****

Practice Sheet #1

****

Practice 1.1

```
//      Hello World Application      //


      class HelloWorldApp {
         public static void main ( String args [ ] ) {
           System.out. println (" Hello  Java !" ) ;
         }
      }
```

Practice 1.2

```
// An applet to print Hello World! //

      import  java.awt.Graphics;
      import java.applet.Applet;
      public class HelloWorld extends Applet {
            public void paint (Graphics g ) {
                  g.drawString ( " Hello World!" 50, 25 );
            }
      }
```

Practice 1.3

```
// Basic format of an HTML document //

< HTML>
  < HEAD>
      <TITLE> A sample HTML demonstration < /TITLE>
  </HEAD>

  <BODY>
      <H1> HTML Demo </H1>
      Here is a simple normal text
      <B> Here is some Bold text </B>
      <I> Here is some text in Italic </I>
  </BODY>
</HTML>
```

Practice 1.4

```
< HTML >
   <HEAD>
       <TITLE > An HTML Containing HelloJavaApplet </TITLE>
   </ HEAD>
```

```
   <BODY>
      <HI> Very simple web page Demo </HI>
        <APPLET CODE = "HelloJavaApplet.class"
                  WIDTH = 200    HEIGHT = 100 >
        </APPLET>
   </BODY>
<HTML>
```

Practice 1.5

```
      /*
       * One more simple Java Application *
       * This application computes square root *
      */
      // This is also a comment (one line comment)

      import java.lang.Math;
      class SquareRoot
      {
          public static void main (String args[ ]) {
            double x = 45;     // Variable declaration and initialization
            double y;    // Declaration of another variable
            y = Math.sqrt(x);
            System.out.println("Square root of "+ x +"=" + y);
          }
      }
```

Assignment 1.1

Write Java Applications to calculate the following:

(a) Factorial of n = 10

(b) Sum of the square of first 10 integers

(c) Calculation of an exponential series with x = 1

Solve the problems both iteratively as well as recursively

Practice 1.6

```
// Application with more than one classes   //

//   Edit PeopleApplin.java

      class FirstClass {
          intidNo;
          iIdNo = 555;
          public static void print(  ) {
             System.out.println ( " First Class citizen"  + idNo );
          }
      }
```

```
class SecondClass {
    int idNo;
    idNo = 111;
    public static void print(  ) {
        System.out.println ( " Second Class citizen " + idNo) ;
    }
}

public class PeopleAppln {
   FirstClass female;
   SecondClass male;
   public static void main( String args[ ] ) {
      System.out.print("People from Java World");
      female.print( );
      male.print( );
   }
 }
```

(This problem has a minor mistake. Identify the mistake and then write the correct code.)

Practice 1.6

```
// Edit the following program as HelloNoClass.java

public static void main (String args[ ] ) {
      System.out.println( "Hello Classless Java!]);
}
```

Type following two commands to run the HelloNoClass.java Application:

```
javac HelloNoClass.java       // To compile
java   HelloNoClass           // To run the program
```

Is the compilation successful? What is about the execution?

Practice 1.7

```
// An Application without any class and inputs are passed as command line

// Edit the program as HelloCommandLine.java


public static void main(String args[ ] ) {
    for(int i = 0; i < 3;i++)
      System.out.println( "Hello "+ args[i]);
}
```

Type following two commands to run the Hello.java Application :

```
javac HelloCommandLine.java         // To compile
java   HelloCommandLine C C++ Java  // To run the program
```

Practice 1.8

```
/*
 * This program passes inputs to the Application
 * through command line arguments
*/


class CommnadLineInputTest
{
   public static void main(String args[ ] ) {
      int count;
      String aString;
      count = args.length;

      System.out.println( "Number of arguments ="+ count);

      for(int i = 0; i < count; i++) {
          aString = args[0];
            System.out.println( "args["+i+"]"+"="+ aString);
      }
    }
}
```

Type following two commands to run the CommandLineInputTest.java Application :

```
javac CommandLineInputTest.java
java  CommandLineInputTest Kolkata Chennai Mumbai Delhi     Bangalore
```

Assignement 1.2

Write Java Applications to calculate the following:
(a) Factorial of n.
(b) Sum of the square of first n integers
(c) Calculation of an exponential for a given value of n.
In each case, read the value of n from the keyborad
(d) Read two integers a and b from the keyboard and then
print the largest value read.

Practice 1.9

```
/*
 * This is a Java Application to read principalAmount,
 * rateOfInterest and numberOfYears from the standard input
 * and then calculates the simpleInterest
*/

import java.io.*;
class InterestCalculator
{
  public static void main(String args[ ] ) {
     Float principalAmount = new Float(0);
     Float rateOfInterest = new Float(0);
     int numberOfYears = 0;
```

```
        DataInputStream in = new DataInputStream(System.in);
        String tempString;

        System.out.print("Enter Principal Amount: ");
        System.out.flush();
        tempString = in.readLine();
        principalAmount = Float.valueOf(tempString);

        System.out.print("Enter Rate of Interest: ");
        System.out.flush();
        tempString = in.readLine();
        rateOfInterest = Float.valueOf(tempString);

        System.out.print("Enter Number of Years: ");
        System.out.flush();

        tempString = in.readLine();
        numberOfYears = Integer.parseInt(tempString);

        // Input is over: calculate the interest
        int interestTotal = principalAmount*rateOfInterest*numberOfYears;

        System.out.println("Total Interest = " + interestTotal);
       }
    }
```

Assignment 1.3

Write a Java Application to read the name of a student (studentName), roll Number (rollNo) and marks (totalMarks) obtained. rollNo may be an alphanumeric string.
Display the data as read.

Practice 1.10

```
// Use of init( ) method in an applet //

import java.out .Graphics ;
import java .applet.Applet;

public class HelloWorld extends Applet {
      public void init( ) {
            resize(200,200);
      }
      public void paint (Graphics g ) {
      g.drawString ( " Hello World !", 50, 25 );
      }
}
```

Practice 1.11

```
// Use of  init( ) to pass value through HTML to applet //

   import java .awt . *;
   import java.applet. * ;

    public class RectangleTest extends applet {
      int x, y, w, h;
      public void init (  ) {
         x  = Integer.parseInt(get Parameter (" xValue" ));
         y  = Integer.parseInt(get Parameter (" yValue" ));
         w = Integer.parseInt(get Parameter (" wValue" ));
         h  = Integer.parseInt(get Parameter (" hValue" ));
      }

      public void paint ( Graphics g ) {
         g.drawRect (x, y, w, h );
        }
    }
```

Corresponding HTML document containing this applet and providing parameter values is mentioned as below :

```
      < applet code = " RectangleTest" width = 150  height = 100 >
      < param name = xValue  value = 20 >
      < param name = yValue  value = 40 >
      <param name  = wValue  value = 100>
      < param name = hValue  value = 50 >
      < /applet >
```

Practice 1.12

```
//Interactive input output

import java.io.*;
class InterestCalculator {
      public static void main(String args[]) {
            Float principleAmount = new Float(0);
            Float rateOfInterest = new Float(0);
            int numberOfYears = 0;
            String tempString= " ";
            try {
                  DataInputStream in = new DataInputStream(System.in);


                  System.out.print("Enter Principal Amount: ");
                  System.out.flush();
                  tempString = in.readLine();
                  principleAmount = Float.valueOf(tempString);

                  System.out.print("Enter Rate of interest: ");
                  System.out.flush();
```

```
                tempString = in.readLine();
                rateOfInterest = Float.valueOf(tempString);

                System.out.print("Enter no of years: ");
                System.out.flush();
                tempString = in.readLine();
                numberOfYears = Integer.parseInt(tempString);

                System.out.print("Enter a string: ");
                System.out.flush();
                tempString = in.readLine();
          } catch(Exception e) { }

          float interestTotal = (principleAmount.floatValue() *
rateOfInterest.floatValue() * numberOfYears)/100 ;
          System.out.println("Total Interest = " + interestTotal);
          System.out.println(tempString);
     }
}
```

Practice 1.13

```
//Interactiveinputoutput Writing to a file and then reading from a file


import java.util.*;
import java.io.*;

class Interactiveinputoutput {
     public static void main(String args[]) {
          float principleAmount= 0 ;
          float rateOfInterest = 0;
          int numberOfYears = 0;
          String tempString;
          String arrayString[] = new String[3];
          try {
                for(int i =0; i<2; i++) {
                DataInputStream din = new DataInputStream(System.in);
                DataOutputStream dos = new DataOutputStream(new
FileOutputStream("bank.txt"));


                System.out.print("Enter Principal Amount: ");
                System.out.flush();
                tempString = din.readLine();
                principleAmount = Float.valueOf(tempString).floatValue();

                System.out.print("Enter Rate of interest: ");
                System.out.flush();
                tempString = din.readLine();
                rateOfInterest = Float.valueOf(tempString).floatValue();

                System.out.print("Enter no of years: ");
                System.out.flush();
                tempString = din.readLine();
                numberOfYears = Integer.parseInt(tempString);
```

```
                System.out.print("Enter  string: ");
                System.out.flush();
                tempString = din.readLine();

                System.out.print("Enter the array of string: ");
                for(int j =0; j<3; j++) {
                System.out.flush();
                arrayString[j] = new String(din.readLine()); }


                dos.writeFloat(principleAmount);
                dos.writeFloat(rateOfInterest);
                dos.writeInt(numberOfYears);
                dos.writeChars(tempString);

                for(int k= 0; k<3; k++)
                {

                      dos.writeChars(arrayString[k]);

                      System.out.println(arrayString[k]);
                }

                dos.close();


                DataInputStream dis = new DataInputStream(new
FileInputStream("bank.txt"));


                float principleAmount1 = dis.readFloat() ;
                float rateOfInterest1 = dis.readFloat() ;
                int numberOfYears1 = dis.readInt();

                try
                {
                String tempString1= dis.readLine();
                System.out.println(tempString1);
                }
                catch(Exception e)
                {
                System.out.println("aa");
                }

                      //String arrayString1= dis.readUTF();

                //float interestTotal = (principleAmount1* rateOfInterest1*
numberOfYears1)/100 ;

                //Writting to console


                System.out.println();
                System.out.println("Principle Amount: " + principleAmount1);
                System.out.println("Rate Of Interest: " + rateOfInterest1);
                System.out.println("No Of Years: " + numberOfYears1);
```

```
                /*System.out.println("Comment: " + tempString1);
                //System.out.println(arrayString1);
                System.out.println();
                System.out.println("Total Interest = " + interestTotal);*/

            }
            } catch(Exception e) {
             System.out.println(e);
             }

      }
}
```

Practice 1.14

```
//Passing objects as parameter

class Test {
      int a,b;
      Test(int i, int j) {
            a = i;
            b = j;
      }

      boolean equals(Test o) {
            if((o.a == a) && (o.b == b)) return true;
            else return false;
      }
}


class PassObjectParameter {

      public static void main(String[] args) {
            Test ob1 = new Test(100,22);
            Test ob2 = new Test(100,22);
            Test ob3 = new Test(-1,-1);

            System.out.println("ob1 == ob2: " + ob1.equals(ob2));
            System.out.println("ob1 <> ob3: " + ob1.equals(ob3));


      }
}
```

Assignment 1.4

Define a class Employee to represent an employee in a typical organization. Create 10 employee objects from the keyboard. Make the object persistent, that is, store them in a file.

Write another Java program to retrieve the records of 10 employees and add three more employees into the list and delete 2 employee’s records from the list. Save the changes and ensure that the changes have been made perfectly.

