboxing.

Defination of autoboxing:-
The implicit conversion of fundamental values into equivalent object type is known as autoboxing.

(4) Converting object type data into the equivalent fundamental type value:-
In order to convert object type data into equivalent fundamental type data we use

the following predefined generalized Instance method present in each & every wrapper class.

wrapper class name ↓

public xxx xxxValue ();

Here xxx represents any fundamental datatype.

ex: Character co = new Character ('A');
char ch = co.charValue ();

Double do = new Double (10.75);
double d1 = do.doubleValue ();

Double ↓
public double doubleValue()

In Jdk 1.4 we convert explicitly the object type value into fundamental type value this process is known as unboxing. In the case of Jdk 1.5 version onwards we have a concept called autounboxing feature.

Defination of autounboxing:-
The implicit conversion of object type data into equivalent fundamental datatype value is known as autounboxing.

(5) Converting string type data into object type data :-
In order to convert string type values into

equivalent wrapper class object we use the following predefined generalized parameterized Constructor by taking string as parameter which is present in each & every wrapper class.

wrapper class name →

wrapper class name (String)

exc: String s2 = "true";
Boolean bo = new Boolean (s2);

String s1 = "20.25f";
Float fo = new Float (s1);

⑥ Converting Object type data into String Type data:

In order to convert object type data into String type data we use following predefined instance method present in each & every wrapper class.

wrapper class Name →

public String toString()

ex:- Integer io = new Integers (100);
String is = io.toString();

ex: String s1 = "100";
Integer io = new Integer(s1);
int i = io.intValue();
String s = String.valueOf(io);
int x = Integer.parseInt(s);
Integer io = new Integer(x);
String s1 = io1.toString();

* Design Patterns :-

Defn:
Design patterns are the best rules developed and proved by industry experts and released to the real industry for eliminating the recurring problems which are occurring in s/w development.

In s/w development we have somany number of design patterns. But as a part of JSE we have two types of design patterns they are,

1. Factory method
2. Singleton class.

25/9/13

* Collection Framework :-

Collection framework is one of the additional service | add-on service developed by SUN MS & supplied as a part of java slw for enhancing the performance of java appl^ns | projects.

Defination:- A collection flw is one of the standardized mechanism of java & it allows us to group ~~diff~~ multiple values either of same type or different type or both types in a single variable with dynamic size in nature. This single variable is known as collection flw variable.

Q. What are the differences between Arrays concept & collection framework ?

Array	Collection framework.
An array is a collective name given to a group of consegative memory location which are all refered by similar type of values.	It is one of the standar dized mechanism
concept of array allows to organize only homogene- ous data but not helerogeiveus	collection flw concept allow us to organize data both homogeneous & heterogeneous values.

Arrays concept contains	Collection flw contains
fixed size in nature	dynamic size in nature

* Goals / Properties / Aims of collection flw :-

1) Collection flw provides high performance to the java appln's

2) Collection flw always provides extendability (If the size of collection flw variable is increasing depending on incoming flow of data then collection flw variable is containing extendability property).

3) Collection flw provides adoptability. (Adding contents of one collection flw variable to another collection flw variable either in the beginning or in the ending or in the middle is known as adoptability).

4) Collection flw is one of algorithmic oriented.
   i) collection flw provides inbuilt sorting techniques.
   ii) Collection flw provides inbuilt Searching techniques.
   iii) Collection flw provides inbuilt priliminary concepts of data structures.

Q. When array of objects of java.lang.Object organizing all types of values, what is the need of using collection flw?

Ans: Array of objects of java.lang.Object allows us to organise the multiple values either of same type or different type or both the types with fixed size in nature & it is unable to provide Dynamic size in nature.

To get dynamic size in nature by holding multiple values either of same type or different type or both the types, we use the concept of collection framework.

26/9/13

* Types of Collection frameworks:-

In the initial days of SUN ms the concept of collection flw was known as data structures. In those days data structures concept was able to fulfill few requirements of the industry & unable to fulfill few more requirements of the industry. Hence to fulfill all the requirements of the industry SUN developers has performed re-engineering or revised operation on data structures & releasing to the real Industry on the name of "new collection flw". The existing data structures carry forwarded on the name of "leagacy collection flw".

In other words collection flw's are classified into two types they are,

1. New Collection flw.
2. Legacy Collection flw.

1. New collection flw :-

Re-Engineering | Revised form of Data structure is known as New collection flw.

New Collection flw is classified into two types They are,

1. 1D | Single Collection flw.
2. 2D | Double Collection flw | Maps.

1. 1D | Single collection flw :-

In 1D Collection flw data is organizing in the form of either rows or columns but not in the form of (key, value) pair or but not in the form of rows & columns.

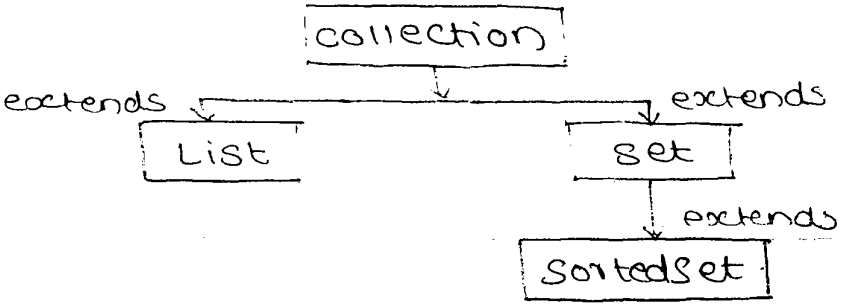
In order to deal with 1D collection flw programming we must learn coresponding interfaces & classes which are present in java.util.* package.

1D Collection flw interfaces :-

1D collection flw interfaces are classified into 4 types they are, ...

1. java.util.Collection
2. java.util.List
3. java.util.Set
4. java.util.SortedSet.

Hierarchy Charts-



The objects of collection, List, Set & SortedSet interfaces are called collection fw variables.

Differences betn collection, List, Set, SortedSet

~~Collection~~	~~List~~	~~Set~~	~~SortedSet~~
Collection	List	set	Sortedset
Collection is one of the predefined Interface available on the top of all JD collection fw interfaces	List is the subinterface of collection	set is a subinterface of the collection	Sorted set is a subinterface of set
An obj of collection interface allows us to add duplicate values	An obj of list allows us to add duplicate values	An obj of set Interface allows us to add unique/ distinct elements	An obj of sorted-set Interface allows us to add unique elements.
Collection Interface obj always displays in Random	list Interface obj always displays in the sequential	Set interface obj always displays In Random	SortedSet Interface obj always displays in Sorted order

An obj of the Collection(I) allows us to add data only at end	An obj of the List(I) allows us to organize data either in beginning or in middle or in the ending	An obj of the Set(I) allows us to organize the data only in the ending	An obj of the SortedSet(I) allows us to organize the data either in the beginning or in the ending or in the middle
An obj of the Collection(I) allows us to retrieve the data only in forward direction but not in other directions	An obj of List(I) allows us to retrieve the data either in forward or in backward or in both directions and in Random directions	An obj of Set(I) allows us to retrieve the data only in the forward direction but not in other directions	An obj of the SortedSet(I) allows us to retrieve the data in forward, backward & random direction

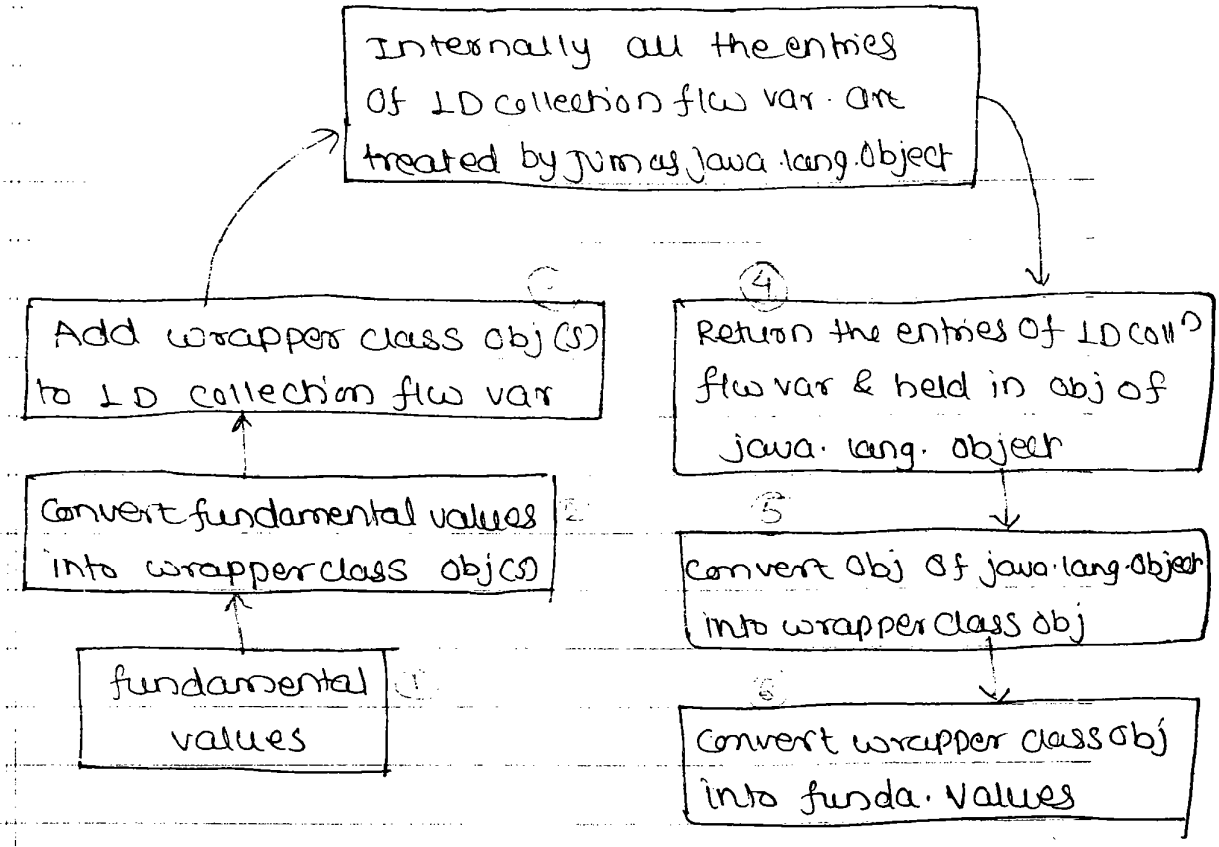
The objects of Collection & Set's are known as uni~~re~~-directional objs bcoz they are allowing us to retrieve the data only in the forward direction whereas the objects of List & sorted set are called multidirectional objects bcoz They allows us to retrieve the data in all the directions.

1D collection flw process contains two phases they are,

① Grouping / Assembling phase

② Ungrouping / Deassembling phase.

The following diagram gives sequence of steps for both grouping & ungrouping phases.



In the above diagram steps ①, ② ③ are used for grouping phase and steps ④ ⑤ ⑥ are used for performing ungrouping phase.

27/9/13

* Object Type Casting :-

The process of converting base class object reference into derived class obj reference is known as Object type casting.

Syntax :-

```
subclassname subobj = (subclassname) baseclass obj ;
```

Need of object Type casting :-

① The result of the java program is available in the object of java.lang.Object

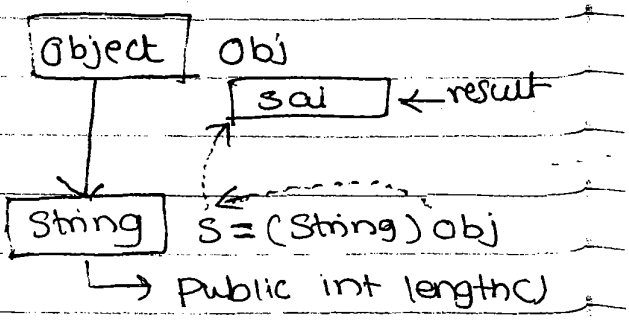
② To process the result of the obj of java.lang.Object we are not having any methods present in java.lang.Object class.

③ To process the result of object of the java.lang.object, we have the special methods present in the subclass of java.lang.Object class.

④ To process the result of obj of java.lang.Object class, with respect to obj of java.lang.object we can not access special methods of the subclass of java.lang.Object class.

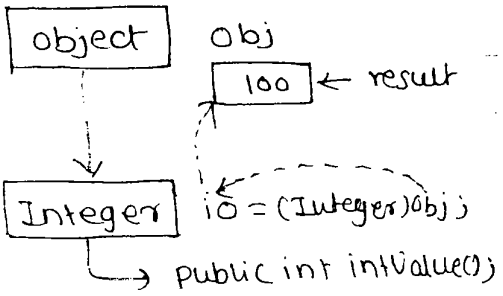
⑤ To process the result of obj of java.lang.Object wrt the special methods of its derived class we must use the concept of Object type casting

```
e.g: Object obj = "sai";
int len = obj.length(); //error
String s=(String) obj; //object type
int len = s.length(); casting
  Sop (len);
```



e.g ②

```
Object obj = 100;
int n = obj.intValue(); //error
Integer s = (Integer) obj;
int n = s.intValue();
sop(n);
```



Example for 1D Collection flw process:-

```
int a = 10;
Integer io = new Integer(10);
c.add(io);
```

c → 10 → obj of java.lang.Object

Unwrapping

```
Object obj = c.get(0);
Integer io = (Integer) obj; //object Type Casting
int x = io.intValue();
```

✱ Methods in java.util.Collection:-

① public int size()
This method is used for finding the number of entries present in any collection flw variable

② public boolean isEmpty().
This method returns true provided collection flw variable is empty (size=0). This method returns false provided collection flw variable is not empty (size > 0)

③ public ~~int~~ boolean add(Object)
This method is used for adding the values to any collection flw variables.

This method returns true provided as long as we call this method wrt Collection List Interface values by adding unique & duplicate values. It also returns true as long as we call this method wrt set & SortedSet by adding unique values.

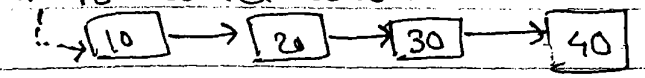
(4) public boolean contains (Object)
This method returns false wrt second SortedSet by adding duplicate values. returns true provided the specified value present in the Collection fw variable otherwise it returns false.

(5) pubiic Iterator iterator() :-
This method is used for retrieving the data from any collection fw variable and forms like a forward direction al Chain. This method returns an object of Iterator (I) interface & & it is by default pointing just before the first element of the chain

ex: sop(c); [10,20,30,40]

```
Iterator itr = c.iterator();
```

itr forward direction




```
int s=0;
while [ itr.hasNext())
{   Object obj = itr.next();
```

```
    Integer io = (Integer) obj;
    int x = io.intValue();
        sop(x);
        s = s+x;
    }
    sop("sum:"+s);
```

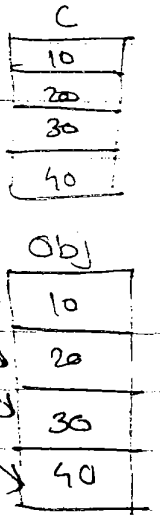
28/9/13

⑥ public Object[] toArray() :-

This method is used for extracting the data from any collection fld variable and return an array of objects of java.lang.Object class

e.g:

```
sop(c);
Object obj[] = c.toArray(); // 4 step
    int s=0;
for(object x: obj)
{
Integer io=(Integer)x; //step5
int x1= io.intValue(); //step6
   sop(x1);
    s=s+x1;
}
sop("sum="+s); //100
```



Iterator Interface :-

It is one of the predefined interface present in java.util.* package.

The purpose of Iterator interface object is to extract the data from any collection fld variable only in forward direction

By default an object of Iterator interface is pointing just before the first element of forward directional chain.