****
Practice Sheet #2
****

Practice 2.1

```
  /*This application read values from the keyboard,
   * Store them into memory, and
   * Display the values stored in.                  */

import java.io.DataInputStream;

class PrimitiveDataTypes
{
  public static void main(String args[ ] ) {
     DataInputStream in = DataInputStream(System.in);
     boolean flag;
     char c;
     byte b;
     short shortInt;
     int i;
     long longInt;
     float x;
     double y;

     System.out.println("Enter a small integer (say 123): ");
     shortInt = System.in.read();

     System.out.println("Enter an Integer (say 12345): ");
     i = Integer.parseInt(in.readLine());

     System.out.println("Enter a Long Integer (say 12345689L): ");
     longInt = Long.parseLong(in.readLine());

     System.out.println("Enter a float number (say 123.45): ");
     x = Float.valueOf(in.readLine()).floatValue();

     System.out.println("Enter a double number (say 9.87e-007): ");
     y = Double.valueOf(in.readLine()).doubleValue();

     System.out.println("shortInt = " + shortInt);
     System.out.println("i = " + i);
     System.out.println("longInt = " + longInt);
     System.out.println("x = " + x);
     System.out.println("y = " + y);
  }
}
```

Assignment 2.1

(a) Define an one dimensional array "vivid" of type float. Read the values from the keyboard to load the array. Calcualte and then print the average of all the values in "vivid".
(b) Define two 2D arrays of integers, namely A[3]4] and B[4][2]. Store the values into them. Store the resul of matrix multiplication into an another 2D array, say C.

Assignment 2.2

Read at most 10 names of students and store them into an array
of String nameOfStudents[10].
Sort the names into the lexicographic order.
Display the sorted list of names.

Assignment 2.3

Define a class Employee with usual member for an employee like
empCode(String), empName(String), dateOfBirth(Date), dateOfJoin(Date)
designationCode(int), salary(float).
Create a list to store data about 10 employees using Vector.
Manipulate the list using the methods in class Vector.

Practice 2.2

```
// Encapsulation  :- Defining a class//

    class Point
    {
      int x,y;
      void getPoint ( int a, int b, ) {
            x = a;
            y = b;
      }
    }

// definition of another class. This is a main class

    class Points
    {
      float distance;
      public static void main (String args[ ] {
          Point p1 = new Point( );
          Point p2 = p1;
          Point p3 = new Point ( );
          Point p4 = new Point ( );
          p1.getPoint (5, 10 );
          p2.getPoint (15, 20);
          p3.getPoint (20, 30);
          p4.getPoint (30, 40);
          System.out.println (" X1 = " + p1.x +  "Y1  = " + p1.y );
          System.out.println ("X2=" + p2.x + "Y2 = " +p2.y );
          int dx = p3.x - p4. x;                        // X2 - X1
          int dy = p3.y - p4. y;                    // y2 - y1
          distance = Math.sqrt (dx * dx + dy * dy );    // (X2-X1)2 + (Y2-Y1)2
          System.out.println ( " Distance = "+ distance );
      }
     }
```

Assignment 2.4

(a) Modify the class definition Points to include a method
float distnaceCalculate(Point p1, Point p2);
This method will calculate the distance between two points p1 and p2 and return the resul then.

(b) Define a class Complex to represent an object for a complex number like Z = X + i.Y with the following methods:

```
Complex add(Complex z1, Complex z2)   // To add two complex numbers
Complex sub(Complex z1, Complex z2)   // To subtract two complex numbers
Complex mul(Complex z1, Complex z2)   // To multiply two complex numbers
float magnitude(Complex z)          // To find the modulus
Complex conjugate(Complex z)        // To find the complex conjugate
```

Write the main class and instantiate the objects of the above mentioned classes

Practice 2.3

```
// Automatic initialization - concept of constructor

class Point (  )  {
      int x, y;
      Point ( int x, int y ) {
         this.x = x ;
         this.y = y;
      }
      printPoint() {
          System.out.println("X = "+ this.x + " Y= " + this.y);
}

class PointCreate {
      public static void main ( String args [  ] ) {
            Point p = new Point (10, 20 );
            p.printPoint();
      }
}
```

Assignment 2.5

Add the necessary methods in the class PointCreate (Practice 2.3) to calculate the area and perimeter of a rectangle given the two corner coordinates.

Practice 2.4

```
// Overloading constructor: Initializing object in several ways.

class Point (  )  {
      int x, y;
      Point(int a, int a) {         // Constructor for integer coordinates
          x = a;
          y = b;
      }
```

```
        Point(float x, float y) {     // Constructor for real coordinates
            this.x = x;
            this.y = y;
        }

        printPoint() {
            System.out.println("X = "+ this.x + " Y= " + this.y);
    }

    class PointCreate {
        public static void main ( String args [  ] ) {
              Point p1 = new Point (10, 20 );
              p1.printPoint();
              Point p2 = new Point (5.5, 4.2 );
              p2.printPoint();
        }
    }
```

Practice 2.5

```
// An Practice of inheritance //

    class Point {
        int x;
        int y;
        Point (  ) {           // A  default constructor
           this (0,0);
        }

        Point (int x,  int y ) {      // Overloaded constructor
           this.x = x;
           this.y = y;
        }
    }

    class Point3D extends Point {
        int z;                      // another ordinate for  a 3D Point //
        Poin3D ( ) {
            this ( 0, 0, 0);
        }

        Point 3D(int x, int y, int z ) {
            super (x, y );           // called the superclass constructor
            this.z = z;
        }

    void display (  )    {             // A method definition
        System.out.println (" X = "+ x + " Y = " y + " Z = " + z );
    }

    class Points {                // Main class
        public static void main ( String args[] ) {
            point p1 = new Point(4, 6) ;                    // A 2-D point
            point3D p2 = new Point3D(2,3,5);          // A 3-D point
            point3D p3 = new Point3D( );             // Another 3-D point
```

```
            System.out.println ("P1 : X = " + p1.x + " Y =" + p2.y ) ;
            p2.display ( ) ;
            p3.display ( ) ;
        }
    }
```

Practice 2.6

```
// Code sharing through super concept //

  class Cat {
      void speak (  )   {
      System. Out . println("Meaon ! ");
  }


  class PetCat extends Cat {        // PetCat is one type of Cat
      void speak ( ) {
            System.out println ( " Meow ! " );
      }
  }

  class MagicCat extends Cat {            // MagicCat is another kind of Cat
      static boolean noOne ;
      void speak (  ) {
            if (noOne )
                  super.speak (  );       // use the super class definition
            else  System.out.println ( " Hello World !");
      }
   }

  class ManyCats {
      public static void main ( String args[ ]  ) {
            Pet Cat c1 = new PetCat (  );
            MagicCat c2 = new MagicCat( );
            c2.noOne = true;
            c2.speak (  );
            c1.speak (  );
            c2.noOne = false ;
            c2.speak (  );
      }
   }
```

Practice 2.7

```
// Polymorphism concept //

      class Point {
            int x,y;
            Point ( int x, int y ) {            //  It is a constructor
                  this.x = x;
                  this.y = y;
            }
```

```
/* P1  */   float distance ( int x, int y)      {     // One definition of distance
                int  dx = this x -  x;
                int  dy = this y - y;
                return float Math.sqrt ( dx* dx + dy * dy );
            }

/* P2*/     float distance (Point p) {          // Overloaded definition of distance .
                return distance ( p.x,  p.y) ;
                   }

      Class Point3D extends Point {
            int z ;
            Point3D ( int x, int y, int z ) {    // Constructor of Point3D
                  super ( x, y );
                        this.z = z;
            }

/*P3 */     float distance (int x,int y, int z )  {// Another definition of distance
                int dx = this.x - x;
                int dy = this.y - y;
                int dz = this.z - z ;
                return  (float) Math.sqrt ( dx * dx + dy*dy + dz*dz );
            }

/*M4 */     float distance (Point3D pt) {
               return distance (pt.x, pt.y, pt.z );
            }
      }

      class PointDistance {
            public static  void main ( String args [  ] ) {
               Point p1 = new Point (10, 5) ;               // 2-D point
               Point  p3 = new Point3D (5, 10, 5)           // 3-D point
               Point p2 = new Point (4, 1) ;                // another 2-D point
               Point p4 = new Point3D ( 2,3,4 );            // another 3-D point
               float d0 = pl.distance ( 0,0);          // P1 will be referred
               float  d1 = p1.distance ( p2);           // P2 will be referncd
               System.out.println ( " Distance from P2 to Origin = " + d0);
               System.out .println ( " Distance from P2 to P1 = " + d1)
               d0 = p3.distance (0,0,0);                // P3 will be referred
               d1 = p4.distance (p3);                  // P4 will be referred
               System.out.println ("Distance from P3 to Origin = " + do);
               System.out println ( "Distance from P3 to P4= "+ d1);
            }
      }
```

Assignment 2.6

Define three different classes namely, Student, Staff and Faculty to define three different types of personlas. These classes are inherited from a super class say, Human. Human has member elements like name, dateOfBirth, father's name. It has its own constructor as well as a method to calcualte age on date.

Students has data like rollNo, branch, rank, dateOfGraduation. It has a method to calculate ageOfScholar on date.

Staff has data like dateOfJoin, designation, salary. It has a method to calcualte age of superanuation.

Similarly, faculty has same data like staff, in addition to those it has qualification,researchGuided and dateOfPromotion. It has a method to calculate promotionDue.

Implements all methods as overloading method, namely vitalMark.
Instantaite few objects for each of the class in the main class and then store vitalDates against the correwsponding name into a list using Vector.

Assignment 2.7

Develop a library information system with the following piece of code as the base code. You should have a UML design prior to this.

```
//Version 1: A simple version of Library Information System example

class LibMember {
	String name;
	int IDno;
	String Bookissued[] = new String[10];
	private int tos = -1;
	LibMember(String n, int i) {
		name = n;
		IDno = i;
	}
	void issueRequest(LibBook b) {
		System.out.println("You want to issue a book");
		if(tos < 4)
			b.issueBook(this);
		else
			System.out.println("You cannot issue any more book");
	}

	void issueUpdate(LibBook b) {
		System.out.println("The book tittled " + b.nameOfBook +" is entered
in your record");
		Bookissued[++tos] = b.nameOfBook;
	}

	void displayBooks() {
		System.out.println("List of books for Library member " + this.name
);
		for(int i =0; i<=tos; i++)
			System.out.println(Bookissued[i]);
	}
}

class LibBook {
	String nameOfBook;
	int ACCno;
	LibBook(String n, int a) {
		nameOfBook = n;
		ACCno = a;
	}
```

```
	void issueBook(LibMember l) {
		System.out.println("The book is issued");
		l.issueUpdate(this);
	}
}



class LibraryExample {
	public static void main(String args[]) {
	LibMember l = new LibMember("Tom", 111);
	LibBook b = new LibBook("Math", 1000);
	l.issueRequest(b);
	l.displayBooks();
	System.out.println("Your job is over");
	}
}



//Library example to show abstract class and inheritance

abstract class LibBook {
	String nameOfBook;
	int ACCno;

	LibBook(String n, int a) {
		nameOfBook = n;
		ACCno = a;
	}

	abstract void issueBook(LibMember l);
	abstract double computeFine(int i);


}


class TextBook extends LibBook {
	TextBook(String n, int a) {
		super(n,a);
	}

	void issueBook(LibMember l) {
		System.out.println("You can issue for 30 days");
		l.issueUpdate(this);
	}

	double computeFine(int days) {
		return days*2;
	}
}
```

```
class RefBook extends LibBook {
      RefBook(String n, int a) {
            super(n,a);
      }


      void issueBook(LibMember l) {
            System.out.println("You can issue for 1 night only");
            l.issueUpdate(this);
      }


      double computeFine(int days) {
            return days*10;
      }
}


class LibMember {
      String name;
      int IDno;
      String Bookissued[] = new String[10];
      int day;
      private int tos = -1;

      LibMember(String n, int i) {
            name = n;
            IDno = i;
      }
      void issueRequest(TextBook b) {
            System.out.println("You want to issue a text book!!");
            if(tos < 4) {
                  day = 30;
                  b.issueBook(this);
                  }
            else
                  System.out.println("You cannot issue any more book");
      }

      void issueRequest(RefBook b) {
            System.out.println("You want to issue a Reference book!!");
            if(tos < 4) {
                  day = 1;
                  b.issueBook(this);
                  }
            else
                  System.out.println("You cannot issue any more book");
      }
      void issueUpdate(LibBook b) {
            System.out.println("The book tittled " + b.nameOfBook +" is entered
in your record");
            Bookissued[++tos] = b.nameOfBook;
      }

      void displayBooks() {
            System.out.println("List of books for Library member " + this.name
);
```

```
            for(int i =0; i<=tos; i++)
                  System.out.println(Bookissued[i]);
      }

      void displayRecords() {
            System.out.print("Name :  " + this.name + "   IdNo:  " + IDno + "
Issued for:  " + day + " days   Books: ");
            for(int i =0; i<=tos; i++)
                  System.out.print(Bookissued[i]+ ", ");
            System.out.println();
      }
}



class LibExaample1 {
      public static void main(String args[]) {
      LibMember l1 = new LibMember("Tom", 111);
      LibMember l2 = new LibMember("Harry", 112);

      TextBook b1 = new TextBook("Math", 1000);
      TextBook b2 = new TextBook("Science", 1001);

      RefBook r1 = new RefBook("Journal1", 2000);
      RefBook r2 = new RefBook("Journal2", 2001);

      l1.issueRequest(b1);
      l1.issueRequest(r1);
      l2.displayBooks();
      l1.displayRecords();

      }
}


//Version 2: Another version of Librray Information System taking the current
date to calculate fine as well as to issue a book


import java.util.*;
import java.io.*;

class Date1 {
      int day;
      int month;
      int year;

      Date1 getDate() {
                  String tempString;
                  try {
                        DataInputStream din = new DataInputStream(System.in);

                        System.out.print("Enter Current Day: ");
                        //System.out.flush();
                        tempString = din.readLine();
                        day = Integer.parseInt(tempString);
```

```
                        System.out.print("Enter Current Month: ");
                        //System.out.flush();
                        tempString = din.readLine();
                        month = Integer.parseInt(tempString);

                        System.out.print("Enter Current Year: ");
                        //System.out.flush();
                        tempString = din.readLine();
                        year = Integer.parseInt(tempString);
                  }
                  catch(Exception e) {
                        System.out.println("Here is the error");
                  }
                  return(this);

      }
}

abstract class LibBook {
      String nameOfBook;
      int ACCno;
      String issuedTo;

      LibBook(String n, int a, String m) {
            nameOfBook = n;
            ACCno = a;
            issuedTo = m;
      }

      boolean search(LibBook b) {
            if(b.issuedTo == "NULL") return true;
            else return false;
      }

      void returnBook(LibMember l) {
            issuedTo = "NULL";
            l.returnUpdate(this);
            }
      abstract void issueBook(LibMember l);
      abstract int computeFine(int i);
}


class TextBook extends LibBook {

      TextBook(String n, int a, String m) {
            super(n,a, m);
      }

      void issueBook(LibMember l) {
            if(search(this)) {
                  System.out.println("You can issue for 30 days");
                  issuedTo = l.name;
                  l.issueUpdate(this);
                  }
            else
```

```
                System.out.println("The book is issued to someone else");
        }

        int computeFine(int days) {
                return days*2;
        }

        void displayTextBook() {
                System.out.println("Book: " +nameOfBook + "         AccNo: " + ACCno
+ "         Issued To : " + issuedTo);
        }
}



class RefBook extends LibBook {

        RefBook(String n, int a, String m) {
                super(n,a, m);
        }


        void issueBook(LibMember l) {
                if(search(this)) {
                        System.out.println("You can issue for 1 night only");
                        issuedTo = l.name;
                        l.issueUpdate(this);
                        }
                else
                        System.out.println("The book is issued to someone else");
        }



        int computeFine(int days) {
                return days*10;
        }

        void displayRefBook() {
                System.out.println("Book: " +nameOfBook + "          AccNo: " +
ACCno + "        Issued To : " + issuedTo);
        }
}



class LibMember {
        String name;
        int IDno;
        String bookissued[] = new String[10];

        Date1 curDate = new Date1();
        Date1 retDate = new Date1();
        Date1 retDate1[] = new Date1[10];
        private int tos = -1;
```

```
LibMember(String n, int i) {
      name = n;
      IDno = i;
}

void issueRequest(TextBook b) {
      System.out.println("You want to issue a text book!!");
      if(tos < 3) {
            curDate.getDate();
            retDate.day = curDate.day+30;
            if(retDate.day > 30) {
            retDate.day = retDate.day - 30;
            retDate.month = curDate.month + 1;
            }
            else
            retDate.month = curDate.month;
            retDate.year = curDate.year;
            b.issueBook(this);
            }
      else
            System.out.println("You cannot issue any more book");
}

void issueRequest(RefBook b) {
      System.out.println("You want to issue a Reference book!!");
      if(tos < 3) {
            curDate.getDate();
            retDate.day = curDate.day+1;
            if(retDate.day > 30) {
            retDate.day = retDate.day - 30;
            retDate.month = curDate.month + 1;
            }
            else
            retDate.month = curDate.month;
            retDate.year = curDate.year;
            b.issueBook(this);
            }
      else
            System.out.println("You cannot issue any more book");
}

void issueUpdate(LibBook b) {
      tos++;
      retDate1[tos] = new Date1();
      retDate1[tos].day = retDate.day;
      retDate1[tos].month = retDate.month;
      retDate1[tos].year = retDate.year;
      bookissued[tos] = b.nameOfBook;
}

void returnUpdate(LibBook b) {
      curDate.getDate();
      int i =0, j =0;
      while(bookissued[i] != b.nameOfBook) {
            i = i+1;
      }
```

```
            j = curDate.month - retDate1[i].month;
            if (j > 0)
            System.out.println("Total fine : "+ b.computeFine(j));
            else
            System.out.println("No fine");
            bookissued[i] = "NULL";
            r
etDate1[i].day = 0;
            retDate1[i].month = 0;
            retDate1[i].year = 0;

      }



      void displayRecords() {
            System.out.println("Name :  " + this.name + "   IdNo:  " + IDno );
            System.out.println("   Return Date:  "  + "   Books: ");
            for(int i =0; i<=tos; i++){
                  System.out.println("    "+ retDate1[i].day + "/" +
retDate1[i].month + "/" + retDate1[i].year +"         " +bookissued[i]);
            System.out.println();
            }

      }
}


class LibExaample2 {
      public static void main(String args[]) {
      LibMember l1 = new LibMember("Tom", 111);
      LibMember l2 = new LibMember("Harry", 112);

      TextBook b1 = new TextBook("Math   ", 1000, "NULL");
      TextBook b2 = new TextBook("Science", 1001, "NULL");
      TextBook b3 = new TextBook("English", 1002, "NULL");

      RefBook r1 = new RefBook("Journal1", 2000, "NULL");
      RefBook r2 = new RefBook("Journal2", 2001, "NULL");
      RefBook r3 = new RefBook("Journal3", 2002, "NULL");

      l1.issueRequest(b1);
      l1.issueRequest(r1);
      l1.issueRequest(b2);
      //l1.issueRequest(b3);
      //l1.issueRequest(r2);

      //l2.issueRequest(b1);
      //l2.issueRequest(r1);
      //l2.issueRequest(b2);
      //l2.issueRequest(b3);
      //l2.issueRequest(r2);

      //b1.displayTextBook();
      //r1.displayRefBook();
```

```
        l1.displayRecords();

        b1.returnBook(l1);
        l1.displayRecords();
        r1.returnBook(l1);
        //l1.returnRequest(b2);

        l1.displayRecords();
        //l2.displayRecords();

        }
}
```

Assignment 2.8

You are to upgrade the above problem incorporating the following things.

1. Three types of library members: student, staff and faculty
2. Allow maximum 3 books to students for maximum 30 days. Allow 6 books to staffs for maximum 90 days. Allow 10 books to faculty members for 180 days.
3. Issue notices to the members whose date of return is just 7 days behind.
4. Allow book reservation facility and if a book is reserved by more than one members then resolve the issue with priority student > staff > faculty or the first-come-first-served for the same member category.

****
Laboratory Practice Sheet #3
****

Practice 3.1

```
// A simple interface in Java

    interface GeoAnalyzer
    {
      final static float pi = 3.142F;
      float area( );
      float perimeter( );
    }


    class Circle implements GeoAnalyzer
    {
        float radius;

      Circle(float r) {

          radius = r;
      }

      public float area( ) {
            return(pi*radius*radius);
      }

      public float perimeter( ) {
            return(2*pi*radius);
      }
    }

    class Ellipse implements GeoAnalyzer
    {
        float major;
      float minor;

      Ellipse(float m, flaot n) {
          major = m;
          minor = n;
      }

      public float area( ) {
            return(pi*major*minor);
      }

      public float perimeter( ) {
            return(pi*(major+minor);
      }

    }

    class Rectangle implements GeoAnalyzer
    {
```

```
    float length;
  float width;

  Rectangle(float l, float w) {
      length = l;
       width = w;
  }

  public float area() {
        return(length*width);
  }

  public float perimeter( ) {
        return(2*(length+width));
  }

}


class Geometry
{

   static void display(float x, float y) {
    System.out.println("Area = " + x + "Perimeter = " + y);
   }

   public static void main(String args[ ]) {
    Circle c = new Circle(5.2);
    Ellipse e = new Ellipse(4.5, 3.6);
    Rectangle r = new Rectangle(6.5, 4.3);

    GeoAnalyzer geoItem;

    geoItem = c;
    display(geoItem.area(), geoItem.perimeter());

    geoItem = e;
    display(geoItem.area(), geoItem.perimeter());

    geoItem = r;
    display(geoItem.area(), geoItem.perimeter());
   }
}
```

Assignment 3.1

Consider an interface memberIEEE with yearMembership and regionIEEE are two fields and memberProfile being a method. Two classes namely, Teacher and Students are to implement the interface. Again, Tecaher inherits from another class say, Employee.

With suitable member and methods in the three classes, write a program exploiting the interface mechanism.

Assignment 3.2

//Example of Interface in GeoObjects. Here vectors and enumeration are used and also interactive input. Best on the code given below, organize all classes into a package called “GeoAnalyzer” and develop a tool suitable for geometric object manipulation.

```
import java.util.Enumeration;
import java.util.Vector;
import java.lang.Math; // for the definition of pi in area computation

interface Printable {
public void printIt();
}


class GeoObject implements Printable {
protected int baseX;
protected int baseY;
protected int itsArea;

public GeoObject (int x, int y) {
baseX = x;
baseY = y;
itsArea = 0;
}


public void printIt(){
System.out.println("The default print: " + this);
}


}


class Rectangle extends GeoObject {
private int itsWidth;
private int itsHeight;

public Rectangle(int x, int y, int width, int height) {
super(x,y);

if (width<0) {
itsWidth = 0;
}

else itsWidth = width;

if (height<0)
itsHeight = 0;
else
itsHeight = height;
itsArea = itsHeight*itsWidth;
}
```

```
public void printIt () {
System.out.println("Rectangle: base = (" +
baseX + ", " + baseY + "), height = " +
itsHeight + ", width = " + itsWidth +
", area = " + itsArea + ".");
}


}



class Circle extends GeoObject {
private int itsRadius;

public Circle(int x, int y, int radius) {
super(x,y);
java.lang.Long results;
itsRadius = radius;
results =
new java.lang.Long(
java.lang.Math.round( 2 * radius
* radius
* java.lang.Math.PI));
itsArea = results.intValue();
}


public void printIt () {
System.out.println("Circle: base = (" +
baseX + ", " + baseY + "), radius = " +
itsRadius + ", area = " + itsArea + ".");
}


}



public class GeoWorld {
public static void main (String args[]) {

System.out.println("Starting GeoWorld");
System.out.println();
System.out.println("Type c to create a circle, " +
"r to create a rectangle, " +
"and q to quite.");

// read in the first character
// if anything goes wrong (including "end of file")
// set the "character" to -99 as a signal that
// this there is no more input available

int in;
try {
```

```
in = System.in.read();
} catch (Exception e) {
in = -99;
}


int count = 0;

Rectangle r;
Circle c;
Object o;

Vector objects = new Vector();

// check for the end of data, create the objects indicated (if any),
// put the new object on the list, and get the next
// character from the input

while (in != 'q' && in != -99) {

// for an r: create a rectangle and add it to the list

if (in == 'r') {

// In place of the following three lines, I could have typed:
// objects.addObject((Object) new Rectangle(20,30,40,50);
// and this would have the same effect. In that case the
// data field r would be unnecessary. I similar substitution
// would be possible for the circle below.

r = new Rectangle(20,30,40,50);
o = (Object) r;
objects.addElement(o);

System.out.println("Created a Rectangle");

count = count + 1;
}


// for a c: create a circle and add it to the list

if (in == 'c') {
c = new Circle(60,70,80);
objects.addElement((Object) c);
count = count + 1;
System.out.println("Created a Circle");
}


// read in a new character and repeat the while
try {
in = System.in.read();
} catch (Exception e) {
in = -99;
}
```

```
 } // end of loop that starts with "while (in != -99)..."

 // now print out all the object we created
 // note that we no longer care what type of object each one is
 // as far as we know, at this point, they are all objects
 // of the GeoObject class (which implements Printable)
 // so we know that they must have a printIt method.

 Enumeration theList = objects.elements();
 GeoObject element;

 for (int i = 1; i <= count; i++) {
 element = (GeoObject) theList.nextElement();
 System.out.println(
 "Printing object number " + i + " from the list");
 element.printIt();
 System.out.println();
 }

 }
 }
```