Methods:-

① public boolean hasNext()
② public Object next().

Method ① returns true provided collection flw variable is having next element otherwise it returns false.

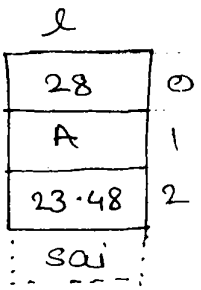
Method ② is used for obtaining next element of any collection flw variable provided method ① returns true.

* **Methods in java.util.List :-**
We know that List is the subInterface of collection sothat all the methods of collection are inherited into List & List Interface contains following special methods

① public void add(int, Object) :-
This method is used for adding an element to the list interface object either in the beginning or in the endding or in the middle

Q. How do you add an element "sai" at the end of existing elements by using add(int, object) method ?

l.add(l.size(), "sai");



After the above stmt is executed we have,

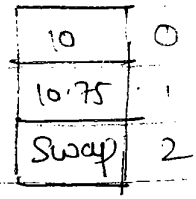
l.add(3, "sai");

② public object get(int) :-

This method is used for obtaining the values of any collection fw variable by passing valid existing position. If the position is Invalid we get a predefined exception called java.lang.IndexOutOfBounds Exception

valid position

exc: Object obj1 = l.get(1);



Invalid position

Object obj2 = l.get(8);

null ⟶ throws IndexOutOfBounds exception

③ public void remove(int)

④ public void remove(Object) :-

The above methods are used for removing the elements of List Interface obj either based on position or based on contents

⑤ public void removeAll() :-

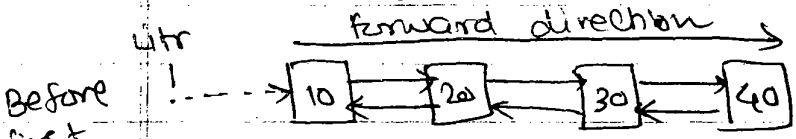
The above method is used for removing all the elements of List interface object

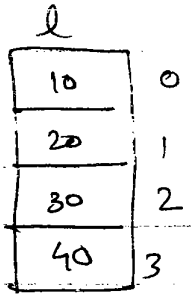
⑥ public ListIterator listIterator():-

This method is used for retrieving the data from any collection flw variable and forms like a Bi-directional chain. This method returns an object of ListIterator(I) Object. & it is by default pointing just before the first element of Bidirectional chain.

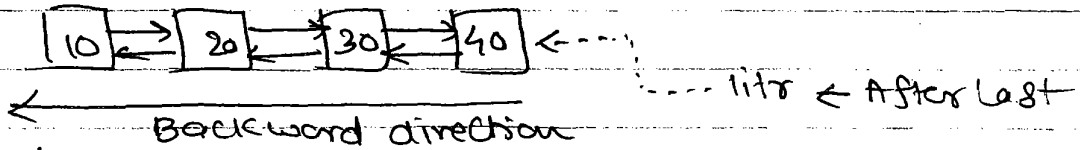
ex:-

```
sop(l);                    // [10, 20, 30, 40]
ListIterator litr = l.listIterator();
```






```
while(litr.hasNext())
{
  Object obj = litr.next();
  sop(obj);                // 10 20 30 40
}
```




```
int s=0;
while(litr.hasPrevious())
{
Object obj = litr.previous();   // 4th step
Integer io = (Integer) obj;     // 5th step
 int x = io.intValue();
  sop(x);                  // 40 30 20 10
  s=s+x;
}
sop("Sum is:" +s);         // 100
```

* ListIterator Interface :-

It is one of the SubInterface of Iterator Interface & it is present in java.util.* pkg

The purpose of ListIterator (I) obj is to extract the data from any collection fw variable in both forward & backward directions.

Methods in ListIterator :-

1. public boolean hasNext () } Inheritted from Iterator ()
2. public Object next ()
3. public boolean hasPrevious ()
4. public Object previous ().

Method ③ returns true provided ListIterator(I) obj is having previous element in the bidirectional chain otherwise it returns false

Method ④ is used for retrieving previous element of bidirectional chain wrt an Object of ListIterator (I) provided method ③ returns true

Q. What is the difference between Iterator (I) & ListIterator (I).

An object of Iterator (I) is always used for retrieving data in forward direction from any collection variable while an object of ListIterator (I) obj is used for retrieving the data from any collection fw variable in both forward & backward directions

Q. What are the methods used for retrieving the data from any collection flw variable in collection flw concept ?

① public Iterator iterator()
② public Object toArray()
③ public Object get(int)
④ public ListIterator listIterator().

* Methods in java.util.Set :-

We know that set is the sub Interface of collection sothat all the methods of collection are inheritted into Set (I)

" Set Interface does not contains any special method except collection Interface methods ".

Even though methods of collection & set interfaces are same, collection methods are defined in some predefined class in such away that duplicates are allowed. whereas set interface methods are defined in some other predefined class in such away that unique elements are allowed. "

In other words even though the methods of collection & set are same, their implementations (definations) are different in different predefined classes.

29/9/13

* Methods in java.util.SortedSet :-

We know that SortedSet is subinterface of Set so that all the methods of set are inherited into SortedSet (indirectly they are methods of Collection).

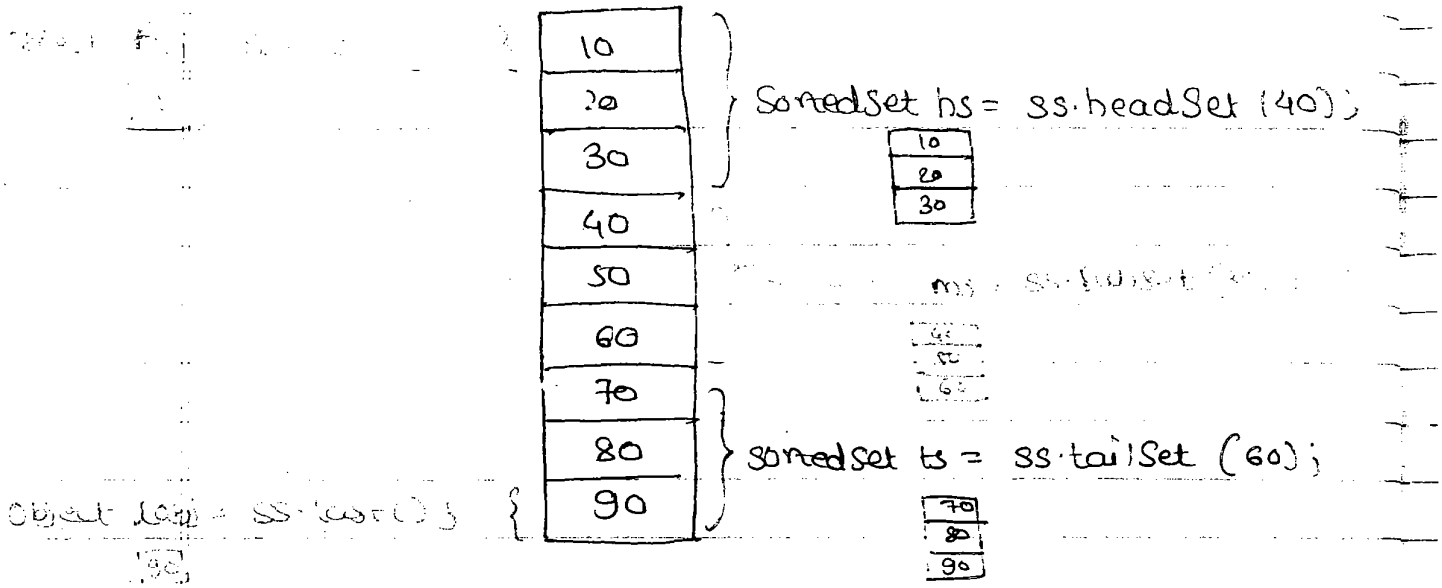
Even though methods of Set & SortedSet are same, their implementations are different in different predefined classes in such away that by satisfying properties of Set & SortedSet.

As a part of jdk1.5 we have the following special methods in SortedSet.

① public Object first()
② public Object last()
③ public SortedSet headSet (Object obj)
④ → x < obj
⑤ public SortedSet tailSet (Object obj)
→ x > obj
⑥ public SortedSet subset (Object obj1, Object obj2
→ obj1 < x < obj2

Methods ① & ② are use for retrieving first & last elements of SortedSet. method ③ is used for retrieving those elements (xi) which are less than target object. (mathematically xi < obj) Method ⑤ is used for retrieving those values (xi) which are greater than target obj obj1 & less than target obj obj2 mathematically obj1 < xi < obj2

Method ④ is used for retrieving those values (xi) which are greater than target object obj mathematically xi > obj



All the above methods are used for splitting the elements of SortedSet into different parts.

* LD Collection flw Classes :-

LD Collection flw Classes Contains defination for those abstract methods which are inherited from LD Collection flw interfaces. The following table gives LD collection flw Interfacename, Corresponding LD collection flw Classname & their hierarchy.

LD Collection flw Interface Name	LD Collection flw Class name	Hierarchy
Collection	① AbstractCollection	← impl Collection
List	① AbstractList	← ext AbstractCollection, ← impl List

i) Set ③ AbstractSet ← ext AbstractCollection
← impl Set

SortedSet

noclass

④ AbstractSequentialList ← ext AbstractList

⑤ LinkedList ← ext
⑥ ArrayList ← ext
} AbstractSequentialList

⑦ HashSet ← ext AbstractSet

⑧ TreeSet ← ext AbstractSet.
← impl SortedSet

In the above hierarchy chart the classes ① ② ③ ④ are predefined JD collection fw abstract classes & they are providing common reusable features & these classes are resolved by Sun ms for the future enhancements in JD Collection fw programming.

The abstract classes ① ② ③ ④ are acting as a middleware layer between JD collection fw interfaces and JD Collection fw concrete classes

The classes ⑤ ⑥ ⑦ ⑧ are treated as JD Collection fw concrete classes & we use them in the real industry.

LinkedList :-

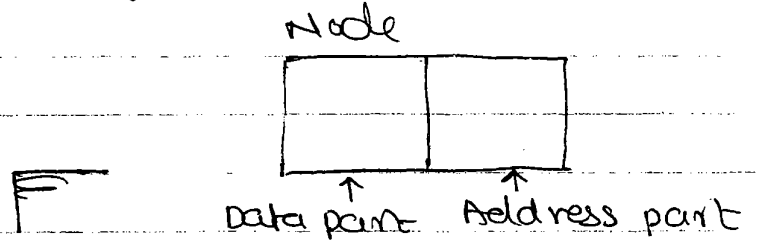
LinkedList is one of the predefined class which is used for implementing the concept of LinkedList. Creating a linked list is nothing but creating an object of LinkedList.

ex: LinkedList ll = new LinkedList();

Here ll is an object ~~object~~ of LinkedList and it is treated as Collection flw variable

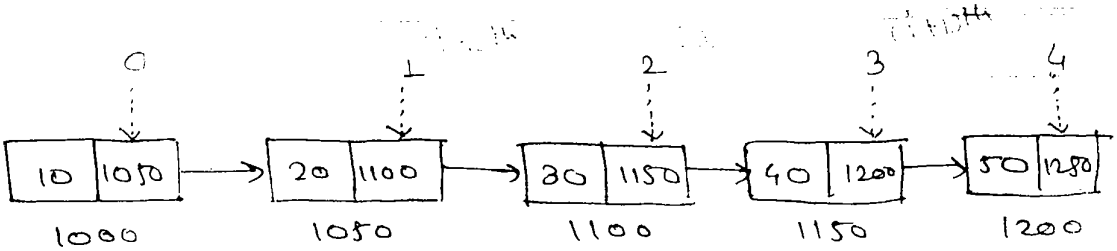
In LinkedList the data is organizing in the form of nodes.

Node contains two parts & whose structure is given below,



Here node represents name of LinkedList class object. Data part represents type of data being stored in the LinkedList. Address part represents type of address ~~being~~ stored in other words it always contains address of next node & address of last node contains null which indicates end of linked list

ex Store the values 10, 20, 30, 40, 50 in linked list



principle:- The data organized in nodes in such away that the address of i^{th} node stored in address part of $(i-1)^{th}$ node, address part of i^{th} node contains address of $(i+1)^{th}$ node & address part of last node contains "null" which indicates end of the linked list.

30/9/13

Advantages of LinkedList:-

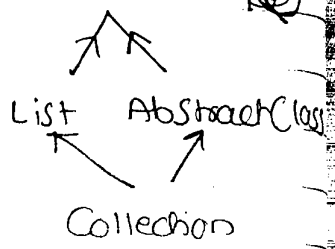
LinkedList class object allowing us to place multiple values either of same type or different type or both the types

Linkedlist class allocoing dynamic size in nature.

Linkedlist allows us to perform dynamic insertions that is in linked list one can insert an element either in the beginning or in the ending or in the middle

The hierarchy of linkedlist class is shown below,

LinkedList ← AbstractSequentialList ← AbstractList



* Profile of java.util.LinkedList class :-

constructors

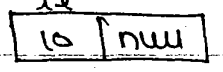
① LinkedList()
② LinkedList(int)

Instance methods

③ public void addfirst(Object)
④ public void addLast(Object).
⑤ public Object getfirst().
⑥ public Object getLast()
⑦ public object removefirst()
⑧ public object removeLast().

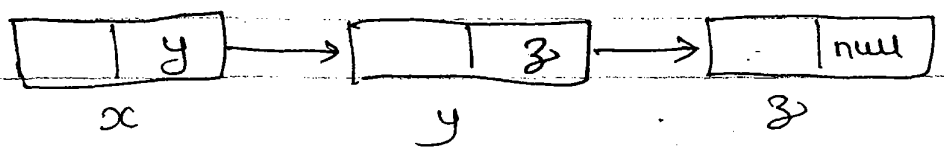
Constructor ① is used for creating an object of LinkedList without specifying no. of nodes to be created.

e.g: LinkedList ll = new LinkedList();
ll.add(10);



Constructor ② is used for creating an object of LinkedList by specifying the number of nodes to be created initially.