Practice 3.2

```
// Practice of Interface sharing member elements //

interface Gender {
      int  FEMALE = 0;
      int  MALE = 1;
}

class Enquire implements Gender  {
      int ask (  ) {
            System.out.println ( " Type you gender : :  Female = 0 or Male = 1"
) ;
            int type = System.in.read (  );
      }
}

class Response implements Gender {
      static void reply (int type ) {
            switch (type ) {
                  case MALE :
                        System.out .println (" Hello Sir !");
                        break;
                  case  FEMALE :
                        System.out.println ( " Hi  Mam !! " );
                        break;
                  default : System.out. println ( " Oh God ! " ) ;
            }
      }

      public static void main ( String args [  ] ) {
            Enquire  = new Enquire( );
            Gender person = e;
```

```
            reply (person.ask ( ));             // Ask to first kind
            reply (person.ask( ) );             // Ask to second kind
            reply ( person.ask( ) ) ;           // Ask to third kind
      }
}

interface Constants {
    double velOfLight = 3.0e+10
    String unitVelOfLight = "m/s";
    ....  ....  ....  ....
}

interface Physics {
    void quantumLaw();
    ... ... ... ...
}


interface lawOfPysics extends Constants, Physics
{
    ..... ..... ...... .....
    ..... ..... ...... .....
}
```

Practice 3.3

```
// User defined Package //

    package MyPackage;
    public class MyClass {
      public void test ( ) {
            System.out.println ( " Welcome to My Class !");
      }
    }

// Save this listing in file called MyClass.java.
// This file should be located in a subdirectory named MyPackage
// Compile the MyClass.java to get MyClass.class

// Import the so created package with the following code.

    import MyPackage.MyClass;
    class PackageTestAppln {
      public static void main ( String args [ ] ) {
          MyClass theClass = new MyClass ( );
            theClass.test ( );
      }

    }
```

Practice 3.4

```
// Static binding of a variable or method //

    class StaticClass {
      static int count;
      StaticClass (  ) {
```

```
        count ++ ;
    }

    public static void printCount (  ) {            // It is a static method
        System.out.println ( " Count = "  + count );
    }

  class  StaticTest {
    System.out. println ( " Initialization of  Static member " );
    count = 0;

    public static void main ( String args [  ] ) {
        StaticClass x, y;

    x.printCount (  ) ;
    y.printCount (  ) ;
    StaticClass.printCount (  );  // A Static method can  be called this way

    x  = new StaticClass (  );
    printCount (  ) ;             // What will be the value of count?
    y = new StaticClass ( ) ;
    printCount (  ) ;             // What will be the value of count?
          }
    }
```

Assignment 3.3

Observe the execution and its outcome without declaring count and printCount( ) as static.

Practice 3.5

```
// Method resolution during execution //

   class A {
      void callMe (  ) {
          System.out. println ( " I am from A ") ;
      }
   }

   class B  extends A {
      void callMe (  ) {
         System.out.println ( " I am from B ");
      }
   }

   class Who {
      public void static main (String args [  ] ) {
         A  a = new B (  ) ;
         a.callMe (   );
      }
   }
```

****
Laboratory Practice Sheet #4
****

Practice 4.1

```
// Practice of a multithreaded program using subclassing Thread

class ThreadA extends Thread
{
     public void run( ) {
      for(int i = 1; i <= 5; i++) {
         System.out.println("From Thread A with i = "+ i);
      }
      System.out.println("Exiting from Thread A ...");
     }
}

class ThreadB extends Thread
{
     public void run( ) {
      for(int j = 1; j <= 5; j++) {
         System.out.println("From Thread A with j = "+ j);
      }
      System.out.println("Exiting from Thread B ...");
     }
}

class ThreadC extends Thread
{
     public void run( ) {
      for(int k = 1; k <= 5; k++) {
         System.out.println("From Thread A with k = "+ k);
      }
      System.out.println("Exiting from Thread C ...");
     }
}


class MultiThreadClass
{
    public static void main(String args[]) {
      ThreadA a = new ThreadA();
      ThreadB b = new ThreadB();
      ThreadC c = new ThreadC();

      a.start();
      b.start();
      c.start();

      System.out.println("... Multithreading is over ");
    }
}
```

Assignment 4.1

Develope the following program. If it is not truly multithreading
do the necessary to make it multithreading.

```
// Creating and running  threads using sub classing Thread //

      class TestThread extends Thread {
            private String whoAmI ;
            private int delay ;

            TestThread (String s, int  d ) {
                  whoAmI = s;
                  delay  = d ;
            }

          public void run( ) {
            sleep (delay);
            System.out.println (" Process is interrupted " );
            System.out.println ( " Hello World ! " +
                  whoAmI + " Slept : " + delay);
          }
      }


      public  class MultiThreadTest       {
           public static void main (String[ ] args ) {
            TestThread t1, t2, t3;
            t1 = new TestThread ("Thread1", (int) (Math.random( ) * 2000 ) );
            t2 = new TestThread ("Thread2", (int) (Math.random( ) * 2500 ) );
            t3 = new TestThread ("Thread3", (int) (Math.random( ) * 3000 ) );
            t2.start( );
            t1.start( );
            t3.start( );
                }
      }  // end of class MultThreadTest.
```

Practice 4.2

```
// Practice of a multithreaded program using Runnable interface

class ThreadX implements Runnable
{
     public void run( ) {
        for(int i = 1; i <= 5; i++) {
           System.out.println("Thread X with i = "+ i);
        }
       System.out.println("Exiting Thread X ...");
     }
}

class ThreadY implements Runnable
{
     public void run( ) {
         for(int j = 1; j <= 5; j++) {
```

```
            System.out.println("Thread Y with j = "+ j);
         }
        System.out.println("Exiting Thread Y ...");
     }
}


class ThreadZ implements Runnable
{
     public void run( ) {
         for(int k = 1; k <= 5; k++) {
             System.out.println("Thread Z with k = "+ k);
         }
             System.out.println("Exiting Thread Z ...");
     }
}


class MultiThreadRunnable
{
    public static void main(String args[]) {
         ThreadX x = new ThreadA(); Thread t1 = new Thread(x);
         ThreadY y = new ThreadY(); Thread t2 = new Thread(y);
         ThreadZ z = new ThreadZ(); Thread t3 = new Thread(z);
         t1.start();
         t2.start();
         t3.start();
         System.out.println("... Multithreading is over ");
    }
}
```

Assignment 4.2

Check whether the following code intended for multithreading using the Runnable interface is serving the purpose or not. If not do the necessary modification.

```
// Two processes thread using Runnable interface //

      class Brother implements Runnable {
            int age = 0;
            String name;
            Brother(int age, string name ) {
                  this.age  = age ;
                  this.name = name;
            }

          public void run( ) {
            Thread.sleep ( age * 1000);
            System.out.println ( "Brother " +name + "age :" +age + "seconds" );
          }
      }
```

```
class Sister implements Runnable {
      int  spring;
      String sweety;
      Sister (int a, String b ) {
            spring  = a;
            sweety = b;
      }

    public void run ( ) {
      System.out.println ("Hi  Sweety" + sweety + "!");
            Thread.sleep ( spring * 1000);
            System.out.println("Spring "  +  spring  + "elapsed !");
    }
}

class RunnableThreads {
    public static void main(String[ ] args) {
      Brother ravi = new Brother (16, Ravi");
      Thread t1 = new Thread (ravi);
      t1.start ( ) ;
      Sister bobby = new Sister(10, "Bobby" );
      Thread t2= now Thread (bobby);
      t2.start ( );
    }
}
```

Practice 4.3

Following Java application create a list of numbers and then sort in ascending order as well as in descending order simultaneously.

```
//Generate n numbers between x and y randomly and store them in an array.
// Concurremtly do the following:
// -- Sort the numbers in ascending order and print the result.
// -- Sort the numbers in descending order and print the result.
// Next, concurrently do the following.
// -- Generate a random number between x and y and insert it into the original
//    list, whether it is there or not.
// -- Generate a random number between x and y and delete the number from the
//    list if it is there.


import java.util.*;

class Numbers {
      public int result[] = new int[10];
      void displayListOfNos() {
            System.out.println("Numbers stored in the array:");
            for( int idx=0; idx<10; ++idx) {
                  System.out.println(result[idx]);
            }
      }

      void fillTheArray(int aUpperLimit, int aArraySize) {
            if (aUpperLimit <=0) {
```

```
                  throw new IllegalArgumentException("UpperLimit must be
positive: " + aUpperLimit);
            }

            if (aArraySize <=0) {
                        throw new IllegalArgumentException("Size of returned
List must be greater than 0.");
            }

              Random generator = new Random();
            for( int idx=0; idx<aArraySize; ++idx) {
                        result[idx] = generator.nextInt(aUpperLimit);
            }

            displayListOfNos();

      }


      synchronized void sortAscending() {
            for(int i=0; i<9; i++) {
                  for(int j=i+1; j<10; j++) {
                        if(result[i] < result[j]) {
                              int temp = result[i];
                              result[i] = result[j];
                              result[j] = temp;
                        }
                  }
            }

            displayListOfNos();
      }



      synchronized void sortDescending() {
            for(int i=0; i<9; i++) {
                  for(int j=i+1; j<10; j++) {
                        if(result[i] > result[j]) {
                              int temp = result[i];
                              result[i] = result[j];
                              result[j] = temp;
                        }
                  }
            }

            displayListOfNos();
      }
}


class ArrangementAscending implements Runnable {
      Numbers n1 ;
      ArrangementAscending(Numbers n) {
      n1 = n;
            new Thread(this).start();
      }
```

```
      public void run() {
            n1.sortAscending();
      }
}



class ArrangementDescending implements Runnable {
      Numbers n2;
      ArrangementDescending(Numbers n) {
      n2 = n;
            new Thread(this).start();
      }


      public void run() {
            n2.sortDescending();
      }
}


class ArrangingNos {

      public static void main(String args[]) {
      Numbers n = new Numbers();
      n.fillTheArray(20,10);
      ArrangementAscending  a1 = new ArrangementAscending(n);
      ArrangementDescending d1 = new ArrangementDescending(n);
      }
}
```

Practice 4.4

```
//Randomly fill the array, sort it, add a number to the sorted list and delete a
number. All the process should be done concurrently.

import java.util.*;

class Numbers {
      public int result[] = new int[20];
      void displayListOfNos() {
            for( int idx=0; idx<10; ++idx) {
                  System.out.println(result[idx]);
            }
      }

      void fillTheArray(int aUpperLimit, int aArraySize) {
            if (aUpperLimit <=0) {
                  throw new IllegalArgumentException("UpperLimit must be
positive: " + aUpperLimit);
            }

            if (aArraySize <=0) {
                        throw new IllegalArgumentException("Size of returned
List must be greater than 0.");
            }
```

```
            Random generator = new Random();
          for( int idx=0; idx<aArraySize; ++idx) {
                      result[idx] = generator.nextInt(aUpperLimit);
          }

          System.out.println("Numbers stored in the array:");
          displayListOfNos();

    }



    void sortAscending() {
          for(int i=0; i<9; i++) {
                for(int j=i+1; j<10; j++) {
                      if(result[i] > result[j]) {
                            int temp = result[i];
                            result[i] = result[j];
                            result[j] = temp;
                      }
                }
          }

          System.out.println("Array after sorting:");
          displayListOfNos();
    }


    synchronized void deleteANumber(int aUpperLimit, int aArraySize) {
          if (aUpperLimit <=0) {
                 throw new IllegalArgumentException("UpperLimit must be
positive: " + aUpperLimit);
          }


           Random generator = new Random();
           int num =  generator.nextInt(aUpperLimit);
           int i = 0;
           while(i < aArraySize && num != result[i]){
                i = i+1;
           }

           if (i== aArraySize)
                System.out.println("Number " + num + " not found in the
array");
           else {
                for(int j = i; j< aArraySize-1; j++)
                      result[j] = result[j+1];
                      result[aArraySize - 1] = 0;
                      System.out.println("Number " + num + " found in the
array and deleted");
          }

          System.out.println("Array after deletion");
          displayListOfNos();
    }
```

```
      synchronized void addANumber(int aUpperLimit, int aArraySize) {
            if (aUpperLimit <=0) {
                  throw new IllegalArgumentException("UpperLimit must be
positive: " + aUpperLimit);
            }


             Random generator = new Random();
             int num =  generator.nextInt(aUpperLimit);
             int i =0;
             while(i < aArraySize && num > result[i]){
                  i = i+1;
             }

             if (i== aArraySize)
                  result[aArraySize] = num;
            else {
                  for(int j = aArraySize; j>i; j--)
                        result[j] = result[j-1];
                        result[i] = num;
            }

            System.out.println("Array after addition of " +num );
            displayListOfNos();
            System.out.println(result[aArraySize]);

      }
}


class AddingANumber implements Runnable {
      int ul;
      int as;
      Numbers n1 ;
      AddingANumber(Numbers n,int upperLimit, int arraySize) {
            n1 = n;
            ul = upperLimit;
            as = arraySize;
            new Thread(this).start();
      }


      public void run() {
            n1.addANumber(ul,as);
      }
}

class DeletingANumber implements Runnable {
      int ul;
      int as;
      Numbers n2;
      DeletingANumber(Numbers n,int upperLimit, int arraySize) {
            n2 = n;
            ul = upperLimit;
            as = arraySize;
            new Thread(this).start();
      }
```

```
      public void run() {
            n2.deleteANumber(ul,as);
      }

}



class AddAndDelete {

      public static void main(String args[]) {
      Numbers n = new Numbers();
      n.fillTheArray(20,10);
      n.sortAscending();
      AddingANumber  a1 = new AddingANumber(n, 20, 10);
      DeletingANumber d1 = new DeletingANumber(n, 20, 10);
      }
}
```

Practice 4.5

```
// Use of yield(), stop() and sleep() methods

class ClassA extends Thread
{
     public void run() {
      System.out.println("Start Thread A ....");
      for(int i = 1; i <= 5; i++) {
          if (i==1) yield();
          System.out.println("From Thread A: i = "+ i);
        }
      System.out.println("... Exit Thread A");
     }
}


class ClassB extends Thread
{
     public void run() {
      System.out.println("Start Thread B ....");
      for(int j = 1; j <= 5; j++) {
          System.out.println("From Thread B: j = "+ j);
          if (j==2) stop();
        }
      System.out.println("... Exit Thread B");
     }
}


class ClassC extends Thread
{
     public void run() {
      System.out.println("Start Thread C ....");
      for(int k = 1; k <= 5; k++) {
          System.out.println("From Thread B: j = "+ j);
```

```
          if (k==3) sleep(1000);
        }
      System.out.println("... Exit Thread C");
     }
}


class ThreadControl
{
    public static void main (String args[]) {
            TheadA t1 = new ThreadA();
      TheadB t2 = new ThreadB();
      TheadC t3 = new Thread3();
      t1.start(); t2.start(); t3.start();
      System.out.println("... End of executuion ");
    }
}
```

Practice 4.6

```
// Use of suspend() and resume() methods

class Thread1 extends Thread {
    public void run( ) {
      System.out.println (" First thread starts running" );
      sleep(10000);
      System.out.println (" First thread finishes running" );
    }
}

class Thread2 extends Thread {
    public void run( ) {
      System.out.println ( "Second thread starts running");
      System.out.println ( "Second thread is suspended itself ");
      suspend( );
      System.out.println (" Second  thread runs again" ));
    }
}

class AnotherThreadControl {
    public static void main (String, args[ ] ) {
      Thread1 fist = new Thread1( );            // It is a newborn thread i.e.
in Newborn state
      Thread2 second= new Thread2( );           // another new born thread
      first.start( );               // first is scheduled  for running
      second.start( );              // second is scheduled for running

      System.out.println("Revive the second thread" );  // If it is suspended
      second.resume( );
      System.out.println ("Second thread went for 10 seconds sleep " );
      Second.sleep (10000);
      System.out.println ("Wake up second thread and finishes running" );
      System.out.println ( " Demonstration is finished ");
    }
}
```

Practice 4.7

```
// Setting priority to threads

class ClassA extends Thread
{
     public void run() {
      System.out.println("Start Thread A ....");
      for(int i = 1; i <= 5; i++) {
          System.out.println("From Thread A: i = "+ i);
        }
      System.out.println("... Exit Thread A");
     }
}


class ClassB extends Thread
{
     public void run() {
      System.out.println("Start Thread B ....");
      for(int j = 1; j <= 5; j++) {
          System.out.println("From Thread B: j = "+ j);
        }
      System.out.println("... Exit Thread B");
     }
}


class ClassC extends Thread
{
     public void run() {
      System.out.println("Start Thread C ....");
      for(int k = 1; k <= 5; k++) {
          System.out.println("From Thread B: j = "+ j);
        }
      System.out.println("... Exit Thread C");
     }
}


class ThreadPriorityTest
{
    public static void main (String args[]) {
            TheadA t1 = new ThreadA();
      TheadB t2 = new ThreadB();
      TheadC t3 = new Thread3();

      t3.setPriority(Thread.MAX_PRIORITY);
      t2.setPriority(Thread.getPriority() + 1);
      t1.setPriority(Thread.MIN_PRIORITY);

      t1.start(); t2.start(); t3.start();
      System.out.println("... End of executuion ");
    }
}
```

Practice 4.8

The following Java application shows how the transactions in a bank can be carried out concurrently.

```
class Account {
      public int balance;
      public int accountNo;
      void displayBalance() {
            System.out.println("Account No:" + accountNo + "Balance: " +
balance);
      }


      synchronized void deposit(int amount) {
            balance = balance + amount;
            System.out.println( amount + " is deposited");
            displayBalance();
      }


      synchronized void withdraw(int amount) {
            balance = balance - amount;
            System.out.println( amount + " is withdrawn");
            displayBalance();
      }
}

class TransactionDeposit implements Runnable {
      int amount;
      Account accountX;
      TransactionDeposit(Account x, int amount) {
            accountX = x;
            this.amount = amount;
            new Thread(this).start();
      }


      public void run() {
            accountX.deposit(amount);
      }
}


class TransactionWithdraw implements Runnable {
      int amount;
      Account accountY;
      TransactionWithdraw(Account y, int amount) {
            accountY = y;
            this.amount = amount;
            new Thread(this).start();
      }
      public void run() {
            accountY.withdraw(amount);
      }

}
```

```
class Transaction {
      public static void main(String args[]) {
            Account ABC = new Account();
            ABC.balance = 1000;
            ABC.accountNo = 111;
            TransactionDeposit t1;
            TransactionWithdraw t2;
            t1 = new TransactionDeposit(ABC, 500);
            t2 = new TransactionWithdraw(ABC,900);
      }
}
```

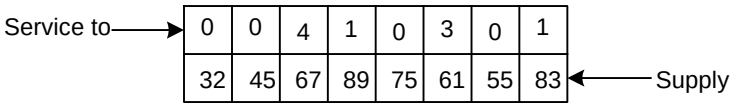
Assignment 4.3

Implement the single producer single consumer critical section problem.

Reference:
Illustration 13.6, Chapter 13, Page 132-133
Object Oriented Programming with C++ and Java
by D. Samanta, Prentice Hall of India

Assignment 4.4

Check whether the following Java application implements the following requirements or not. If not do the necessary.



A set of supplies is available. Four competitors C1, C2, C3 and C4 are there competing to acquire services. A competitor say C2, generate its request (randomly) and if its request match with any one of the supply and sypply is not yet granted to anybody else grant it to C2 by setting 2 in the Srevice to field of the corresponding supply.
The process will run concurrentl for all competitors and continue till the all supplies are exhausted.
Report the competitor who is the ultimate gainer.

```
class TaskRecord {
      TaskAllotmentInfo tai[] = new TaskAllotmentInfo[11];
      void enterInfo(int upperLimit, int arraySize) {
            int tn;
            int found = 0;
            if(upperLimit <=0) {
                  throw new IllegalArgumentException("UpperLimit must be
positive: " + upperLimit);
            }

            if (arraySize <=0) {
                        throw new IllegalArgumentException("Size of returned
List must be greater than 0.");
            }
```

```
            int i = 1;
            Random generator = new Random();
            tai[0] = new TaskAllotmentInfo();
            tai[0].taskNo = generator.nextInt(upperLimit);
            tai[0].taskAssigned = 0;


            while(i < arraySize) {
                  tn = generator.nextInt(upperLimit);
                  for(int j = 0; j <i; j++) {
                        if (tai[j].taskNo == tn) {
                              found = 1;
                              break;
                              }

                        else {
                              found = 0;
                        }
                  }
                  if (found == 0) {
                        tai[i] = new TaskAllotmentInfo();
                        tai[i].taskNo = tn;
                        tai[i].taskAssigned = 0;
                        i = i+1;
                  }
            }
      }

      public boolean searchLocal(int j, int t) {
      int k =0;
            while(k<j) {
                  if(tai[k].taskNo == t)
                        return false;
                  else k++;
            }

            return false;

      }

      public void display() {
            for(int i =0; i < 10; i++) {
                  System.out.println("Task Assigned to Client "
+tai[i].taskAssigned + "  Task Number: " + tai[i].taskNo);
                  }

      }
      public boolean search() {
            int i = 0;
            while(i<10){
                  if(tai[i].taskAssigned == 0 && (i<10))
                        return true;
                  else i++;
            }

            return false;
      }
```

```
	public boolean searchIfTaskAvailable(int n) {
		int i = 0;
		while(i<10){
			if(tai[i].taskNo == n && (i<10))
				return true;
			else i++;
		}

		return false;
	}


	public boolean taskNotAlloted(int j) {
		int i = 0;
		while(tai[i].taskNo != j) {
			i = i +1;
		}

		if(tai[i].taskAssigned == 0)
			return true;
		else return false;
	}



	public void alloted(int tn, int ta) {
		int i = 0;
		while(tai[i].taskNo != tn) {
			i = i +1;
		}

		tai[i].taskAssigned = ta;
		System.out.println("tai[i].taskNo: " + tai[i].taskNo +"
tai[i].taskAssigned : " + tai[i].taskAssigned);
	}
}


class Client implements Runnable{
	int jobNo;
	int cName;
	TaskRecord t= new TaskRecord();
	//Thread t2 = new Thread();
	Client(TaskRecord t1,int name){
		t = t1;
		cName = name;
		new Thread(this).start();
		//t2.start();
	}

	void getJobNo(int uLimit) {
		if(uLimit <=0) {
			throw new IllegalArgumentException("UpperLimit must be
positive: " + uLimit);
		}

		Random generator = new Random();
```

```
            jobNo = generator.nextInt(uLimit);

            }


      public void run(){
            while(t.search()) {
                  getJobNo(20);
                  if(t.searchIfTaskAvailable(jobNo)) {
                        if(t.taskNotAlloted(jobNo)) {
                              t.alloted(jobNo, cName);
                        }

                  }
            }

      }
}



class TaskAllotmentcurrent {
      public static void main(String args[]) {
      TaskRecord t = new TaskRecord();
      t.enterInfo(20,10);
      t.display();
      Client c1 = new Client(t,1);
      Client c2 = new Client(t,2);
      Client c3 = new Client(t,3);
      Client c4 = new Client(t,4);
      /*System.out.println("Thread one is alive; " +c1.t2.isAlive());
      System.out.println("Thread two is alive; " +c2.t2.isAlive());
      System.out.println("Thread three is alive; " +c3.t2.isAlive());
      System.out.println("Thread four is alive; " +c4.t2.isAlive());*/
      try {
            Thread.sleep(100);
            /*c1.t2.join();
            c2.t2.join();
            c3.t2.join();
            c4.t2.join();*/
      }

      catch (InterruptedException e) {
            System.out.println("Main Thread interrupted");
      }

      /*System.out.println("Thread one is alive; " +c1.t2.isAlive());
      System.out.println("Thread two is alive; " +c2.t2.isAlive());
      System.out.println("Thread three is alive; " +c3.t2.isAlive());
      System.out.println("Thread four is alive; " +c4.t2.isAlive());*/

      t.display();
      }

}
```

Assignment 4.5

Four event mangers are there (see Figure below) who manges certain activities as per the following policy.