LinkedList ll = new LinkedList(3);



method ③ & ④ are used for inserting the elements in the LinkedList at first & last positions respectively.

e.g: ① ll.addFirst(10);

-: or :-

ll.add(0,10);

② ll.addLast(50);

-: or :-

ll.add(ll.size(), 50);

method ⑤ & ⑥ are used for retrieving first & last elements of LinkedList respectively

e.g: ① Object fobj = ll.getFirst();

-: or :-

Object fobj = ll.get(0);

② Object lobj = ll.getLast(0);

-: or :-

Object lobj = ll.get(ll.size()-1);

method ⑦ & ⑧ are used for removing first & last element of LinkedList respectively

e.g: ① Object fobj = ll.removeFirst();

-: or :-

Object fobj = ll.remove(0);

② Object lobj = ll.~~ge~~removeLast();

—:OR:-

object lobj = ll.remove (ll.size()-1);

Q. Write a java prg which illustrate the concept of linked list

```
import java.util.*;
Class LL
{
    psvm (-)
    {
        LinkedList ll = new LinkedList();
        Sop( " Content of ll:" + ll);
        Sop ( "Size of ll: " + ll.size());

        ll.add (20);
        ll. add (30);
        ll add (40);
        Sop( " Content of ll : " + ll);
        Sop(" Size of ll:" + ll.size());

        ll.addfirst (10);
        ll. addLast (50);
        sop (" Content of ll:" + ll);
        sop ( " Size of ll:" + ll.size());

        sop(" extract data from ll:iteratator() ");
        int s=0;
        Iterator itr = ll.iterator();
```

```
while (itr.hasNext())
{
    Object obj = itr.next();
    Integer io = (Integer) obj;
    int x = io.intValue();
    SnP(x);
    S = S+x;
}
SOP("sum using iterator() = "+S);
SOP("extract data from ll: ListIterator FWD");
ListIterator litr = ll.listIterator();
while (litr.hasNext())
{
    Object obj = litr.next();
    SOP(obj);
}
SOP("extract data from ll: ListIterator() BWD");
while (litr.hasPrevious())
{
    Object obj = litr.previous();
    SOP(obj);
}
SOP("extract data from ll: toArray()");
int S1=0;
Object obj1[] = ll.toArray();
for (Object x : obj1)  //enhanced model for loop
{
    Integer io = (Integer)x;
    int y = io.intValue();
    SOP(y);
    S1=S1+y;
}
```

```
        sop("Sum using toArray(): "+s1);
        sop("extract the data from ll: get()");
        for(int i=0; i<ll.size(); i++)
        {
          Object obj = ll.get(i);
          sop(obj);
        }
    // extract the random element
      Object robj = ll.get(2);
       sop("random element:" +robj); //30
  }
}
```

Q. Write a java prg which will determine biggest of 3 numbers & which will return 3 source values & biggest number?

```
// Big.java
package bp;
import java.util.LinkedList;
public class Big
{
    int a,b,c;
    public void Set (int a, int b, int c)
    {
       this.a=a;
       this.b=b;
       this.c=c;
    }
    public LinkedList findBig()
    {   int big=a;
```

```
if (b > big)
{   big = b; }
if (c > big)
{   big = c; }

LinkedList ll = new LinkedList();
ll.add("val of a = " + a);
ll.add("val of b = " + b);
ll.add("val of c = " + c);
ll.add("Biggest = " + big);

return (ll);
}
} // Big (Business Logic Class)
```

BP
→ Big.class

```
// BigDemo.java
import bp.Big;
import java.util.*;
class BigDemo
{
    psvm(-)
    {
        Scanner sc = new Scanner(System.in);
        sop("Enter first no:");
        int n1 = Integer.parseInt(sc.nextLine());
        sop("Enter second no:");
        int n2 = Integer.parseInt(sc.nextLine());
        sop("Enter third no:");
        int n3 = Integer.parseInt(sc.nextLine());
```

```
Big bo = new Big();
bo.set(n1, n2, n3);

LinkedList ll = bo.findBig();

Iterator itr = ll.iterator();
while (itr.hasNext())
{
    Object obj = itr.next();
    Sop(obj);
}
}
}
```

NOTE:- for the above prg if we input any float values then we get NumberFormatException & the entire block of stmts in main() method must be enclosed within try & catch blocks by writing NumberFormatException.

* Limitations of java util LinkedList :-

LinkedList will take more memory space (explicitly memory space for data part & address part of the node is created in Heap memory and takes double amount of space)

~~Rete~~ Retrieving the data from LinkedList will take more time (slow. On overall the prformance of LinkedList is less & Thus my is not recommended to use LinkedList

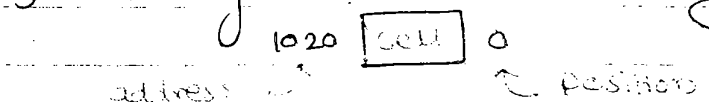
Hence to avoid the above problems we have an another ~~one~~ Collection called ArrayList.

* Java.util.ArrayList :-

ArrayList is one of the bottommost 1D collection flw concrete subclass.

Creating an Arraylist is nothing but creating an object of Arraylist class.

e.g: ArrayList al = new ArrayList();



Here al is one of the collection flw variable

In ArrayList the data is organizing in the form of cells. cell values are stored in Heap memory & cell addresses are stored in Associative memory. cell values can be processed by making use of positions. (If we store anything in associative memory Then it takes negligible amount of space and it is always recommended to store fixed values or figurative constants or jvm generated values. If we retrieve anything from associative memory then it takes negligible amount of time).

* Advantages of ArrayList over LinkedList :-

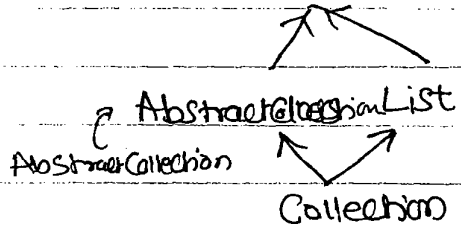
① ArrayList related applications takes less memory space (Because cell addresses are storing in associative memory

② Retrieving the data from ArrayList takes negligible amount of time. that is fast in retrieval.

③ On overall the performance of ArrayList class based applications are enhanced.

Hierarchy of ArrayList :-

ArrayList ← AbstractSequentialList ← AbstractList



Q. Write a java prg which illustrate the concept of ArrayList?

1) Copy LL.java of pg: 180 into new window & save it as AL.java
2) Substitute ArrayList in place of LinkedList
3) Substitute ll.add(0, 10) in place of ll.addFirst(10);
4) Substitute ll.add(ll.size(), 50) instead ll.addLast(50);

In general whereever we come across LinkedList class it is highly recommended to substitute ArrayList instead of Linkedlist.

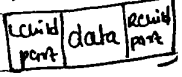
* Profile of java.util.ArrayList:-

constructor

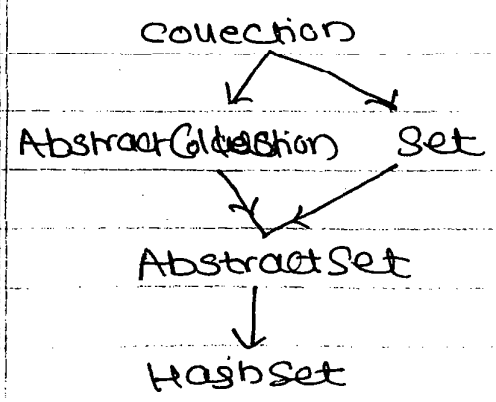
① ArrayList(): This constructor is used for creating an Object of ArrayList class.

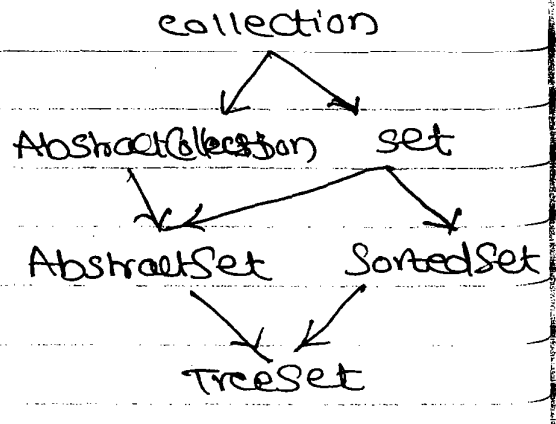
e.g: ArrayList al = new ArrayList();

Here HashSet & TreeSet are the two bottommost predefined concrete subclasses in 1D collection fw

HashSet	TreeSet
Hashset class obj ~~never~~ allows unique elements but not duplicates	Tree set class obj also allows unique elements but not duplicates
Hashset class obj allows us to organize the data in the form of Hashtable by following hashing mechanism	TreeSet class obj organizes the data in the form of nodes ( by following Binary trees concept.
We can not determine in which order Hashset class obj displays data bcoz SUN ms developers did not disclosed which hashing mechanism is followed at the time of developing HashSet class.	TreeSet class obj alway displays the data in sorted order.

In HashSet the operations like Insertion, Deletion & modification are considered as time consuming operations	In TreeSet operations like Insertion, Deletion & modification takes less amount of time that is less time consuming operations.
Retrieving data from the HashSet will take considerable amount of time that is Retrival time is more/ slow	Retrieving the data from TreeSet will take negligible amount of time. i.e. Retrival time is less/ fast.
Creating HashSet obj is nothing but creating an object of Hashset class HashSet hs = new HashSet();	Creating TreeSet is nothing but creating an obj Of TreeSet class TreeSet ts = new TreeSet();
Profile :- HashSet()	profile :- Treeset().





By observing the above characteristics Treeset is highly Recommended to use compared to Hashset.

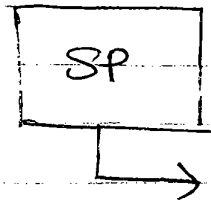
Q. Write a Java program which illustrate the concept of Treeset & Hashset ?

```
import java.util.*;
class hsts
{
   psvm (-)
   {
      Treeset ts = new Treeset ();
      sop ("content of ts :" + ts);
      sop (" size of ts: " + ts.size());
   // add data to ts.
      ts.add(100);
      ts.add (1);
      ts.add (10);
      ts.add (90);
      ts.add (90); // duplicates are not allowed
                      to add at Runtime
      sop("content of ts :" +ts);
      sop(" size of ts:" + ts.size());

      Iterator itr = ts.iterator();
      while (itr.hasNext())
      {
         Object obj = itr.next();
         sop (obj);
      }
   }
```

Q. Write a java prg which will sort the list of numbers by accepting list of numbers from keyboard. Print those numbers in both ~~asen~~ ascending and descending order & also print highest number & ~~list~~ least number

```
// sort java
package SP;
import java.util.*;
public class Sort
{
   public Treeset getSortedData()
    {
        TreeSet ts = new TreeSet();
        Scanner sc = new Scanner(System.in);
       Sop(" Enter How many no you want to enter");
      int non = Integer.parseInt(sc.nextline());
      for( int i=1; i<=non; i++)
      {
         Sop("Enter "+i+" Number");
        int n = Integer.parseInt(sc.nextline());
          ts.add();
      }
      return ts;
    }
}
```

(cmd) javac -d . sort.java



```
//SortDemo.java

import java.util.*;
import sp.Sort;

class SortDemo
{
psvm(-)
{
    Sort so = new Sort();
    TreeSet ts = so.getSortedData();

    SOP(" Ascending order:");
    Object obj[] = ts.toArray();
     for (Object x:obj)
     {
        SOP(x);
     }
    SOP (" Descending order:");
    for( int i = ts.size()-1; i>=0; i--)
    {
      SOP (obj[i]);
    }
    SOP(" Highest no:" + obj[ts.size()-1]);
    SOP (" Lowest no:" + obj[0]);
}

}
```

4/10/13

* 2D Collection framework :-

In 2D collection fw the data is organizing in the form of key, value pair.
In key, value pair the values of key must be unique & the values of value may or may not be unique that is duplicates are allowed.
ex:-

Deposit

key	acno	bal value
	10	1.5
	20	2.5
	30	3.5

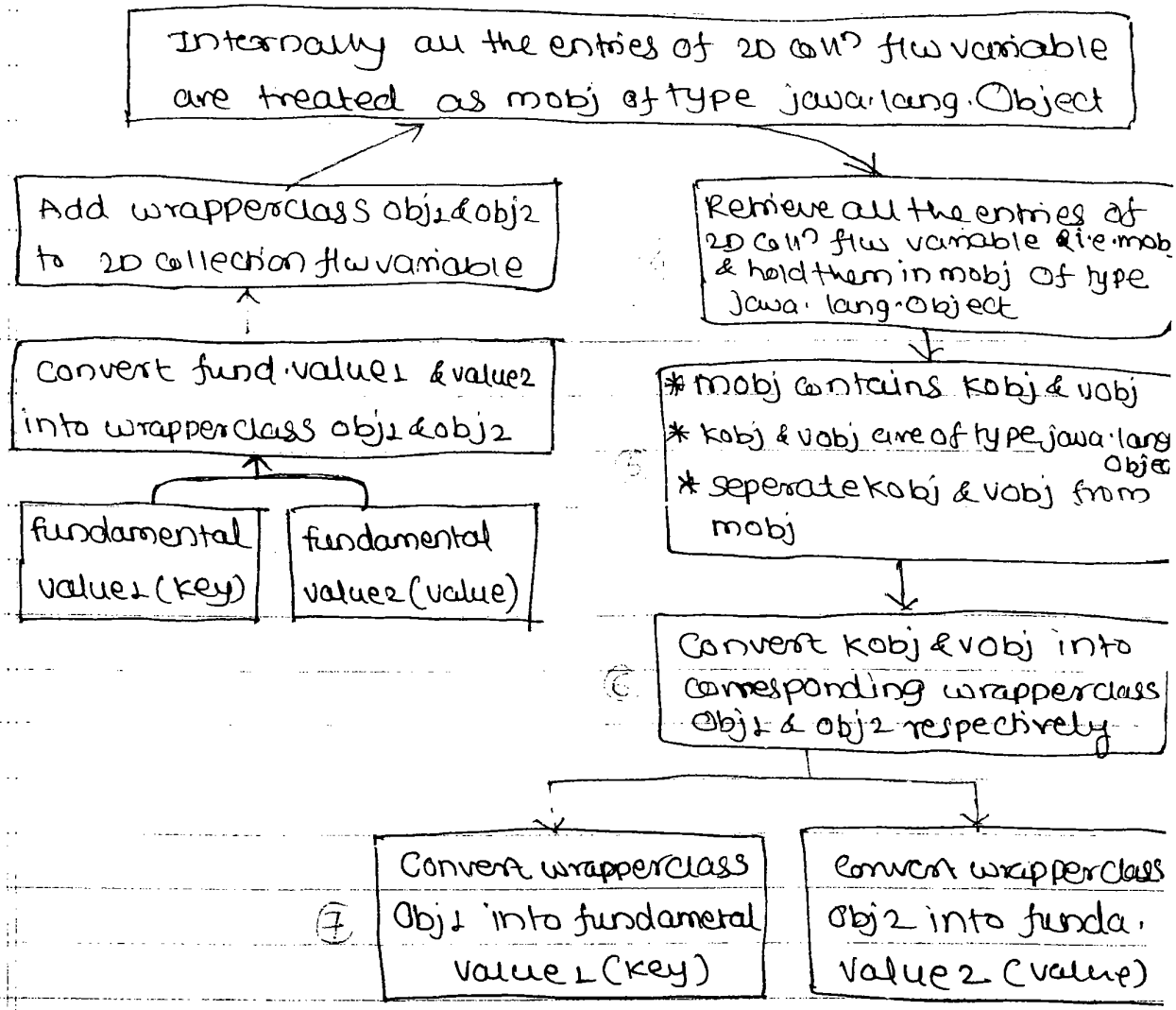
Deposit → 2D coll° fw var
acno → name of key
bal → name of value
(10,20,30) → values of key
(1.5,2.5,3.5) → values of value

* 2D collection fw process :-

2D collection fw process contains two phases they are,
① Grouping phase / Assembling phase.
② Un-grouping phase / De-assembling phase.

The following Diagram gives sequence of steps for grouping phase & ungrouping phase.

In the diagram steps ① ② ③ are used for performing grouping phase. The steps ④ ⑤ ⑥ ⑦ are used for ungrouping phase



In order to deal with 2D collection flw programming we must learn 2D collection flw classes, Interfaces which are present in java.util.*;

* 2D collection flw Interfaces / Map Interface

Map interfaces are classified into 3 types they are,

1. java.util.Map
2. java.util.Map.Entry
3. java.util.SortedMap.

① java.util.Map :-

- It is available on the top of hierarchy of all 2D collection flw interfaces.
- An object of Map interface allows us to organize the data in the form of (key, value) pair. In (key, value) pair the values of key represents unique & values of value may or may not be unique.
- Map interface object displays the data in random order that is in whichever order we add in the same order the data will be displayed.

5/10/13

Profile of java.util.Map Interface :-

① Public int size() :-
This method is used for finding the number of entries present in 2D collection flw variable

② Public boolean isEmpty() :-
This method returns true provided 2D collection flw variable is empty (size = 0). It returns false in the case of non-empty. (size > 0)

③ public void put(Object, Object) :-
This method is used for performing both insertion & updating the values of 2D collection flw variable.

If we are placing any (key, value) pair data in 2D collection flw variable & if the value

of key is not existing in 2D collection flw variable then (key, value) pair will be considered as inserted entry.

If the value of key is already existing in 2D collection flw variable then existing value of value is modified/ ~~edit~~ replaced with new value of value

e.g:
```
m.put (10, 1.5);
m.put (20, 1.5);
m.put (30, 2.5);
```

m → obj of Map(I)

Inserted entries

10	1.5
20	1.5
30	2.5

```
SOP(m);
m.put (10, 9.5);
SOP(m);
// {10=9.5, 20=1.5, 30=2.5}
```

m →

modified entry

10	9.5
20	1.5
30	2.5

④ public Object get(Object):-

This method is used for obtaining value of value by passing value of key.

e.g:
```
Object vobj = m.get (10);
```
g.5

```
Object vobj = m.get (88);
```
null.

⑤ Public Set entrySet():-

This method is used for retrieving all the entries of any 2D collection fw variable. this method returns an object of set. An object of set is holding all the entries of 2D collection fw variable which are returned by entrySet() method

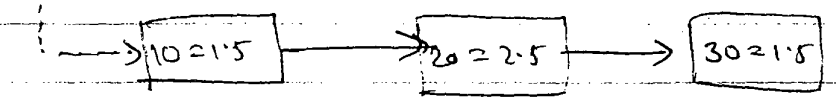
e.g.

```
SOP(m);
```

m

10	1.5
20	2.5
30	1.5

```
Set s = m.entrySet();
Iterator itr = s.iterator();
```



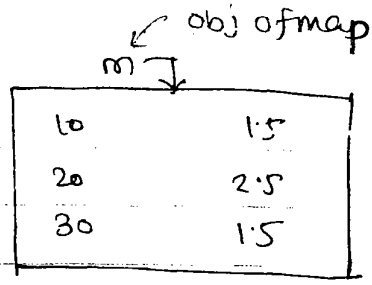

```
while (itr.hasNext())
{
  Object mobj = itr.next();   // 4-step
                10=1.5
  Map.Entry me = (Map.Entry) mobj;
  Object kobj = me.getKey();      } 5-step
  Object vobj = me.getValue();
  Integer io = (Integer) kobj;    } 6-step
  Float fo = (Float) vobj;
  int acno = io.intValue();       } 7-step
  Float bal = fo.floatValue();
  SOP(bal + " is balance of:" + acno);
}
```

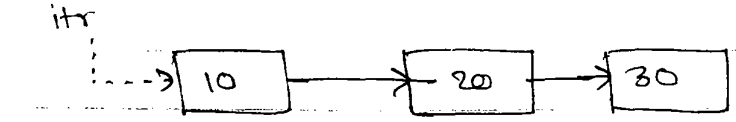
⑥ public Set keySet() :-

This method is used for obtaining set of keys and holding in the object of java.util.Set(I

e.g:

```
Sop (m);

Set S = m.keySet();

Iterator itr = S.iterator();

while (itr.hasNext())
{
   Object kobj = itr.next();
   Object vobj = m.get(kobj);
   Integer io = (Integer) kobj;
   Float fo = (float) vobj;
   int acno = io.intValue();
   float bal = fo.floatValue();
   SOP( bal+ " is a bal of" +acno);
}
```





7/10/13

② java.util.Map.Entry :-

Map is one of the interface & Entry is one of the nested / inner interface defined in Map interface.

Map.Entry is used for seperating (key, value) pair from any 2D collection flw variable.

* Methods in Entry Interface :-

① public Object getKey()
② public Object getValue().

These two methods are used for seperating (key, value) pair from any 2D collection flw variable.

③ java.util.SortedMap :-

It is one of the sub interface of Map interface so that all the methods of Map are inheritted into SortedMap.

SortedMap organizes the data in the form of (key, value) pair.

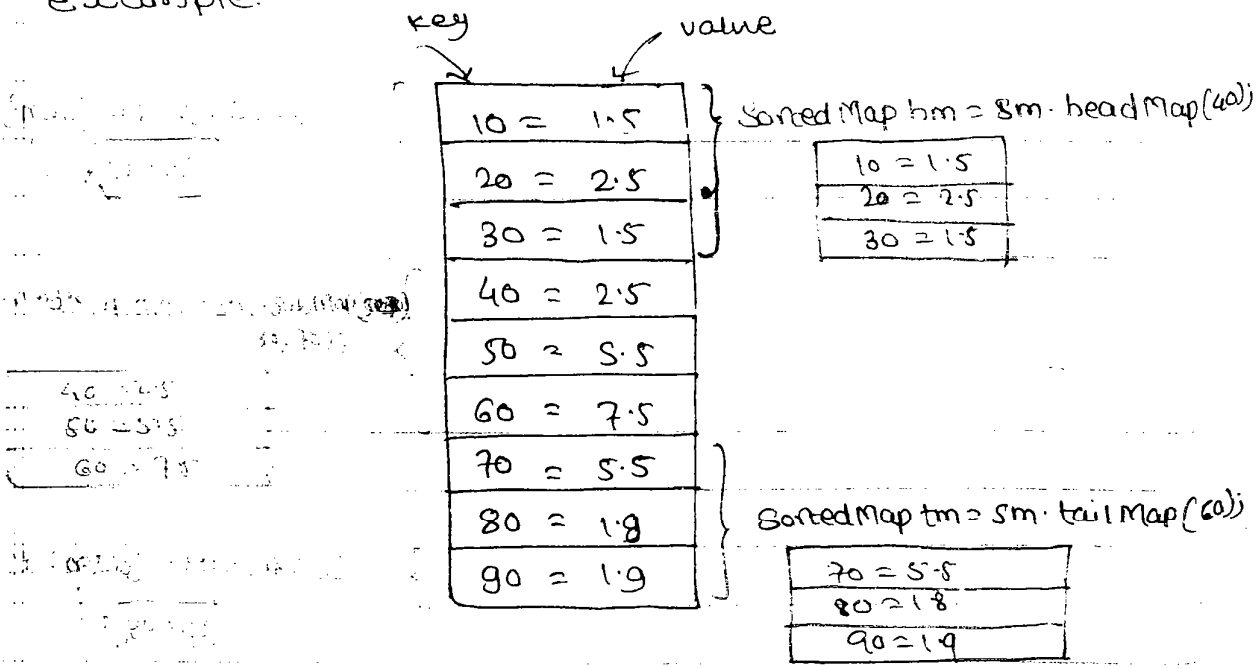
SortedMap interface object displays the data in the Sorted order & sorting will be done based on key

Profile of java.util.SortedMap :-

① public Object first()
② public Object last()
③ public SortedMap headMap(Object kobj)
↳ $x_i < kobj$
④ public SortedMap tailMap(Object kobj)
↳ $x_i > kobj$
⑤ public SortedMap subMap(Object kobj1, Object kobj2)
↳ $kobj_1 < x_i < kobj_2$

For more explanation of above methods see the methods of SortedSet interfaces pg: 173.

Example:-



* 2D Collection flw Classes :-

2D collection flw classes contains defination for those abstract methods which are inherited from 2D collection flw interfaces. The following table gives 2D collection flw interface names, corresponding 2D collection flw classname & their hierarchy.

2D C. flw Interfacename	2D Collection flw Classname	Hierarchy
① Map	① AbstractMap	← impl — Map
② Map.Entry	AbstractEntry	← impl — Map.Entry

③ SortedMap	② AbstractSortedMap ← ext AbstractMap, ← impl SortedMap
	③ HashMap ← ext AbstractMap
	④ TreeMap ← ext AbstractSortedMap

In the above the classes ① & ② are the two predefined 2D collection flw abstract classes & these two classes are reserved by SUN MS developers for the future enhancements in 2D collection flw programming & these two classes are acting as a middleware layer between 2D collection flw interfaces & ~~classes~~ concrete classes.

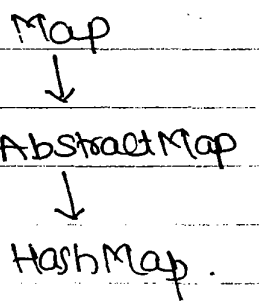
The classes ③ & ④ are called 2D collection flw concrete classes & we use them in the real world applications

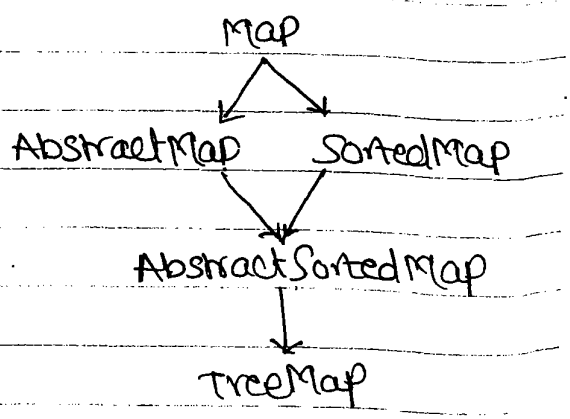
* HashMap & TreeMap :-

HashMap & TreeMap are the two bottommost concrete subclasses of 2D collection flw abstract classes.

HashMap	TreeMap
HashMap class obj never allows duplicate keys	TreeMap class obj never allows duplicate keys
HashMap class obj organizes the data in the form of (key, value) pair by following Hashing mechanism	TreeMap class obj organizes the data in the form of key, value pair by following Binary Tree's concept

We can not determine in which order HashMap class obj displays data. that is undetermined form.	TreeMap class obj displays the data in Sorted order (sorting can be done based on keys).
The operations like insertion, deletion & Modification are taking more time on HashMap	the operations like insertion, deletion & Modification takes less time on TreeMap.
Retrieval time of HashMap is more / slow.	Retrieval Time of TreeMap is less / fast.
Creating HashMap is nothing but creating an Object of HashMap class. `HashMap hm = new HashMap();`	Creating TreeMap is nothing but creating an object of TreeMap class `TreeMap tm = new TreeMap();`
Constructor: `HashMap()`	Constructor: `TreeMap()`





Q. Write a Java program which illustrate the concept of HashMap & TreeMap. (By observing the characteristics of HashMap & TreeMap, TreeMap is highly Recommended to use Compared to HashMap).

```
import java.util.*;
class hmtm
{
   psvm(-)
   {
      TreeMap hm = new TreeMap();
      sop(" Content of hm:" +hm);
      Sop ("Size of hm:" + hm.size());
      // add the data - (K,V) to 2D ref var - hm
      hm.put (100, 1.5);
      hm.put ( 1, 2.5);
      hm.put (10, 1.5);
      hm.put (90, 5.6);
      hm.put (100, 8.8); // considered as modified entry
      sop(" Content of hm:"+hm);
      Sop (" Size of hm:" + hm.size());
   // extract data from (K,V) to 2D ref var hm
   sop(" extract data from 2D ref var => entrySet()");
      Set s=hm.entrySet ();
      Iterator itr = s.iterator();
      while (itr.hasNext())
      {
         Object mobj = itr.next();
         Map.Entry me= (Map.Entry) mobj;
```

```
        Object kobj = me.getKey();
        Object vobj = me.getValue();
        Integer io = (Integer) kobj;
        Double do1 = (Double) vobj;
        int acno = io.intValue();
        double bal = do1.doubleValue();
        Sop(bal+" is bal of "+acno);
    }
Sop("extract the data from 2DC flow var keySet()")
    Set s1 = hm.keySet();
    Iterator itr1 = s1.iterator();
    while(itr1.hasNext())
    {
        Object kobj = itr1.next();
        Object vobj = hm.get(kobj);
        Iterat
        Integer io = (Integer) kobj;
        Double do1 = (Double) vobj;
        int acno = io.intValue();
        double bal = do1.doubleValue();
        Sop(bal+" → "+acno);
    }
}
}
```

8/10/13

② Legacy collection framework :-

We know that renamed form of existing data structures is known as legacy collection flw.

Like new collection flw legacy collection flw also contains two types of collection flw's They are,

① 1D Legacy collection flw.
② 2D legacy collection flw.

The Difference between legacy collection flw & new collection flw is that legacy collection flw classes & interfaces are by default belongs to synchronized & they always provided thread safety (which provides consistent results).

Where as new collection flw classes & interfaces are belongs to non synchronized. and not providing Thread safety.

SUN MS developers always recommended to use legacy collection flw classes & interfaces In JSE applications (standalone, Desktop, Two tier applications). If multiple threads are accessing the data of legacy collection flw variable then legacy collection flw by default provides thread safety & gives Consistent results.

If we use new collection flw in JSE appl^s & if multiple threads are accessing new collectn flw variables data then by default they generate in-consistant results. because new collection flw classes & interfaces are not syncronized. In this circumstance to provide the thread safety as a java programmer we must write either syncronized methods or syncronized blocks programmatically which may be complex for the programmer. Hence industry is not recommended to use new collectn flw as a part of JSE applications. and always recommended to use legacy collectn flw.

In JEE applications (web appl^s, distributed appl^s, enterprise appl^s) it is always recommended to use new collectn flw even though they are not providing thread safety bco2 every JEE appl^n runs in the context of server. and server slw will provide inbuilt thread safety. It is not recommended to use legacy collectn flw in JEE applications bco2 syncronization mechanism is implementing in two levels. They are @ at server slw level (b) at legacy collection flw level & they makes multiple times locking & multiple times unlocking which is one of the time consuming process & not a recommended process.

Hence every legacy collection flw classes & interfaces always recommended to use in non server side applications & new collection flw classes & interfaces Are recommended to use in serverside applications.

To deal with legacy collection flw programming we need to learn interfaces & classes of 1D & 2D of legacy collection flw which are present in java.util.* package.

As a part of legacy collection flw we have the following interfaces & classes

① java.util.Enumeration → (Interface)
② java.util.Vector } 1D legacy coll^n flw classes
③ java.util.Stack }
④ java.util.Dictionary } 2D legacy coll^n flw classes
⑤ java.util.Hashtable }
⑥ java.util.properties }

① java.util.Enumeration :-
Enumeration is one of the predefined legacy collection flw Interface.

The purpose of Enumeration (I) object is that to extract or retrieve the data from any legacy collection flw variables in forward direction.

Like Iterator (I) object Enumeration (I) object

is also pointing just before the first element of forward directional elements of legacy collection flw variable.

The functionality of Enumeration (I) is more or less similar to Iterator (I) but Enumeration (I) object belongs to synchronized & Iterator (I) object belongs to non synchronized.

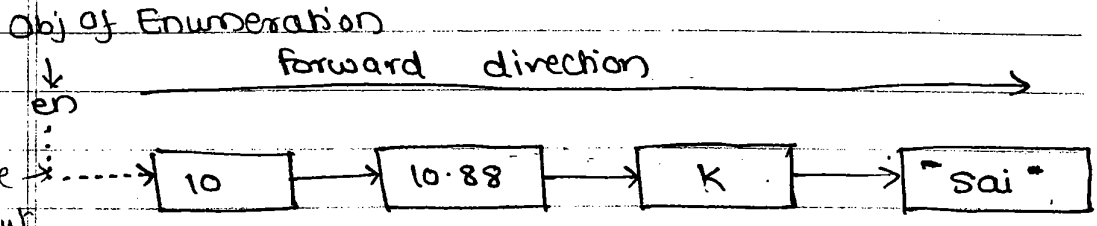
Methods in Enumeration (I) :—

① public boolean hasMoreElements ()

② public Object nextElement ().

Method ① returns true provided Enumeration(I object is having next element in legacy collection flw variable. Otherwise it returns false.