Process Manager manintains *processMangerQ* (size: n)
Ready Manger manintains r*eadyMangerQ* (size: m << n)
Run Manger manintains *runMangerQ* (size: j)
Wait Manger manintains *waitMangerQ* (size: i < j)
finishQ (size: n)

1. Initially *Process Mangaer* is loaded (fully) with n number of programs to be serviced. Let each program is identified by a *ProgramID* and *ExecutionTime*.
2. *Process Manger* delivers a program to *Ready Manger* getting a signal from *Ready Manger* that the *redayMangerQ* is not full.
3. *Ready Manager* stores the program in the ascending order of program sizes in the *readyManagerQ*.
4. *Ready Manager* transfer a program to the *Rum Manger* on the following two cases:
   a. When a process from *runMnagerQ* moves to *waitMangerQ*.
   b. A process in *runMangerQ* moves to the *finishQ*
5. *Run Manager* processes each program in round-robin fashion with a time quantum *t*. *Run Manger* sends a program under process to *Wait Manger* when it completes *t* amount of time for it.
6. *Wait Manger* sends a process to *Ready Manger* after getting a signal from *Ready Manger*.
7. All managers run concurrently and they are communication each other stores a received program into their respective queues. When all queues are empty except the *finishQ* sytems signals a message “I D L E”

(Hint: Draw the process transition graph and identify the events for all the necessary transitiona. Implement each transition.)

****

Practice Sheet #5

****

Assignment 5.0

Study the following HTML page and browse it in your Desktop

```
<HTML>
     <!This is a simple HTML page
     >

     <HEAD>
            <TITLE>
          HTML Page Practice
      </TITLE>
     <HEAD>

     <BODY>
      <CENTER>
            Welcome to the World of Applet
      </CENTER>
      <BR>
      <RIGHT>
            <APPLET
                 CODE = HelloWorld.class
                 WIDTH = 400
                 HEIGHT = 200
            </APPLET>
      </RIGHT>
      </BODY>
</HTML>
```

How parameters say a String and an Integer can be passed to the applet HelloWorld class
in the HTML?

Hint:

Use the following syntax:

```
  public class HelloWorldWithParameters extend Applet
{
   String aString;
   int anInteger;

   public void init()
   {
      aString = getParameter("string");
        anInteger = Integer.parseInt(getParameter("intVal");

      if(aString == null)
            aString = "Hi";
      aString = "Hello" + aString;

   }
```

```
        public void paint(Graphics g)
        {
           String intToString;
           g.drawString(aString, 10, 75);
           intToString = String.valueOf(anInteger);
           g.drawString(intToString, 100, 75);
        }
  }
```

Now, folowing is the code to be embedded as Applet Tag in the HTML file

```
      <APPLET
          CODE = HelloWorld.class
          WIDTH = 400
          LENGTH = 200
            ALLIGN = RIGHT

          <PARAM NAME = "string" VALUE = "Java">
          <PARAM NAME = "intVal" VALUE = 96 >
      >
      </APPLET>
```

Practice 5.1

```
//  To create a button  //

import java.applet.Applet;
import java.awt.*;                        // To import the Button class

public class ButtonTest extends Applet {
    public void init( ) {
        Button    b1,b2;                  // Create two Button objects
        b1 = new Button ("Welcome");            // One button is initialized
        add(b1);                     // Add it into the applet
        b2 = new Button (  );                   // Second default button
        add(b2);                     // Add second button to the applet
    }
}
```

Assignment 5.1

How you can change the label of a button after it is adding to an applet?

Hint: Use the public void setLabel(String label) method defined in the Button class.

Practice 5.2

```
// To create a checkbox  //

import java.awt .*;                 // To import the check bon class.
import java.applet.*;

public class CheckboxText extends  Applet       {
    public void init ( ) {
        Checkbox c1 = new Checkbox (  );         // default constructor without
any label
        Checkbox c2 = new Checkbox ( " Solaris 2.x");  // Check box with default
layout
        Checkbox c3 = new Checkbox ( "Macintosh", null, false);
        Checkbox c4 = new Checkbox ( "Windows 95", null, true);
        add (c1);
        add (c2);
        add (c3);
        add (c4);
    }
}
```

Assignment 5.2

How a label can be set to a checkbox with an empty label?
How two different set of checboxes can be added into an applet?

Practice 5.3

```
//  To create Choice component   //

import java.awt.*;
import java.applet.*;

public class ChoiceDemo extends Applet {
    public void init ( ) {
      int width = Integer.parseInt (getParameter( " width" ));
      int height = Integer.parseInt (getParameter(" height"));
                              // To get the width and height of the applet.
      Choice os = new Choice ( );         //  os is an instance of choice class.
      Choice browser = new Choice ( );    // browser is the another instance.
      os.addItem ( " Windows 95" );
      os.addItem ( " Solaris 2.x " );
      os.addItem ( " Mac OS 7.5 ");
      browser.addItem (" Netscape Navigator 2.0");
      browser.addItem ( "Internet Explorer 4.0" );
      browser.addItem ("Hot Java ") ;
      add (os);
      add (browser );
      os.reshape ( 0, 0, width/2, height/2 );
      browser.reshape (0, height/2, width, height);
    }
}
```

Note: Here, you must pass the input to the applet through HTML file

Assignment 5.3

Observe the effect of passing different parameters in the reshape() method. Also try to change the name of the item or add more items after a choice already being added into the applet.

Practice 5.4

```
// To create a frame

import java.awt.*;
import java.awt.applet.*;

public class MyFrame {
    public static void main (String args[ ] ) {
      Frame frame =  new Frame ( " Frame in Java " );
      frame.resize (500, 500);           // component method to resize the frame
      frame.setBackground (Color.blue ); // Background of the frame will be blue
      frame.show ( )                    // Method to show the frame on the screen
    }
}
```

Note: This applet does not have init() methos as in other applet. This applet runs continuously and hence to quit the execution, press Control+C

Assignment 5.5

Observe the effect of the appearence of frame with different parameters to the methods resize() and setBackground()

See how title of a frame can be passed as command line argument.

Practice 5.5

```
// To create a Panel

import java.awt.* ;
import java.awt.applet.*;

public class MyPanel  {
    public static void main ( String args [ ] )  {
      Frame frame = new Frame( "Frame with panel");
      Panel panel = new Panel(  );
      frame.resize(200, 200);
      frame setBackground (Color.blue);
      frame.setLayout (null);             // override default layout
      panel.resize (100, 100) ;
      panel.setBackground (Color.yellow );
      frame.add (panel);                  // add the panel into the frame
      frame.show ( );                     // display the panel
    }
}
```

Practice 5.6

```
//    To create label   //

import.java.awt.*;
import.java.applet.*;

public class LabelDemo extends Applet  {
    public void init ( ) {
      setLayout (null );
      int width = Integer.parseInt ( getParameter ("width" ));
      int height = Integer.parseInt (get.Parameter (" height" ));
      Label left = new Label ( );          // default alighment is left )
      Label center = new label (  " Center", Label.CENTER); // For Centered alignment
      Label right = new Label("Right", Label.RIGHT);   // for Right justified label

      // Add the labels on the applet //
      add(left);
      add (right );
      add (center );

      // Place the label properly on the applet; otherwise deafault placing will occur //
      left.reshape ( 0, 0, width, height/3);
      right.reshape(0, height/3, width, height/3);
      center.reshape (0, 2 * hight/3 , width, height/3);
    }
}
```

Practice 5.6

```
// To create list //

import java.awt.*;
import java.applet.*;
public class ListDemo extends Applet  {
    public void init ( ) {
      setLayout (null );                  // It is a default Layout setting
      int width = Integer.parseInt (getParameter (" width" ) );
            int height = Integer.parseInt (getParameter (" height" ) );

      List os  =  new List (3, true);   // List object for list of Oss, multiple selection
      List browser  = new List (5, false);// List object for Browsers, single choice

      // Add the Items into the lists
        os.addItem ( " Windows 95" );
      os.addItem ("Solaris  2.x " );
      os.addItem (" Mac OS 7.5 " );
      browser.addItem ("Netscape Navigator" );
      browser.addItem ("Internet Explorer" );
      browser.addItem ("Hot Java" );
      browser.addItem ("Mosaic" );
            browser.addItem ("Applet Viewer" );

      browser.select(1) ;     // Second item in the Browser list is selected
```

```
      // Add and place the lists onto the applet
      add(os);
      add(browser);
      os.reshape (0, 0, width,   height /2 );
      browser. reshape ( 0, height /2, width, height /2 );
    }
}
```

Assignment 5.6

```
It can be noted that, the List class has two constructors namely :
            public List(  ); and
            public List ( int numberOfIemToDisplay, boolean multiSelectionChoice
);
```

Apply both constructors.

Practice 5.7

```
// To create scroll bar  //


import java.awt.*;
import java.applet.*;

public class ScrollDemo extends Applet {
    public void init ( ) {
      setLayout( new Borderlayout ( ) );        // default border layout setting
      int width = Integer.parseInt (getParameter (" width" ));
      int height = Integer.parseInt (getParameter (" height"));
      Scrollbar hScroll = new Scrollbar ( Scrollbar. HORIZONTAL, 50, width /10,
0, 100 );
      Scrollbar vScroll = new Scrollbar ( Scrollbar.VERTICAL, 50, hight /2, 0,
100);
      add (hScroll );
      add (vScroll );
    }
}
```

Practice 5.8

```
// To create a text field

import java.awt.*;
import java.applet.* ;
public class TextFieldDemo extends Applet       {
   public void init ( ) {
      add ( new TextField("Type your Name", 20);
      // create a text field window of 20 character widths

      Label login = new Label ( "Login : " , Label.LEFT );
      Label pswd = new Label ( "Password : " , Label.CENTER );
```

```
      TextField log  = new TextField (8) ;      // Text field for 8 characters
      TextField  pas = new TextField (8);
      pas.setEchoCharacter ( '*' );             // echo hide the type pass

      add (login);  add (log);           // add labels in the applet and
      add (pswd );  add (pass );         //  add corresponding text fields
   }
}
```

Practice 5.9

```
// To create text area  //

import java.awt.*;
import java.applet.*;
public class TextAreaDemo extends Applet  {
   TextArea text;
   String multiLineText =
      " To learn Java, you will first"
            + " need to obtain \ n   two different pieces of  "
            + "softwares : \ n "
            + " The first is the JDK (Java Development Kit"
            + "and \n the second piece of software "
            + "is a Java capable browser."
            + " \n JDK software is needed for Writing, "
            + " Compiling, and Testing Java applet \n and
            + " Applications.\n \n" ;

    public void init (  ) {
        setLayout(null);
      text = new TextArea (multiLineText, 20, 40);
      add (text);
    }
}
```