Method ② is used for obtaining next element of any legacy collection flw variable wrt the object of Enumeration (I) provided method ① must returns true.




```
while (en.hasMoreElements())
{
    Object obj = en.nextElement();
```

```
class c = obj.getClass();
                → java.lang.Integer class
String cname= c.getName();
                → java.lang.Integer
if (cname.equals("java.lang.Integer"))
{
   Integer io= (Integer)obj; // Dynamic obj
   int x = io.intValue();          type casting
}

if (cname.equals ("java.lang.Double"))
{
   Double do1 = (Double)obj; // Dynamic obj
   double d = do1.doubleValue(); type casting
}
   .
   .
   .
}
```

The process of converting reference of object of java.lang.Object into required derived class object reference dynamically is known as dynamic object type casting.

For further Notes Refer Notebook ③ from pg number 91 onwards-

9/10/13

② java.util.Vector :-

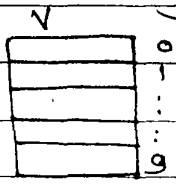
Vector is one of the concrete LD legacy collection fw class present in java.util.* pkg

Vector class object organizing the data in the form of cells.

Cell values are storing in Heap memory & cell addresses are storing in associative memory.

Creating a vector is nothing but creating an object of vector class.

e.g: vector v = new vector();



The default capacity of the vector is 10.
Default size of the vector is 0.
capacity represents no. of cells initially created & size represents number of values occupied in cells.

Profile of java.util.Vector :-

Constructor

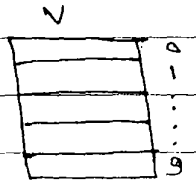
① vector()
② vector(int)

Instance method

③ public int capacity()
④ public int size()
⑤ public void addElement(Object).
⑥ public void addElement(int, objea).
⑦ public Object getElement(int)
⑧ public void removeElementAt(int).
⑨ Public void removeElement(object)
⑩ public Enumeration elements()

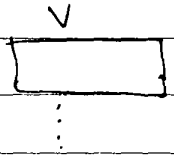
constructor ① is used for creating an object of vector without specifying number of cells to be created initially

e.g: vector v = new vector();



Constructor ② is used for creating an object of vector class by specifying number of cells to be created initially

e.g: vector v = new vector(1);



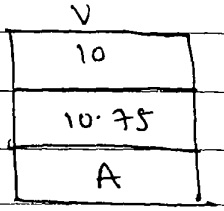
Here 1 represents one cell is created in the beginning & its expanded depends on the incoming flow of the data i.e. expandable

Method ③ & ④ are used for obtaining Capacity & Size of the vector class object.

```
e.g: vector v = new vector();
     SOP ("Capacity of v =" + v.capacity()); // 10
     SOP(" Size of v = " + v.size()); //0.
```

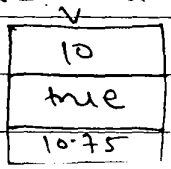
Method ⑤ is used for adding an element to vector class object one by one.

```
e.g: v.addElement (10);
     v.addElement (10.75);
     v.addElement('A');
     SOP (v); // 10, 10.75, A
```



Method ⑥ is used for adding an element to the vector at specific position.

```
e.g: v.addElement (1, true);
     SOP (v); // 10, true, 10.75, A
```



Method ⑦ is used for obtaining the value of vector class object by passing valid existing position. if we pass invalid position then we get java.lang.IndexOutOfBounds exception

e.g: Object obj1 = v.getElement(1); // true

Object obj2 = v.getElement(8); // IndexOutOfBound exception

Method ⑧ & ⑨ are used for removing the elements of vector either on the basis of position or on the basis of content.

e.g: v.removeElement(1); // Based on index

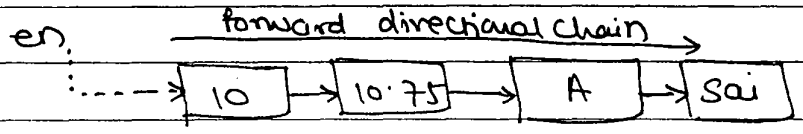
:OR:

v.removeElement(10.75); // Based on content

Method ⑩ is used for extracting the data from vector class object & forms forward directional chain. this method returns an object of Enumeration interface & it is bydefault pointing just before the first element of the chain.

e.g: sop(v);

Enumeration en = v.elements();



v
10
10.75
A
sai

while (en.hasMoreElements())

{

Object obj = en.nextElement();

sop(obj); // 10, 10.75, A, sai

}

Q. Write a java program which illustrate the concept of vector?

```
// vector.java
import java.util.*;
Class vector
{
    psvm(-)
    {
        vector v = new vector();
        sop("content of v =" +v);
        sop(" capacity of v =" + v.capacity());
        sop(" size of v =" + v.size());
        // add the data to v
        v.addElement (10);
        v.addElement (10.75);
        v.addElement ('A');
        v.addElement(" swap");
        v.addElement( 100);
        v.addElement( 100.85);
        sop(" content of v =" +v); // [ ]
        sop(" capacity of v =" +v.capacity()); // 10
        sop(" size of v =" +v.size()); // 6
        // Extract the data from v
        int is = 0;
        double ds = 0.0;
        Enumeration en = v.elements();
        while (en.hasMoreElement())
        {
            Object obj = en.nextElement();
            sop (obj);
            Class C = obj.getClass();
            String cname = C.getName();
```

```
if(cname.equals("java.lang.Integer"))
{
  Integer io = (Integer) obj;
   int x = io.intValue();
    is = is+x;
    if(name.equals("java.lang.Double"))
    { Double do1 = (Double) obj;
      double x= do1.doubleValue();
        ds = ds+x;
    }
}
sop(" int Sum =" + is);
sop(" double sum =" + ds);
}
}
```

③ "java.util.Stack :-

- Stacks is one of the concept of data structure implemented in java programming & suppüed in the form of predefined class called java.util.Stack.

Creating a stack is nothing but creating an Object of stacks class.

e.g: Stack s = new Stack ();

Stack class is one of the sub class of vector class so that all methods of vector are inherited into Stack Class.

Stack class object organizes the data in the form of cells like vector

The basic working principle of stack is LIFO. LIFO principle says whatever the element which is inserted last is that element will be taken out first.

10/10/13

The following diagram gives positional view about Organization of elements in the Stack.

Relative pos. of stack	absolute pos of stack	S	Absolute pos. Array	Relative position of Array
4	3	10	0	1
3	2	20	1	2
2	1	30	2	3
1	0	40	3	4

Sop(S); 11 10 20 30 40 ← Right to left associativity

Profile of java.util.Stack :-

Constructor

① Stack()

Instance methods

② public boolean empty()

③ public void push(Object)

④ public Object pop()

⑤ Public Objects peak()

⑥ public int search(object)

Constructor ① is used for creating an obj of Stack class. Method ② returns true provided stack is empty (size=0) it returns false in the case of non empty (size>0)

Method ③ is used for inserting an element into the stack. Method ④ is used for removing Topmost element from the stack permanently. Method ⑤ is used for retrieving topmost element of the stack. method ⑥ is used for searching an element in the stack. if the element is found in the stack then it returns a positive value (stack relative position) if element is not found then it returns -1.

Q. Write a java program which illustrate the concept of stack operations.

```
// StackDemo.java
import java.util.*;
class StackDemo
{
    psvm(String args[])
    {
        Stack s = new Stack();
        sop("Content of s = " + s); // []
        sop(" Size of s = " + s.size()); // 0
        sop(" is s empty ? = " + s.empty()); // True
        // add data to s
        s.push(10);
        s.push(20);
        s.push(30);
        s.push(40);
        sop("Content of s = " + s); // [10,20,30,40]
        sop(" Size of s = " + s.size()); // 4
        sop(" is s empty ? = " + s.empty()); // false
        //remove topmost element.
        sop(" deleted element = " s.pop()); // 40
        sop(" content of s = " + s); // [10,20,30]
        // extract topmost element
        sop(" topmost element = " s.peak()); // 30
        sop(" Content of s = " + s); // [10,20,30]
        // search the elements 10 & 100
        int srp = s.search(10);
        sop("Stack relative position of 10 = " + srp); // 3
        int srp1 = s.search(100);
        sop(" Stack relative position of 100 = " + srp1); // -1
    }
}
```

java.util.Dictionary :-

Dictionary is one of the 20 legacy collection flw abstract class.

Dictionary class object organizes the data in the form of (key, value) pair.

Since Dictionary is an abstract class whose object can not be created directly but its object can be created indirectly. wrt its subclasses.

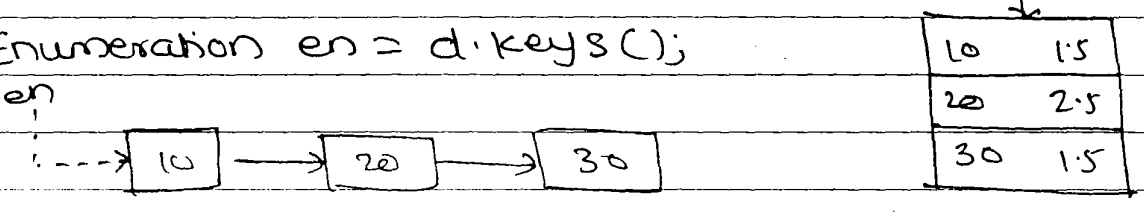
Profile of Dictionary class :-

Instance method

① public abstract int size()
② public abstract boolean isEmpty()
③ public abstract void put (Object, Object)
④ public abstract Object get (object)
⑤ public abstract Enumeration keys()

for methods ①,②,③,④ explanation see the methods of Map interface.
Method ⑤ is used for obtaining the set of keys. Pass these keys to the get(-) method and obtain values of value

```
Sop(d); // {(10=1.5), (20=2.5), (30=1.5)}
Enumeration en = d.keys();
```



```
while (en.hasMoreElements())
{
  Object kobj = en.nextElement();
  Object vobj = d.get(kobj);
  sop( vobj + " is bal of " +kobj);
}
```

java.util.Hashtable :-

It is one of the sub class of Dictionary sothat all methods of Dictionary are defined in Hashtable.

Hashtable class object organizes the data in the form of (key, value) pair by following hashing mechanism (which is not disclosed by SUN developers).

Since hashtable is following hashing mechanism for organizing the data in the form of key, value pair, we cannot determine in which order Hashtable class object displays. that is undetermined form.

The functionality of Hashtable is more or less similiar to HashMap but Hashtable Class object belongs to syncronized & never allows null values for (key, value) pair. whereas HashMap class object belongs to non syncronized and allows null values for (key, value) pair.

Profile of java.util.Hashtable :-

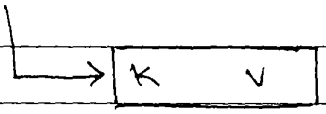
Constructor

① Hashtable ()

② Hashtable (int)

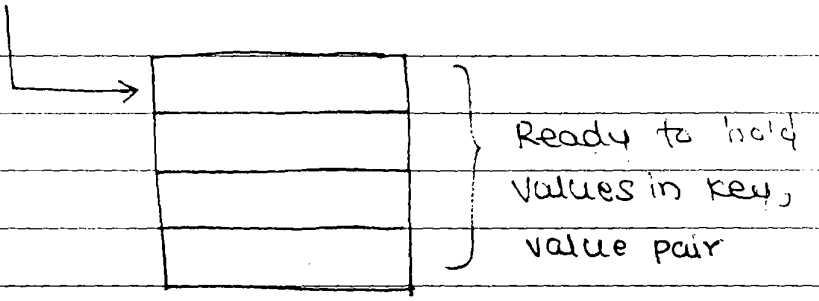
Constructor ① is used for creating an object of Hashtable without specifying number of initial entries to be created.

eg: Hashtable ht = new Hashtable ();



Constructor ② is used for creating an object of Hashtable by specifying number of entries to be created initially.

eg: Hashtable hb = new Hashtable (4);



Q. Write a java prg which will implement the following problem.

States	Capitals
MH	Mumbai
AP	Hydrabad
Goa	Panaji
KAR	Banglore

Validations:-

① Enter the state & find its capital.
Ex. Mumbai is capital of MH.

② If state is not found then display "NO Idea"

③ If state is not entered then disply "plz Enter State".

Ensure thread safety.

```
// Htable.java
import java.util.*;
Class Htable
{
    psvm(-)
    {
        Hashtable ht = new Hashtable();
        ht.put("MH", "Mumbai");
        ht.put("AP", "Hyderabad");
        ht.put("GOA", "panaji");
        ht.put("KAR", "Banglore");
        sop("----------");
        Enumeration en = ht.keys();
        while(en.hasMoreElements())
        {
            Object state = en.nextElement();
            Object cap = ht.get(state);
            sop(state + "  " + cap);
        }
    sop("----------");
    if (args.length==0)
    {
        sop("plz enter one state value:");
    }
    }
```

```
        else
        {
            String state = args[0].toUpperCase();
            Object cap = ht.get(state);
            if( cap != null) // capital found
            {
                sop(cap+" is capital of "+state);
            }
            else // capital not found
            {
                sop("No Idea");
            }
        }
    }
}
```

11/10/13

Limitations Of Hashtable :-

~~① Properties class Obj~~

① Hashtable class object is unable to read the data from properties file/ resource bundle file

② Hashtable class object is unable to develop flexible applications

③ Hashtable class obj is unable to read the data of environmental variable names & env. variable values.

Conclusion: To eliminate the above problems of Hashtable we use an another predefined class called Properties class.

Properties Class :-

It is one of the predefined subclass of Hashtable class & all the methods of Hashtable are inherited into properties class.

Advantages of Properties class over Hashtable:-

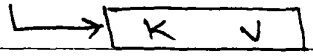
① properties class obj is able to read the data from properties file.

② properties class object allows us to develop flexible applications.

③ Properties class object allows us to read the data of env. variable names and env. variable values.

* Creating properties is nothing but creating an object of properties class.

eg: Properties p = new Properties();

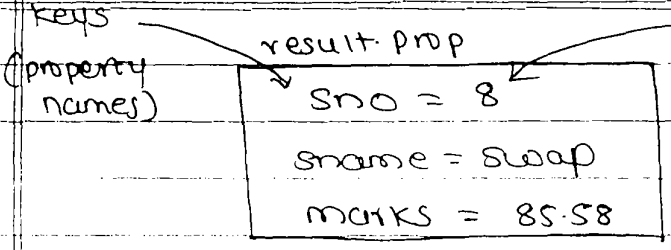


Here p object resides in main memory (heap memory).

Properties file:-

Defn:- A property file is one of the external text file which is organizing the data in the form of (key, value) pair

property file to be created in any of the text editors (Notepad, Editplus) and it should be saved on some filename with an extension either .prop



Here property file result.prop resides in secondary memory (Hard disk).

Profile of java.util.Properties :-

Constructor

① Properties()

Instance Method

② public void load(FileInputStream)

③ public String getProperty(String).

Constructor ① is used for creating an object of properties class.

e.g: Properties p = new Properties();

Method ② is used for loading or transfering the content of properties file into the properties class object by opening the property file in read mode with the help of FileInputStream class.

```
e.g: FileInputStream fis = new FileInputStream(
                                "result.prop");
     p.load(fis);
```

Method ③ is used for obtaining property value by passing property name/ key

e.g: String sno = p.getProperty("sno");

* **Steps | Guidelines for reading the data from properties file :-**

① Create the property file on any text editor and save it as some file name with an extension ".prop" or ".rbf".

② Open the property file in read mode with the help of FileInputStream class.

③ Create an object of properties class.

④ Load | Transfer the content of property file (from secondary memory into properties class object in main memory)

⑤ Obtain the property values (value of value) by passing property names (value of key)

⑥ Close the property file which was opened in read mode in Step ②

Q. Write a java program which will read the data from property file by making use of properties class.

```
// prop1.java
// Step1: we must ensure to Create property file as
        result.prop
        sno = 8
        sname = suraj
        marks = 85.58
import java.util.*;
import java.io.*;
```

```
class Prop1
{
    psvm (-)
    {
        try
        { // Step②
          Scanner sc= new Scanner(System.in);
          Sop(" Enter properties filename: ");
          String fname = sc.nextLine();
          FileInputStream fis = new FileInputStream
                                    (fname);
          // Step③
          Properties p = new Properties();
          // Step④
            p.load (fis);
          // Step⑤
          Sop(" Student no: "+ p.getProperty ("sno"));
          sop (" Student name: "+ p.getProperty ("sname"));
          sop (" stud marks: "+p.getProperty ("marks"));
          // Step⑥
            fis.close();
        }
        Catch ( FileNotFoundException fe)
            sep(" Prop. file doesn't exist");
        catch ( Exception e)
            sep (e);
    }
}
```

Q. Write a java program for obtaining env. variable names & their values

```
// prop2.java
import java.util.*;
class prop2
{
    psvm(-)
    {
        Properties p = System.getProperties();
        Enumeration en = p.keys();
        while(en.hasMoreElements())
        {
            Object evn = en.nextElement();
            Object envv = p.get(evn);
            sop(evn + "------>" + envv);
        }
    }
}
```

In order to obtain environmental variable names and their values we use the following method.

java.lang.System

↓

public static Properties getProperties()

16/10/13

* Files / String Handling (java.io.*) :-

In the context of file programming one can write 2 types of applications they are,

① Non-persistent applications / volatile applications.

② Persistent applications / Non-volatile applications.

A non persistent appln is one whose result stored in main memory temporary as long as power is on the data will be available & once the power is off the main memory data will be lost.

eg: All our previous examples comes under non persistent applications.

The persistent appln is one in which the result of the appln stored permanently

The process of storing the data permanently is known as data persistency.

Data persistency can be archieved in 2 ways they are,

① By using files concept.

② By using Database slw.

If we store the data permanently in the form of files then the data of the files can be manipulated by any unauthorised user. Hence industry is not recommended to store the data permanently in the form of files because files of any programming language are not providing security in the form of [illegible]

If we store the data permanently in the form of database slws then the data cannot be manipulated by any unauthorized users because most of the popular database products provides effective security in the form of usernames & passwords. Hence industry is highly recommended to store the data permanently in the form of database slws only.

In order to deal with file programming of java we must import a predefined package called java.io.* whereas to deal with database programming of java (JDBC) we must import
① java.sql.*
② javax.sql.*.

File :-

Defn :- (flat file).

"A file is a collection of records If we want to store the data permanently then we must choose one file we know that files resides in secondary memory (Hard Disk)"

e.g: Student.data

10	sai	99.9	← Record 1
18	Suraj	80.8	← Record 2
⋮	⋮	⋮	⋮
⋮	⋮	⋮	← Record n.

Here Student.data is the name of the file resides in secondary memory

Every record of a file is one of the java object programmatically an object data becomes record in the file & a record of a file becomes object in java.

Defination Of Record :-

" Collection of fields values (datamembers) is known as record "

e.g: (10, Sai, 99.9) is called Record.

Defination of Stream :-

" Flow of data/bytes/bits between main memory & secondary memory is known as Stream "

* Operations on files :-

On files we can perform two types of operations they are,

1. Write / output Stream operation.
2. Read / Input Stream operation.

1. Write operation :-

The purpose of this operation is transfering the data from main memory into secondary memory

Steps for Write operation :-

1. Choose the file name
2. open the file in write mode
3. perform cycles of write operations.

While we are performing write operation we get the following consequences.

① Trying to write the data into the file where ~~we~~ get the secondary memory (Hard disk) is full.

② Trying to write the data into read only file because of these two consequences we get predefined exception called java.io.IOException

② Read operation :-

This operation is used for reading the data from the file or transfering the data from the file of secondary memory into main memory

Steps for read operation :-

① Choose the filename

② open the file in read mode.

③ perform the cycle of read operations.

While we are performing read operation we get the following consequences.

① trying to read the data from the file where the specified file does not exists in secondary memory.

with this consequence we get a predefined exception called java.io.FileNotFoundException

② Trying to read the data from corrupted file with this consequence we get a predefined exception called java.io.IOException

Conclusion :-

Hence in file programming of java we get 2 types of exception.

① java.io.FileNotfoundException.

② java.io.IOException.

Both the above exceptions are belongs to checked exceptions & FileNotfoundException is one of the subclass of IOException.

17/10/13

* Types of Streams :-

Based on transfering the data in the form of bytes between main memory & secondary memory, streams are classified into two types they are,

① Byte streams.

② Character streams.

① Byte streams :-

Def^{n} :- In Byte streams the data will be transfered in the form of byte by byte in other words byte by byte transfering the data between main memory & secondary memory is known as byte streams

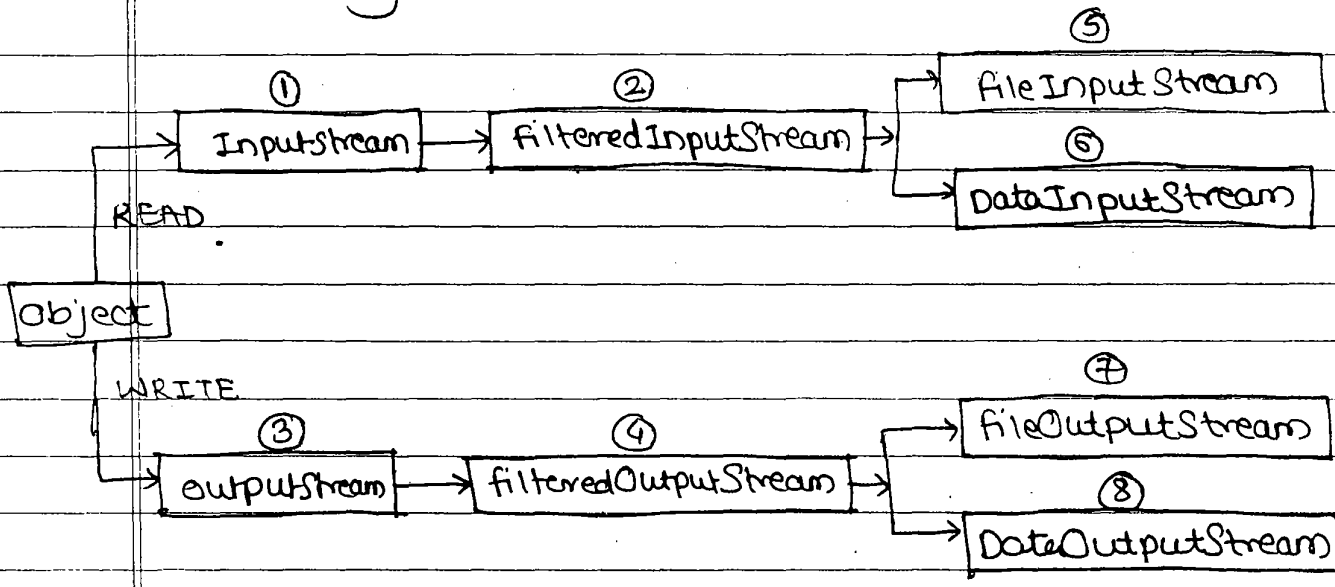
byte streams contains two categories of classes they are,

@ some category of classes participates in write operation which will transfer one byte of data from main memory to secondary memory.

⑥ Some other category of classes participates in read operation which will transfer one byte of data from secondary memory to main memory.

Both write operation & read operation related classes of byte streams are present in the package called java.io.*.

Hierarchy chart of byteStreams :-



In the above hierarchy chart the classes ① ② ③ ④ are predefined abstract classes & they are providing background role for the concrete Subclasses & hence we won't use directly as a part of file programming

The classes ⑤ ⑥ ⑦ ⑧ are bottommost concrete Subclasses of abstract classes of byte stream & hence we use these classes as a part of file programming

FileOutputStream :-

The purpose of FileOutputStream class is to open the file in write mode in other words opening the file in writemode is nothing but creating an object of FileOutputStream class.

Profile of FileOutputStream class :-

Constructors

① FileOutputStream (String) } Throws
② FileOutputStream (String, boolean) } IOException

Instance methods.

③ public void writeXxx (xxx value)
④ public void writeLine (String)
⑤ public void write (xxx)
⑥ public Void close ()

constructors ① & ② are used for opening the file either newly or existing file by specifying either true or false for appending or overlapping respectively

Case1 :- If file is not existing newly selected file & boolean = True or false then The new file is opened in write mode & an object of fileOutputStream is pointing to starting entry of file.

ex:- FileOutputStream fos = new FileOutputStream ("student.data");

:OR:

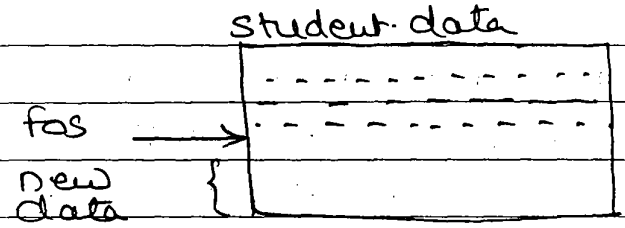
fileOutputStream fos = new fileOutputStream("student.data", false);

:OR:

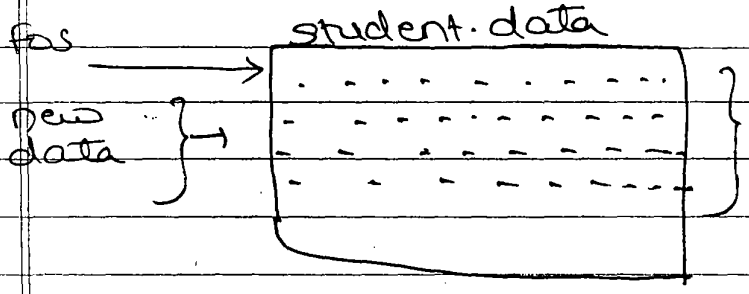
fileOutputStream fos = new fileOutputStream ("student.data",

fos

Case2:- If filename already exists & boolean is equals to true then
Existing filename is opened in write mode and an object of fileOutputStream is pointing at the end of the existing data & new data is appending
ex:- FileOutputStream fos = new fileOutputStream ("student.data", true);



case 3:- If filename already exists & boolean = false then
Existing filename is opened in write mode & an object of fileOutputStream is pointing to existing data & it overlaps with new data.
ex:- FileOutputStream fos = new fileOutputStream ("student.data", false);
: OR:
FileOutputStream fos = new fileOutputStream ("student.data");



In method ③ Xxx represents any fundamental datatype. Method ③ is used for writing the fundamental data to the file which is opened in write mode.

Method ④ is used for writing any type of data in the form of String type. to the file which is opened in write mode

Method ⑤ xxx represents either fundamental datatype or string type data. Method ⑤ is one of the overloaded method meant for writing both fundamental & string type data to the file which is opened in write mode

Method ⑥ is used for Closing the file which is opened in write mode

ex:-

```
finally
{
    try
    {
        if( fos != null)
        {
            fos.close();
        }
    }
    catch (IOException ioe)
    {
        Sep("unable to close file in
            write mode");
    }
}
```

Resource Relinquishing logic

18/10/13

FileInputStream :-

The purpose of FileInputStream class is to open the file in read mode programmatically opening the file in read mode is nothing but creating an object of FileInputStream class.

Profile of FileInputStream Class :-

Constructor

① FileInputstream (String) throws FileNotfoundException, IOException.

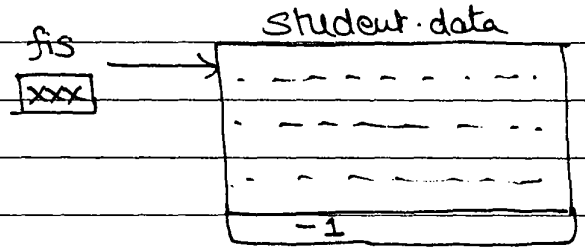
Instance Methods

② public xxx readXxx ( )
③ public String readLine ( )
④ public int read ( )
⑤ public void close ( ).

Constructor ① is used for creating an object of FileInputStream class by opening the file in read mode

Case 1:- if filename exists then
File is opened successfully in read mode and an object of FileInputStream is pointing to starting enmy of file.

Ex: FileInputStream fis = new FileInputStream ("student.data");



If file is opened successfully then fis contains not null & it can be used for reading the data from the file until end of file is taking place In java programming end of file is indicated by -1

As and when reading is in progress, automatically fis object is advancing to next entries in the file until end of file is taking place.

Case 2 :- If filename does not exists then file is unable to open and an object of fileInputStream contains null.

Ex: fileInputStream fis = new fileInputStream(
null "Student.data");
↳ Throws fileNotfoundException

Q. Write a Java program for implementing following requirements.
(a) Write 1 to 10 numbers from file
(b) Read 1 to 10 numbers from file.

(a)
```
// FileWrite.java
import java.io.*;
import java.util.Scanner;

class FileWrite
{
    psvm (-)
    {
        FileOutputStream fos=null;
```

```
try
{  // Step 1
   Scanner sc = new Scanner(System.in);
   sop(" Enter filename to write 1-10 nos:");
   string fname = sc.nextLine();
   // Step 2
   fos = new fileOutputStream(fname, true);
                                  // appending
   // Step 3
   for (int i=1; i<=10; i++)
   {
      fos.write(i);
   }
   sop(" Data written to file");
} catch
catch(IOException e)
{ sop(" unable to open file in write mode");

catch(Exception e)
{ sop(e); }

finally
{  try
   {  if(fos1 = null)
      {  fos.close();
         sop("file closed");
      }
   }
   catch(IOException e)
   {  sop(e); }
}
}
}
```

FileRead.java.

```
import java.io.*;
import java.util.Scanner;

public class FileRead
{
    psvm(-)
    {
        FileInputStream fis = null;
        try
        { // Step1
          Scanner sc = new Scanner(System.in);
          sop("Enter filename to read data");
          String fname = sc.nextLine();
          // Step2
          fis = new FileInputStream(fname);
          // Step3
          int x;
          while ((x=fis.read()) != -1)
          {
              sop(x);
          }
        }
        catch(FileNotFoundException fe)
        { sop("file does not exist"); }

        Catch (IoException ioe)
        { sop("unable to read from cumupted file");}

        Catch (Exception e)
        { sop(e); }
```

```
finally
{ sop(" I am from finally block");
  try
  {   if (fis!=null)
      {
          fis.close();
          sop("file closed");
      }
  }
  catch(IOException e)
  {  sop(e); }
}
}
```

19/10/13

Q Write a java prg which will copy content of one file into another file or implement copy command of DOS

```
import java.io.*;
class FileCopy
{
   psvm(-)
   {
      if (args.length!=2)
      {
         sop(" plz enter two file names");
      }
      else
      {
         FileInputStream fis=null;
         FileOutputStream fos=null;
```

```
try
{
    String sfile= args[0];
    String dfile= args[1];
    fis = new FileInputStream(sfile);
    fos = new FileOutputStream(dfile, true);
    // copy process code
    int K;
    do
    {
        K= fis.read();
        char ch= (char) K;
        fos.write(ch);
    }
    while(K!=-1);
    sop(sfile+" content is copied to"+dfile);
}
catch (FileNotFoundException fe)
{ sop(" source file doesn't exist"); }

catch ( IOException ioe)
{ sop(" unable to open file in write mode");}

catch( Exception e)
{ sop(e); }

finally
{
  try
  { if (fos != null)
      { fis.close();
        sop(" Source file closed");
```

cmd> type <filename>


```
            if (fos != null)
                fos.close(); sop("Dest file closed");
        }
        catch(Exception e)
        { sop(e); }
    } // finally
} // else
} // main
} // class
```

cmd> javac fileCopy.java
cmd> java fileCopy x.txt y.txt

must be present in HDD.