Practice 5.10

```
// To create Flow layout

import java.awt.*;
import java.applet.*;
import java.util.*;
public class FlowLayoutDemo extends Applet {
    public void init ( )      {
      setLayout (new FlowLayout (FlowLayout.RIGHT, 10, 3));
      String val =  " Data is not Information" +
                        " is not knowledge is not wisdom";
      StringTokenizer str = new StringTokenizer (val ); // Read a token from the String val
      while (str. hasMoreTokens ( ) ) {
          add (new Button (str.nextToken( ) ) );   // For one token add a label into the panel
      }
    }
}
```

Assignment 5.7

Create two layout manager by breaking the string into two parts and with different allignment

Practice 5.11

```
// To Create Border Layout Manager

import java.awt.*;
import java.applet.*;
import java.util.*;

public calss BorderLayoutDemo extends Applet {
    public void intit ( ) {
      Frame frame = new Frame ( );
      frame.setLayout ( new BorderLayout (20,40));
      Button  buttonN, buttonS, buttonE, buttonW,  buttonC;

      buttonN = new Button ( " NORTH " );
      buttonS  = new Button ( " SOUTH" );
      buttonE = new Button ( " EAST" );
      buttonW = new Button (" WEST" );
      buttonC = new Button("CENTER");
      frame.resize (250,250);        // resize the frame is height = 250 pixels , width 250 pixels
      frame.add ( "North", buttonN );
      frame.add (" South ", buttons );
      frame.add ( "East", buttonE );
      frame.add (" West" , buttonW );
      frame.add("Center",buttonC);
      frame.show ( );
     }
}
```

Practice 5.12

```
// Grid Llayout Manager //

import java.awt.*;
import java.applet.Applet;

public class GridLayoutDemo extends Applet {
    Button b1,b2,b3,b4,b5,b6 ;
    public void init ( ) {
      setLayout (new GridLayout (3,2) ) ;       // 3 rows, 2 columns
      b1 = new Button ("B1");
      b2 = new Button ("B2");
      b3 = new Button ("B3" );
      b4 = new Button ("B4" );
      b5 = new Button ("B5" );
      b6= new Button ("B6" );
      add (b1); add (b2); add (b3);
      add (b4); add (b5); add (b6);
    }
}
```

Practice 5.13

```
// Craet Card Layout

import java.awt.*;

public class Crads extends java.applet.aApplet {
    CardLayout   layout;
    public void init ( ) {
      layout = new CardLayout ( ) ;
      setLayout (layout );
      add ("1", new Button ("Card1" ));
      add ("2", new Button ("Card2" ));
      add ("3", new Button ("Card3" ));
      add ("4", new Button ("Card4" ));
      add ("5", new Button ("Card5" ));
    }

    public boolean key Down (Even e int key ){
      layout next (this);
      return (true );
    }
}
```

Practice 5.14

```
// Event handling

import java.awt.*;
import java.applet.*;
public class EventDemo extends Applet {
    static final int n = 4;
    Label label;
    public void init(  ) {    // To initialize the applet for 16-cells grid
      setLayout (new GridLayout (n,n);          // To for 4x4 cells grid
      setFont (new Font ("Helvetica",  Font.BOLD, 24 ));     // Font of Helvetica of size 24
      for (int i =0; i < n; i++ )   {
         for (int j = 0; j < n; j++ ) {
            int k = i * n + j;
            if (k > 0)
               add (new Button ("  " + k);    // Place a button at (i,j)-th cell
         }
      }

      label = new Label (" * " , Label.CENTER );
      label.setFont (new FONT (" Times Roman", Font.ITALIC, 24 ));
      add (label );
    }

    public boolean action(Event e, Object obj) {              // Overridden event handler
      if (e.target instanceof  Button ) ) {                   //  Wait for clicking a button object
         label.setText ((String ) obj );                      // Print the value of the Clicked butoon
      }
      return false ;
    }
}
```

Practice 5.15

```
// Graphics Test   //

import java.awt.*;
import java.applet.*;
public class GraphicsText extends Applet  {
    public void init (  ) {
      public void paint (Graphics g) {
      Dimension d = size ( ) ;                    // returns the width and height  of the applet
      g.setColor (Color.black );                  // set the drawing color as black
      g.drawRect (0, 0, d.width-1, d.height -1);
      g.setColor (Color.blue);
      g drawRect (10, 10, d.width-10, d.height-10 );     // draw another inner blue Rectangle
      g.setColor (Color.red );
      g.drawOval (15,15, 50,50 );
      g.setColor( Color.yellow );
      g.fillOval ( 20, 20,40, 40 );
      g.setColor (Color.red );
      g.drawArc (230, 10, 50, 50, -90, 270 );
      g.fillArc (190, 10, 50, 50, -90, 270 );
      g.setColor (Color.pink );
      g.drawRoundRect (130, 70, 50, 50, 10, 10 );
      g.fillRoundRect (190, 70, 50, 50, 10, 10 );
      int [ ] xPoints  =      { 10,70,120,120,70,10,10 };
      int [ ] yPoints  =      { 130, 150, 130, 180, 170, 180, 130 };
      Color myColor = new Color (255,100,150);        // set a RGB color
      // A color can be choosed, R, G, B  integers ranges between  0 to 255
      g.setColor (myColor );
      g.drawPolygon (xPoints , yPoints, 7);           // To draw a polygon with 7 points
      g.setColor (Color.gray );
      g.draw3DRect (10, 190, 50, 50, true );
      g fill3DRect (70, 199, 50, 50, true);
      g.setColor (Color.green );
      g.draw3DRect (130, 190, 50, 50, false );
      g.fill3DRect (190, 190, 50, 50, false );
    }
   }
}
```

****

Practice Sheet #6

****

Practice 6.1

Observe the differences in the following two pieces of codes.

// Program with no Exception handler //

```
class DivideZero {
 static int anyFunction ( int x, int y ) {
    int a = x/y;
    return (a);
 }

 public  static void main (String args [  ] ) {
    int result = anyFunction (25, 0) ;          // Exception occurs here as y = 0
    System.out.println ( " Result : " + result );
 }
}
```

// Folowing is the program dealing with the exception

```
class DivideZero {
  static int anyFunction (int x, int y ){
    try {
       int a = x/y;
       return(a);
    }
    catch (ArithmeticException e) {
      System.out.println ( " Division by zero" );
    }
    }

 public static void main (String args[ ] {
  int a,b, result;
  System.out.print("Enter any two integers : ");
  a = System.in.read( );
  b = System.in.read( );
  result  = any Function (a, b);
  System.out.println ( "Result : " + result);
 }
}
```

Assignment 6.1

Through commnad line pass few argumnets to the program. All argumnets other than integer will be treated as invalid. Count the number of valid and invalid arguments and print them using try-catch block

Practice 6.2

```
// Using multiple try-catch

class MultiCatch  {
   public static void main (String args[ ]) {
     try {
      int i = args.length;
      String myString = new String[ i ];
      // If i = 0 then  myString null pointer error
// #1 //
      if(myString[0].equals("Java"));
         System.out.println("First word is Java !");
       System.out.println( " Number of arguments = " + i );
// # 2 //
      int x = 18/ i;
      int y[ ] = {555, 999};
        // y is an array of size 2 and index are 0,1
// #3 //
      y[ i ] = x;
      // Index is out-of-range may occur if i > 1
      }
      catch (ArithmeticException  e ) {         // To catch the error at #2
      System.out.println ( " Div by 0 : "+ e );
      }
      catch (NullPointerException  e ) {        // To catch the error at #1
      System.out.println ( " A null pointer exception : " + e );
      }
      catch (ArrayIndexOutofBoundsException e ) { // To catch the error at  #3
      System.out.println ("Array Index OoB : " + e);
      }
   }
}
```

Practice 6.3

```
// Multiple Errors with one catch

class ExceptionTest {
    public int j;
    static void main (String   args[ ] ) {
            for (int i = 0; i < 4; i++ ) {
      try {
        switch (i) {
           case 0 :
            int zero = 0;
            j = 999/ zero;          // divide by zero
            break;
           case 1:
            int b[ ] = null;
            j = b[ 0] ;             // Null pointer error
            break;
           case 2 :
            int c = new int [2] ;
            j = c[10];        // Array index is out-of-bound
```

```
            break;
           case 3 :
              char ch = "Java".charAt(9) ;      // String index is out-of-bound
            break;
      }
      }
      catch (Exception e) {         // To catch an exception
      System.out.print( " In Test case # " + i + "\n " );
      System.out.println (e);
      }
   }
}
```

Practice 6.4

```
// Use of finally in try-catch //

class FinallyDemo {
   public static void main (String [ ] args ) {
      int i = 0;
      String greetings [ ] = {
          "Hello Twinkle !",
            "Hello Java !",
          "Hello World ! "
        };

      while ( i < 4) {
        try {
          System.out.println (greetings [i] );
        } catch (Exception e ) {
            System.out.println (e.toString );       // message of exception e in String format
            System.out.println("Quiting the program execution ....");
          }
        finally {
            System.out.println (" Hi !");
               i++;
          }
      }
    }
}
```

Assignment 6.2

Modify the above code without using any catch block but with similar effect of execution.

Practice 6.5

```
// Use of throw construct

class UserException extends Exception
{
    UserException(String message)
    {
       super(message);
    }
}

class ThrowExceptionDemo
{
    public static void main(String args[])
    {
       int a = 12; b = 246;
       a = System.in.read();
       b = System.in.read();

        try {
           float x = (float) a/(float) b;
           if (x < 0.05)
             throw new UserException("Invalid divisor");
        }
        catch (UserException e)
        {
          System.out.println("Cuaght the Exception....");
        System.out.println(e.getMessage() );

        }
        finally {
          System.out.println("The ulimate bock is executed");
        }
    }
}
```

Practice 6.6

```
//   To test the methods in the class java.io.File   //

import java.io.File

class FileTest  {
   public static void main (String args [ ] ) throws IOException {
      File  fileToCheck;
      if (args.length > 0 ) {
         for (int i = 0; i < args.length; i++ ) {
            fileToCheck = new File(args[ i ]);
            getNames (fileToChecks );
            getInfo(fileToCheck);
         }
      }
      else
          System.out.println (" Usage : Java file test <filename (s) >);
    }
```

```
    public static void getNames (File f ) throws IOException {
      System.out.println ("Name : " + f. getName( ) );
      System.out.println ("Path : " + f. getPath ( ) );
      System.out.println ("Parent : " + f.getParent ( ) );
    }

    public static void getInfo (File f ) throws IOException  {
      if (f.exists ) {
          System.out.print ("File exists ");
          System.out.println (f.canRead( ) ? "and is readable" : "");
          System.out.println ( f.canWrite( ) ? "and is writable" : "");
          System.out.println ("File is last modified : " + f.lastModified( ));
          System.out.println ("File is " + f.length( ) + "bytes" );
      }
      else
          System.err.println (" File does not exist." );
    }
}
```

Practice 6.7

```
//   Input Stream : Displaying a file   //

import java.io.*;
import java.util.*;

class InputStreamTest   {
   public static void main (String args [ ] ) {
      int size;

      // To open a file input stream.
      FileInputStream fin;
       fin = new FileInputStream (" C: \WINDOWS\SYSTEM\SYSTEM.INI");
       size = fin.available( );                    // returns the number of bytes available
      System.out.println("Total bytes ::" + size);

      System.out.println ( " First ¼ is displayed : Using read( )");
      for (int i = 0; i < size /4 ; i++ )       {
               System.out.println ((char) fin.read( ) );
        }

        System.out.println (" Remaining bytes :" + fin.available( ) );
        System.out.println ("Next ¼ is displayed : Using read( b[ ] )");
        byte b[] = new byte[size/4];
        if (fin.read (b) != b.length )
          System.err.println ("File reading error : ");
        else {
            String temp = new String (b, 0, 0, b.length );
            System.out.println (temp) ;              // display text string

            System.out.println ( " Still available : " + fin.available( ) );
            System.out.println ( " skipping ¼ : Using skip ( )" );
            fin.skip(size/4);
            System.out.println ("File remaining for read:" + fin.available( ) )
        }
```

```
      fin.close ( );                // Close the input stream
   }
}
```

Note: This program assumes file SYSTEM.INI which is in C:\WINDOWS\SYSTEM directory.