Q. Write a java program which will display the content of the file —:OR:— Implement "type" command of dos

```
import java.io.*;
class Displayfile
{
    psvm(-)
    {
        if (args.length != 1)
        {
            sop(" Enter one file name");
        }
        else
        {
            FileInputStream fis = null;
```

```
try
{
    String fname = args[0];
    fis = new FileInputStream(fname);
   // type process code
    int k;
   while ( (k = fis.read()) != -1)
   {
       char ch = (char) k;
        S.o.print(ch);
   }
}
catch (FileNotFoundException fe)
{ S.o.p(" file does not exist"); }

catch ( IOException ie)
{ S.o.p(" Unable to read from currupted file"); }

finally
{
   try
    { if (fis != null)
       { fis.close();
          S.o.p("file closed");
       }
    }
    catch (Exception e)
   { S.o.p(e); }
}
}
}
```

[illegible] javac DisplayFile.java (cmd)\ java DisplayFile x.txt

DataInputStream :-

The purposes of DataInputStream class are,

① To read the data from keyboard (local reading)

② To read the data between multiple machines which are located either in same network or in different n/w (Remote Reading).

Profile Of DataInputStream class:-

Constructor

① DataInputStream (InputStream)

Instance methods

② public xxx readXxx ()

③ public String readLine().

Constructor ① is used for creating an obj of DataInputStream class by passing an obj of InputStream class for obtaining the purposes of DataInputStream class.

ex: DataInputStream dis = new DataInputStream("System.in");

In method ② xxx represents any fundamental DataType. Method ② is used for reading any type of fundamental data either from keyboard or from Remote machine.

Method ③ is used for reading any type of data in the form of string type

UTF: Universal Text Format ⇒ text + Smilyings + Hyperlinks

DataOutputStream :-

The purpose of DataOutputStream class is that to write the data between multiple machines which are located either in same n/w or in different network (Remote writing)

Profile of DataOutputStream :-

Constructor

① DataOutputStream (OutputStream).

Instance Methods.

② public void writeXxx (xxx value)

③ public void writeUTF (String)

④ public void write (xxx);

Constructor ① is used for creating an object of DataOutputStream class by passing an Object of OutputStream class. for getting the purpose of ~~do~~ DataOutputStream class.

In Method ② xxx represents any fundamental datatype method ② is used for writing fundamental data. 2011

Method ③ is used for writing universal text format data (UTF).

In Method ④ xxx represents any fundamental datatype & String Method ④ is used for writing all types of values & it is one of the overloaded method.

Q. Write a java prg which will accept two numbers from keyboard by using DataInputStream & compute their sum

DataRead.java

```
import java.io.*;
Class DataRead
{
    psum(-) throws Exception   otherwise write all stmts in try & catch block
    {
     DataInputStream dis = new DataInputStream(
                                    System.in);
     sop(" Enter first no:");
     int x = Integer.parseInt(dis.readLine());
     Sop ("Enter second no:");
      int y = Integer.parseInt(dis.readLine());
     Sop(" Sum =" +(x+y));
    }
}
```

20/10/13

* Serialization & Deserialization :-

Serialization :-

Let us assume there exist an object in the main memory with n bytes of data. If we want to transfer n bytes of data of Object from main memory to the file of secondary memory by using ordinary streams (byteStreams or characterStreams) then we have to perform either n redundent write operations or n/2 redundent write operations. Which is one of the time consuming process & appl will take more amount of time

To ~~ete~~ eliminate this type of problem & for enhancing performance of ordinary streams we have a concept called serialization.

In serialization concept by performing single write operation one can transfer/send the entire object data into the file of secondary memory.

In other words serialization advantage is to eliminate redundent write operations & it performs single write operation.

Defination :-
The process of saving/transfering the entire content of the object of main memory into the file of secondary memory by performing single write operation is known as Serialization.

Serialization concept always participates in write operation.

Deserialization :-

We know that a file is a collection of records. A Record is a collection of fields values. If we want to transfer the entire content of a record of a file, which is containing n bytes & if we use ordinary streams (either bytestreams or characterstreams) then we have to perform either n read operations or n/2

read operations. Which is one of the time consuming process & appln will take more execution time.

To eliminate the above redundent read operations of ordinary streams, we use the concept of deserialization.

In deserialization concept irrespective of number of bytes present in the record with the single read operation we transfer the record of a file of secondary memory into the main memory

In other words the advantage of deserialization concept is that it eliminates redundent read operations

Defination :-
The ability to restore the entire content of the record of a file of secondary memory into main memory by performing with single read operation is known as deserialization.

Deserialization concept always participates in read operation.

* Phases for Developing Serialization & Deserialization Application :-

① Developing serializable Subclass.
② Developing Serialization process.
③ Developing Deserialization process.

① Developing serializable subclass :-

Whichever class object want to participate in Serialization & deserialization ~~concept or~~ process, the corresponding class defination must implements a predefined interface called ~~java.io.Serializable~~ java.io.Serializable. & it is one of the Marker / tagged Interface

Marker Interface :-

A marker interface is one which is not containing any explicit abstract methods (containing internally some abstract methods (hidden)) & it provides runtime behaviour to the object of Subclass of marker interface (By defining automatically the abstract methods of marker interface in the context of derived class. during Runtime by JVM).

Steps for Developing Serializable Subclass :-

① Choose an appropriate package for placing Serializable Subclass for common access & ensure it should be first executable Stmt.

② Choose an appropriate user defined class & ensure whose modifier must be public

③ The class which is selected in step ② must implements java.io.Serializable interface Hence step ② class is known as Serializable subclass.

④ Choose appropriate datamembers in each & every Serializable subclass.

⑤ Define set of SetXxx(-) methods for each & every datamember of Serializable subclass (Used for setting values to datamembers)

⑥ Define set of getXxx() methods for each & every datamember of Serializable subclass. (which are used for obtaining the values from Serializable subclass object)

⑦ Whichever userdefined Serializable subclass we are placing in the package, that class name will be given as filename with an extension ".java".

Example :- Develop a Serializable subclass for student.

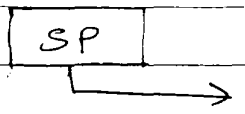
```
// Student.java  — ⑦
Package sp  — ①
public class Student implements java.io.Serializable
{                    ②                    ③
    int stno;
    String name;      } ④
    float marks;
```

```
public void setStno (int stno)
{ this.stno = stno; }

public void setName (String name)
{ this.name = name; }

public void setMarks (float marks)
{ this.marks = marks; }

public int getStno ()
{ return stno; }

public String getName()
{ return name; }

public float getMarks()
{ return marks; }

}
```

(5)

(6)

cmd> javac -d . Student.java



* Each & every object of serializable subclass is an object of java.lang.Object.
For example object of student class is an object of java.lang.Object.

21/10/13

② Development of serialization process :-

Steps:-

1) Create an object of appropriate serializable subclass.

eg: Student so = new Student();

so	
0	stno
null	name
0.0	marks

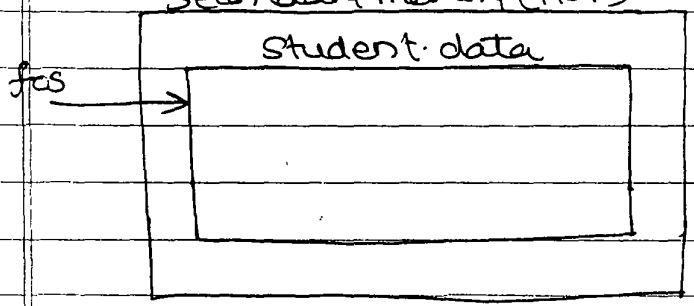
2) set the values for the datamembers of the Serializable subclass by calling setXxx(-) methods of serializable subclass.

eg: so.setStno(10);
so.setName("Sivaji");
so.setMarks("88.96f");

so	
10	stno
Sivaji	name
88.86f	marks

3) Choose appropriate file and open it into "write mode" by making use of FileOutputStream Class.

e.g: FileOutputStream fos = new FileOutputStream("student.data");



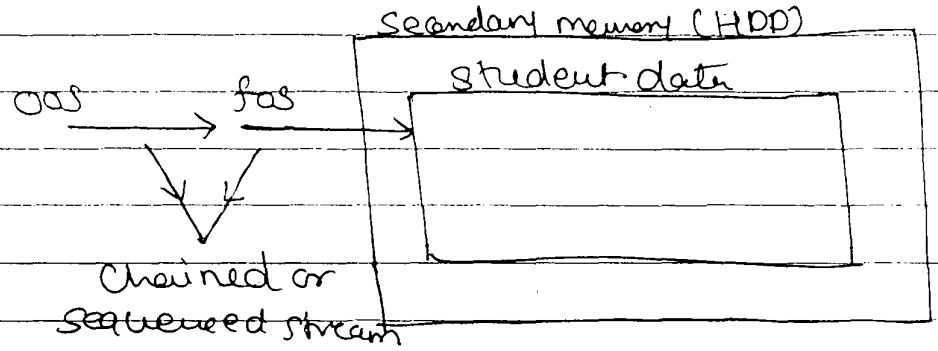
4) Create an object of ObjectOutputStream by passing an object of FileOutputStream (bcoz an object of FileOutputStream does not contain any methods for transferring ~~the~~ entire content of Serializable subclass obj into the file of secondary memory.

java.io.ObjectOutputStream →

ⓐ ObjectOutputStream (fileOutputStream).

ⓑ public void writeObject (object).

ex: ObjectOutputStream oos = new ObjectOutputStream(fos);



Because of above step an env. is provided for writing the entire content of serializable subclass obj into file of secondary memory

NOTE:- If one stream obj(oos) is pointing to another stream obj (fos) & if both of them belongs to same type then such stream objs are called chained or sequenced streams

5) Transfer or write the entire content of Serializable subclass obj into the file of secondary memory,

ex: oos.writeObject (so);

SOP(" student obj data saved successfully into the file");

6) Close/Terminate the stream objects which are pointing to the file which was opened in write mode.

e.g: oos.close(); fos.close();

Q. Write a java prg which will illustrate the concept of serialization (save student record into the file of secondary memory).

```
import sp.Student;
import java.io.*;
import java.util.Scanner;
Class Serp
{
   psvm(-)
    {
      try
       {
          // Step1
          Student so = new Student();
          // Step2
          Scanner sc = new Scanner(System.in);
          sop(" Enter Student no");
          int sno = Integer.parseInt(sc.nextLine());
          sop(" Enter Student name:");
          String name = sc.nextLine();
          sop(" Enter Student marks:");
          float marks = float.parsefloat(sc.nextLine)
          so.setSno(sno);
          so.setName(name);
          so.setMarks(marks))
          // Step3
          sop(" Enter filename to write stud data");
          String fname = sc.nextLine();
          FileOutputStream fos = new FileOut-
                putStream(fname);
```

```
// Step4
ObjectOutputStream oos = new ObjectOut
                         putStream (fos);
// Step5
oos.writeObject (so);
sop(" student datas saved to file sucessfully");
// Step 6
oos.close();
fos.close();
}
catch( IOException ioe)
{ sop( unable to open file in write mode");}

catch (Exception e)
{ sop(e);}
}
}
```

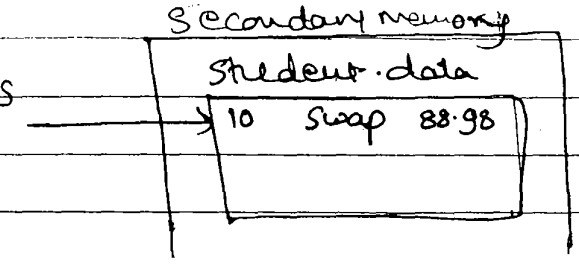
(3) Development of Deserialization process :-

Steps:-

1) Create an object of Serializable subclass for holding deserialized data (Record of a file). Eg: Student so = new Student();

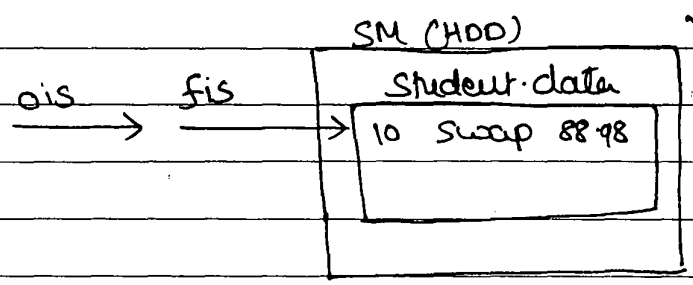
2) Choose the filename & open it into read mode by making use of FileInputStream class.

ex: FileInputStream fis = new FileInputStream("Student.data");



3) Create an object of ObjectInputStream by passing an object of fileInputStream (bcoz an object of fileInputStream class is unable to transfer the entire record of the file of secondary memory into main memory).

e.g: ObjectInputStream ois = new ObjectInputStream(fis);



4) Read or transfer or obtain entire record of a file of secondary memory into main memory

e.g: Object obj = ois.readObject();

10 swap 88.98 deserialized data

5) Typecast an object of java.lang.Object into serializable subclass object.

e.g: so = (Student) obj;

6) Obtain Deserialized data & display on the console

eg: sop("student no: " + so.getSno());

7) Close Streamobjects which are pointing to the file which was opened in read mode.

fis.close();
ois.close();

Q. Write a java program which illustrate the concept of Deserialization process (Read Student Record from file).

```
import sp.Student;
import java.util.*;
        .io.*;

Class DSerp
{
    PSVm (-)
    {
        try
        {   // step1
            Student so = new Student ();
            // step2
            Scanner sc = new Scanner(System.in);
            SOP(" Enter filename to read student
                data from file");
            String fname = sc.nextLine();

            FileInputStream fis = new FileInputSt-
                ream(fname);
            //step3
            ObjectInputStream ois = new ObjectIn-
                putStream (fis);
            // step4
            Object obj = ois.readObject();
            //step5
            So = (Student) obj;
```

```
// Step 6
sop("student no:" + so.getStno());
sop ("Stud name:" + so.getName());
sop ("Stud marks:" + so.getMarks());
//Step 7
ois.close();
fis.close();
}
catch (FileNotFoundException fe)
{ sop(" file doesn't exist"); }

catch (IOException ie)
{ sop (" unable to read from currupted
        file"); }

catch (Exception e)
{ sop(e); }

}
}
```

22/10/13

* Types of serialization :-

Serializations are classified into two types they are,

1. Complete Serialization.
2. Selective Serialization.

1. Complete serialization :-

In complete serialization all the data members of serializable subclass participates in serialization process.

e.g: SP.Student (See previous pages)- Student.java program(Pg: 135) Here Student class obj participate in complete serialization.

② Selective serialization :-
In Selective serialization some of the data members of serializable subclass participates in serialization and some of the datamembers will not participate in serialization

Whichever datamembers don't want to participate in serialization process those datamembers declaration must be preceded by transient keyword Hence transient variable never participates in serialization whereas non-transient variables in serialization process.

ex: class Bank implements java.io.Serializable
{
    transient int tokenno;
    int accno;
    String cname;
    float bal;
}

Here Bank Class obj participates in selective serialization

* Hence each & every object of subclass of Serializable subclass participates in serialization process

(2) Character Streams :-

Character Streams are those which will transfer two bytes at a time bet^n main memory & Secondary memory.

Character Streams are more effective Compare to byte Streams

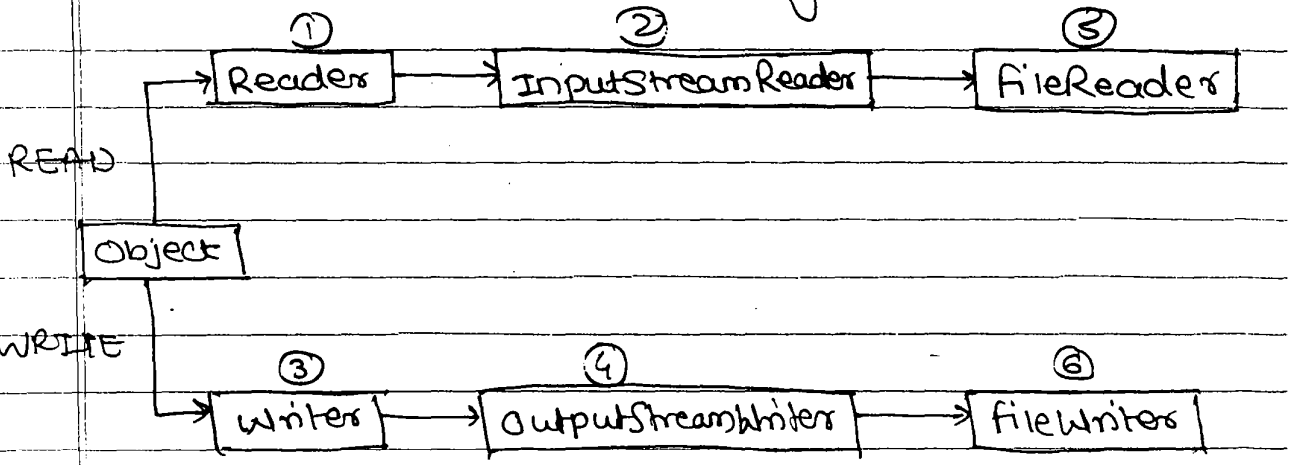
Character Streams Contains two categories of classes. They are,

a) Some of the classes participating in write operation which will transfer two bytes of data from main memory to secondary memory.

b) Some of the classes participating in read operation & they transfer two bytes of data from Secondary memory to main memory

All the above two categories of classes are present in java.io.* package.

Character Streams Hierarchy chart :-



In the above hierarchy chart the classes ① ② ③ ④ are the predefined abstract classes present in characterstreams & they are reserved by SUN ms developers for the future enhancements in characterStreams & we wont use them directly

The classes ⑤ & ⑥ are the bottom most concrete subclasses in character Streams.

fileReader class funchonality is more or less similar to fileInputStream Class. (Both of them are meant for opening the file in read mode). But fileReader Class obj reads two bytes of Data from file of Secondary memory into main memory whereas fileInputStream class obj reads one byte of data.

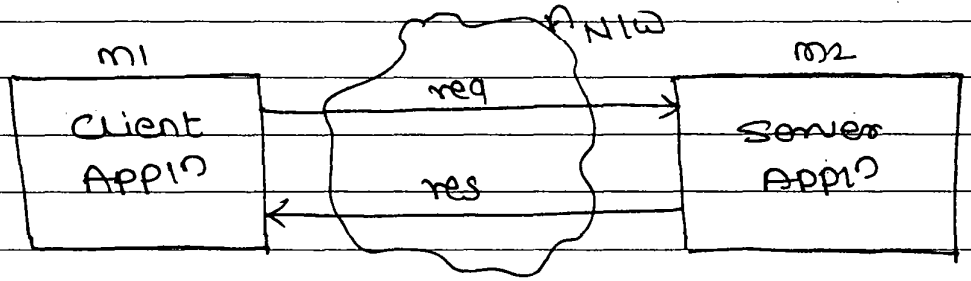
The funchonality of fileWriter class is more or less similar to fileOutputStream class. (Both of them are meant for opening file in write mode). But fileWriter class obj writes two bytes of Data from secondary memory into main memory whereas ~~file~~ fileOutputStream class obj writes one byte of data

* Network Programming :- (java.net.*).

The basic aim of networking is to share the data between multiple machines.

Programatically Network programming is always used for developing client server appln's.

In client server appln development we write two types of prgs they are,
a) Client side programs / client appln.
b) Server side programs / server appln.



Client Appln :- A client appln is a java prg which is always used for making a request to get the services from server appln.

Server appln :- A server appln is a java prg which will receive a client request, process the client request & gives response back to client appln.

To develop the client server appln client appln must be developed in one machine and server appln must be developed in another m/c. where both the m/c must

present within the network otherwise data sharability is not possible.

DNS (Domain Naming Service) :-
It is the name of the physical m/c where Server side prg resides-
e.g: The default ~~DSN~~ DNS of each & every computer is "localhost."

IP Address (Internet Protocol Address) :-
It is one of the four parts numerical address of physical m/c where Server side appln resides.

The default IP address of each & every computer is 127.0.0.1

Port number :-
It is one of the logical numerical id where Server side program is running

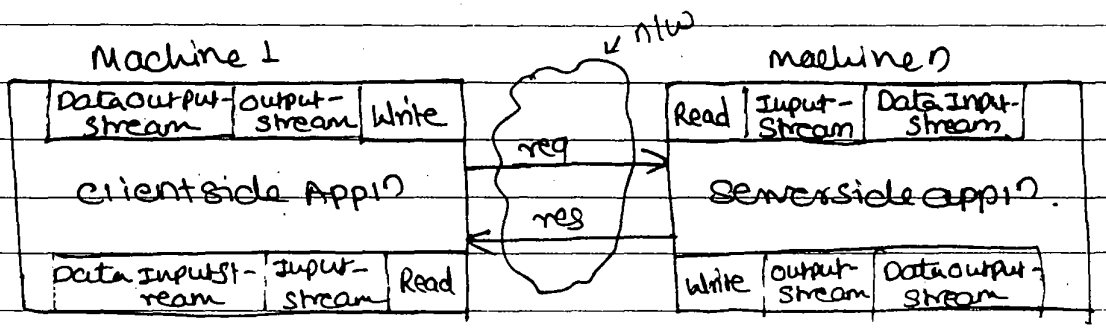
Hence in the network one m/c can contain single DNS, single IP address & multiple port numbers.

To exchange the data between Client & Server side appln's, first there must exist connection/ bridge between Client & server side appln's so that Client appln must establish connection with Serverside appln by passing (DNS name or Ipaddress, portno).

23/10/13

To exchange the data between client & server side appl^ns is nothing but performing cycle of write and read operations or OutputStream & InputStream are DataOutputStream & DataInputStreams respectively.

The following Diagram gives view about data exchanging bet^n client & serverside appl^ns.



* Steps / Guidelines for Developing server side Applications :-

① Write server side program.
② Each & every server side prg must run at certain distinct port number
③ Serverside prg accept clientSide prg request
④ Server side prg reads request parameter data by obtaining an object of InputStream & pass it to DataInputStream.
⑤ Server side prg process the client side prg request. by executing Business logic.
⑥ Serverside prg gives response to client side prg by obtaining an obj of outputStream & pass it to DataOutputStream
⑦ Server side prg closes the client request.

* Steps | Guidelines for developing Client side appl^n:

① Write a client side prg.
② Client side prg obtains a connection from server side prg by passing (DNS | IPadd, portno)
③ Client side prg makes a request to server side prg by passing an obj of OutputStream to DataOutputStream.
④ Clientside prg receives response from server side prg by obtaining an obj of InputStream & pass it to DataInputStream.
⑤ Clientside prg display the result of server side prg on client console m/c.
⑥ Client side prg closes its request by itself

* In order to develop network programming applications we must import java.net.*

API required for network programming

In java.net.* we have two predefined classes
① java.net.Socket.
② java.net.ServerSocket.

① java.net.Socket :-
Socket is one of the predefined class & whose obj is used for developing Clientside applications.

Profile of Socket class :-

Constructor

① Socket (String DNS, int portNo) throws UnknownHo-stException, IoException

Instancemethod

② public OutputStream getOutputStream ().

③ public InputStream getInputStream ().

④ public void close().

Constructor ① is used for creating an object of Socket by passing DNS / IP address & portnumber. Creating an object of soccet is nothing but establishing an connection from Serverside program i.e. from ServerSocket.

If we enter wrong DNS then we get predefined exception called java.net.UnknownHostException. If we enter wrong portno then we get java.io.IoException.

e.g. Socket S = new Socket ("localhost", 9999);
SOP (" client side prg obtains Connection from server side prg");

Methods ② & ③ are used for obtaining the objs of OutputStream & InputStream for making request, receiving request, sending response & receiving response.

Method ④ is used for closing client request by itself.

② java.net.ServerSocket :-

It is one of the predefined class used for developing serverside prgs.

Profile of ServerSocket class :-

Constructor

① ServerSocket(int portno) throws IOException

Instance Methods.

② public Socket accept();

③ public void close();

8080

Constructor ① is used for creating an object of ServerSocket by passing ~~port~~ distinct portnumber. If we choose a portnumber, & if it is always selected by some other serverside program then wrt our java prg we get a predefined Exception called IOException

Ex: ServerSocket ss = new ServerSocket(9999);

Method ② is used for accepting clientside prg request. Programmatically accepting client request is nothing but obtaining an Obj of Soacet

Ex: Socket s = ss.accept();

Method ③ is used for closing server-Socket which is nothing but stopping the execution of serverside program.

@ Write a client server appln for developing the following

@ write a serverside appln which will receive a numerical integer value, compute its square & give the result back to client side prg

ⓑ write a clientside prg which will send a numerical Integer value & obtains its square from serverside prg

Server.java ——— ①

```
import java.net.*;
import java.io.*;
class Server
{
    psvm(-)
    {
        try
        {  // step ②
            int pno = Integer.parseInt(args[0]);
            ServerSocket ss = new ServerSocket(pno);
            sop("server is ready to accept request");
            while (true)
            {  // step ③
                Socket s = ss.accept();
                // step ④
                InputStream is = s.getInputStream();
                DataInputStream dis = new DataInputStream(is);
                int cv = dis.readInt();
                sop("val of client at server:"+cv);
                // step ⑤
                int res = cv * cv;
```

```
// Step (6)
OutputStream os = s.getOutputStream();
DataOutputStream dos = new DataOutput
                       Stream (os);
dos.writeInt (res);
// Step (7)
 s.close ();
        }
    }
    catch (Exception e)
    { sop(e); }
  }
}
```

Client.java —— (1)

```
import java.net.*;
import java.io.*;
class Client
{
    psvm (-)
    {
        try
        { // Step (3)
          String dns = args [0];
          int pno = Integer.parseInt (args[1]);
          int no = Integer.parseInt (args [2]);

          Sockets = new Socket (dns, pno);
          Sop(" Client obtained connection with
                Server ");
```

```
// Step (3)
OutputStream os = s.getOutputStream();
DataOutputStream dos = new DataOutput-
                       Stream (os);
// Step (4)
InputStream is = s.getInputStream();
DataInputStream dis = new DataInput -
                      Stream (is);
// Step (5)
int res = dis.readInt();
SOP (" result from server to Client: " + res);
// Step (6)
s.close();
}
catch (Exception e)
{ sop(e); }
}
}
```

```
Cmd> javac Server.java
Cmd> java Server 8888
```

Open another dos prompt:

```
Cmd) javac Client.java
Cmd) java Client localhost 8888 5
                OR)
Cmd) java Client 127.0.0.1 8888 5
```

We will get Result from Server on Client console.

* Generics :-

Generics is one of the new facility available from the version jdk 1.5 onwards.

The concept of generics is always applicable on the concept of overloading.

We know that the overloaded concepts in java can be method overloading & constructor overloading

The concept of generics can be applied in 3 cases They are,

case1:- Let us assume there exists family of overloaded method calls. With the ordinary programming of Java for each & every overloaded method we need to define a seperate defination which is one of the time consuming process & redundancy of the code is more. To avoid these problems we apply the concept of generics on family of overloaded methods & allows us to write only one method defination and it is known as generic method.

Generic Method:-

Defination :- A generic method is one whose defination exist only once irrespective of family of overloaded method calls.

case2:- Let us assume there exists family of overloaded constructors. With the ordinary programming of java, it is mandatory to the java

programmer to write a seperate defination for each & every overloaded constructor. which is one of the time consuming process and redundancy of code is more.

To avoid these problems, we apply the concept of generics on overloaded constructors def° and permits us to define only one defination irrespective of family of overloaded constructors.

Generic constructor :—

Defination :— A generic constructor is one whose defination exists only once irrespective of family of overloaded constructor calls

Case3 :— Let us assume that there exist a class defination which is containing two integer data members & if we create an object of this class then we can enter integer values only but not other type of values. In order to enter other type of values it is mandatory to the java programmer to write seperate class definations which is one of the time consuming process & redundancy of the code is more. To avoid this problem we apply the concept of generics on redundant class definaion and permits us to define only one class def° known as generic class.

Generic class :-

Defination :- A generic class is one whose defination exists only once and whose objects allows us to place multiple values of different type.

* Advantages of Generics :-

If we apply the concept of generics on overloaded based applications then we get the following advantages,

1. Application Development time is less.
2. Redundancy of code is Reduced.
3. Application performance is enhanced.

Internally the concept of generics always operates on object type data but not in the form of fundamental type data. i.e. programmer is permitted to represent fundamental values but internally fundamental values are implicitly converted into equivalent wrapper class objects (known as autoboxing)

* Syntax for generic class :-

```
class <clsname> (<Gen var1, Gen var2, ..... Gen varn>)
{
   variable declaration belongs to Gen var1, Gen var2,
   ..... Gen varn;

   <clsname> (list of formal params belongs to
                        Gen var1, Gen var2, .... Gen varn)
   {
```

```
    Block of stmts;   // Initialization.
}

Return type  methodname ( list of formal params
    belongs to Gen var1, Gen var2, ..... Gen varn)
{
    Block of stmts;   // operation = Business logic.
}
```

If a method is not returning any value then the return type of method is void. If method is returning any value whose return type must be generic variable type.

Syntax for Generic object creation:-

```
<clsname> <clsname1, clsname2, ..... clsnamen>
    objname = new <clsname> < clsname1, clsname2,
    ..... clsnamen> (list of values if required);
```

In the above syntax <clsname> represents name of the generic class.
<clsname1, clsname2, ... clsnamen> represents list of wrapper class names & String class.

Q. Write a Java prg which illustrate the concept of generics regarding generic class, generic constructor & generic methods.

```
// GenDemo.java

Class Gen <T,S>
{
   T a;
   S b;
    Gen()
    {
    }
    Gen(T a, S b)
    {
       this.a = a;
       this.b = b;
    }

   Void setA(T a, T b)
    {
       this.a = a;
       this.b = b;
    }

    T getA()
    {
      return a;
    }

    S getB()
    {
      return b;
    }
}
```

```
class GenDemo
{
    psvm (-)
    {
        Gen <Integer, Integer> o1 = new Gen <
            Integer, Integer> ();
        o1.set (10, 20);

        Gen <Float, Float> o2 = new Gen < Float,
            Float> (10.5f, 20.5f);

        Gen <Character, Character> o3 = new Gen <
        Character, Character) ();
            o3.set ('A', 'B');

        Gen <String, String> o4 = new Gen <String,
            String> ("sai", "swapnil");

        int x1 = o1.getA();
        int x2 = o1.getB();

    sop ("val of x1: " + x1);
    sop ("  val of x2: " + x2);

    s float y1 = o2.getA();
      float y2 = o2.getB();

    sop ("val of y1: " + y1);
    sop ("val of y2: " + y2);
```

```
        char c1 = o3.getA();
        Char c2 = o3.getB();

        Sop (" val of c1:" + c1);
        sop (" val of c2:" + c2);

        String k1 = o4.getA();
        String k2 = o4.getB();

        sop (" val of k1:" + k1);
        Sop (" val of k2:" + k2);
    }
}
```

Generics must be always suppied individually by considering same number of parameters by considering diff type & order.

If we have different number of parameters then we write seperate generic class.