Practice 6.8

```
// Read from keyboard and copy them into a file //

class FileInOut   {
   public static void main (String args [])     {
      byte buffer = new byte [512] ;                    // A temp buffer to store text
      int i = 0, size = 0;
      String inFile = null;
      String outFile = null;
      //FileInputStream = fin;
      FileOutputStream= fout1, fout2;
      try {
         outFile = args[0];
         int  c = 0;
         while (( c = System.in.read ( ) )!=-1 )  {        // Read from keyboard till ctrl+Z
            buffer[i] = (byte) c;
            i++;
         }
         size= i;                                       // Number of bytes read
           fout = new FileOutputStream (args[o]);
         fout1.read (buffer);                           // Write whole buffer into the file
         fout1.close ( );
      } catch (ArrayindexOutofBoundException e) {

    }
}
```

Assignment 6.3

Write a program to read a file and copy the file into another file.

Practice 6.9

```
// Using the class RandomAccessFile

import java.io.*;
class RandomAccessTest
{
    public static void main(String args[])
    {
        RandomAccessFile file;
        try {
          file = new RandomAccessFile("aFile.txt", "rw");
            file.writeChar('A');
```

```
          file.writeInt(123);
          file.writeDouble(1.2345);

          file.seek(0);
            System.out.println(file.readChar());
            System.out.println(file.readInt());
          System.out.println(file.readDouble());

          file.seek(2);
          System.out.println(file.readInt());

          file.seek(file.length());
          file.writeBoolean(false);

          file.seek(file.length());
          System.out.println(file.readBoolean());
      } catch (IOException e) {System.out.println(e);}
        finally {
          file.close();
        }
    }
}
```

Assignment 6.4

Using the class RandomAccessFile repeat the Assignment 6.3

Project 1

Topic: Graphical User Interface based Interest Calculator. Input the Principal, Rate of Interest and Year through applet. Calculate the Simple Interest and Compound Interest based on the choice clicked by the user.

```
import java.awt.*;
import java.applet.*;
import java.lang.Math;
//import Mypackage.*;

public class TextFieldDemo extends Applet{
      TextField prin,inter,year,stotal,ctotal;
      Label Prin,Inter,Year,Stotal,Ctotal;
      float p,i,y,tcI,tsI;
      Button sI,cI;

      public void init(){
            prin=new TextField("0.0",10);
            inter=new TextField("0.0",10);
            year=new TextField("1.0",10);
            stotal=new TextField("0.0",10);
            ctotal=new TextField("0.0",10);
            stotal.setEditable(false); // Make output field not editable.
            ctotal.setEditable(false);

             Prin=new Label("Principal");
             Inter=new Label("Interest");
             Year=new Label("Year");
           Stotal=new Label("SI Total",Label.CENTER);
           Ctotal=new Label("CI Total",Label.CENTER);

            sI=new Button("Simple Interest");
            cI=new Button("Compound Interest");
            //TextField year=new TextField("1.0",10);
            add(Prin);        add(prin);
            add(Year);        add(year);
            add(Inter);       add(inter);
            add(Stotal);      add(stotal);
            add(Ctotal);      add(ctotal);
            add(sI);
            add(cI);
      }

      public void calculate(){
            String temp;
            //Graphics g;
      //    Myclass myclass=new Myclass();

            temp=prin.getText();
            p=Float.valueOf(temp).floatValue();

            temp=year.getText();
            y=Float.valueOf(temp).floatValue();

            temp=inter.getText();
            i=Float.valueOf(temp).floatValue();
```

```
            //tsI=myclass.getsI(p,i,y);
            tsI=p+p*i*y/100;
            //tcI=myclass.getcI(p,i,y);
            tcI=p*(float) (Math.pow( (1+i/100),y) );
      }

      public void paint(Graphics g){

            int i[]=new int[2];
            int j=1;
            Graphics g2;
            i[0]=(int)(360*p/tsI);
            i[1]=(int)(360*p/tcI);

            System.out.println("i =\t"+i);

            //catch(NumberFormatException nfe)


            g.setColor(Color.blue);
            g.fillArc(100,100,100,100,180,i[0]);
            g.setColor(Color.red);
            g.fillArc(100,100,100,100,180,-360+i[0]);

            g.setColor(Color.blue);
            g.fillArc(200,100,100,100,180,i[1]);
            g.setColor(Color.red);
            g.fillArc(200,100,100,100,180,-360+i[1]);
      }

public boolean action (Event evt, Object arg) {

    String temp;
              // Button hit
     if (evt.target instanceof Button) {

               if (evt.target == sI) {
                 calculate();
                 temp=String.valueOf(tsI);
                    stotal.setText(temp);
               }

               else  if (evt.target == cI) {
                 calculate();
                 temp=String.valueOf(tcI);
                    ctotal.setText(temp);
               }
             }
             return(true);
            }

   }
```

Project 2
Java Calculator: Electronic hand held calculator using Java Applet

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/*
<applet code="CalculatorApplet.class"WIDTH=300 HEIGHT=300>
</applet>
*/

public class CalculatorApplet extends JApplet
{
    public void init()
    {
      Container contentPane = getContentPane();
      CalculatorPanel panel = new CalculatorPanel();
      contentPane.add(panel);
    }
}

class CalculatorPanel extends JPanel
{
    public CalculatorPanel()
    {
      setLayout(new BorderLayout());

      result = 0;
      lastCommand = "=";
      start = true;

      display = new JTextField("0.0");
      display.setEditable(false);
      add(display,BorderLayout.NORTH);

      ActionListener insert = new InsertAction();
      ActionListener command = new CommandAction();

      panel = new JPanel();
      panel.setLayout(new GridLayout(8,4));

      for(int i=0;i<10;i++)
          addButton(""+i,insert);

      addButton(".",insert);
      addButton("+",command);
      addButton("_",command);
        addButton("x",command);
      addButton("/",command);
      addButton("=",command);
      addButton("+/-",command);
      addButton("Sin",command);
      addButton("Cos",command);
      addButton("Tan",command);
      addButton("x^y",command);
      addButton("Sqrt",command);
```

```
      addButton("log",command);
      addButton("ln",command);
      addButton("x^(1/y)",command);
      addButton("n!",command);
      addButton("Pi",insert);
      addButton("e",insert);
      addButton("A/C",command);
      addButton("aSin",command);
      addButton("aCos",command);
      addButton("aTan",command);

      add(panel,BorderLayout.CENTER);
    }

    private void addButton(String label,ActionListener listener)
    {
      JButton button = new JButton(label);
      button.addActionListener(listener);
      panel.add(button);
    }

private class InsertAction implements ActionListener
{
    public void actionPerformed(ActionEvent event)
    {
      String input = event.getActionCommand();
        if(start)
          {
            display.setText("");
            start = false;
          }
      if(input.equals("e"))
          {
          if(display.getText().equals("-"))
            display.setText("-"+Math.E);
            else
              display.setText(""+Math.E);
          start=true;
          }
      else if(input.equals("Pi"))
          {
            if(display.getText().equals("-"))
            display.setText("-"+Math.PI);
            else
              display.setText(""+Math.PI);
          start=true;
          }
      else
          display.setText(display.getText()+input);
    }
}

private class CommandAction implements ActionListener
{
    public void actionPerformed(ActionEvent event)
    {
      String command = event.getActionCommand();
```

```
      String co = command;
      if(co.equals("A/C"))
          {
            result = 0;
            display.setText(""+result);
            lastCommand = "=";
            start=true;
          }
      else if((co.equals("+/-"))||(co.equals("Sin"))
               ||(co.equals("Cos"))||(co.equals("Tan"))||(co.equals("Sqrt"))
         ||(co.equals("n!"))||(co.equals("ln"))||(co.equals("aSin"))
             ||(co.equals("aCos"))||(co.equals("aTan"))||(co.equals("log")))
                {
                lastCommand=command;
           calculate(Double.parseDouble(display.getText()));
           lastCommand="=";
           start=true;
                }
           else if(start&&(lastCommand.equals("="))&&(result==0)
                                        &&(display.getText().equals("0.0")))
          {
            if(command.equals("_"))
                {
                  display.setText("-");
                  start = false;
                }
            else
                lastCommand = command;
          }
      else
            {
            calculate(Double.parseDouble(display.getText()));
               lastCommand=command;
                start= true;
          }
    }
}

public void calculate(double x)
{
    if(lastCommand.equals("+")) result += x;
    else if(lastCommand.equals("_")) result -= x;
    else if(lastCommand.equals("x")) result *= x;
    else if(lastCommand.equals("/")) result /= x;
    else if(lastCommand.equals("+/-")) result = -x;
    else if(lastCommand.equals("Sin")) result = Math.sin(x*Math.PI/180.0);
    else if(lastCommand.equals("Cos")) result = Math.cos(x*Math.PI/180.0);
    else if(lastCommand.equals("Tan")) result = Math.tan(x*Math.PI/180.0);
    else if(lastCommand.equals("=")) result = x;
    else if(lastCommand.equals("x^y")) result = Math.pow(result,x);
    else if(lastCommand.equals("Sqrt")) result = Math.sqrt(x);
    else if(lastCommand.equals("n!")) result = fact((int)x);
    else if(lastCommand.equals("ln")) result = Math.log(x);
    else if(lastCommand.equals("aSin")) result = Math.asin(x)*180/Math.PI;
    else if(lastCommand.equals("aCos")) result = Math.acos(x)*180/Math.PI;
    else if(lastCommand.equals("aTan")) result = Math.atan(x)*180/Math.PI;
    else if(lastCommand.equals("log")) result = Math.log(x)/Math.log(10.0);
```

```
    else if(lastCommand.equals("x^(1/y)")) result = Math.pow(result,1.0/x);

    display.setText(""+result);
}

 private int fact(int n)
      {
          if((n==0)||(n==1)) return 1;
          else return n*fact(n-1);
      }

private JTextField display;
private JPanel panel;
private double result;
private String lastCommand;
private boolean start;
}
```

****
Practice Sheet #7
****

Network and Socket Programming in Java

Networking in Java

Java makes the networking easy and at the same time powerful. It provides `java.net` package containing number of classes to support networking. Java supports Internet' s TCP/ IP Protocol. Let us give a quick look about what do we mean Internet's TCP/IP. TCP/IP stands for **T**ransmission **C**ontrol **P**rotocol / **I**nternet **P**rotocol - the two data communication protocols. TCP presents a connection-oriented service (like the telephone network. Here, we have to establish a connection between two communicating processes and as soon as the connection is established, data can be transferred. This protocol thus allows to send or receive arbitrary amount of data without any boundaries. TCP guarantees delivery of data.

On the other hand, IP is a connection-less transport service. It is a `datagram` protocol which means that data should be sent and received in individual packets. Each IP packet travels its own, like an individual letter in a postal network; here thus, delivery is not guaranteed, packets may be duplicated , lost, or arrive in a different order to that in which they were sent.

The other subtle difference between TCP and IP is that IP allows you to send a packet to an individual machine on the Internet i.e. only single destination address is allowed; whereas, using TCP, one can add more than one destination address.

Although, IP is not reliable but it is more efficient than TCP. There is one more TCP/IP protocol which builds on IP to achieve its functionality : UDP (**U**ser **D**atagram **P**rotocol). This came from the fact that - IP is theoretically limited to sending a 64k byte packet, which would be insufficient for sending many files or even many large GIF images embedded in web pages. One way to solve this problem by breaking up the user's data stream into separate IP packets, numbering them and then reassembling them on arrival. UDP is thus like a cross reference between IP and TCP- it is a data gram protocol with the same 64k packet size limit of IP, but also allows port addresses to be specified.

Java supports both the TCP and UDP protocol families. First, we shall discuss the internet addressing method in Java and the applications of two protocols shall be discussed then.

Internet addressing method in Java

Java looks the whole Internet as a collection of host machines - each host is identified by a number called port number. But Internet names are in the form of user readable strings like "ds@iitkgp.ac.in" instead of raw numbers. Java handle this situation nicely. In `java.net` package there is `InetAddress` class that allows you to specify an address in a high level fashion "host.subdomain.domain"; this class then converts this textual name into 32 bits binary string form (for e.g. 259.161.69.192, (expressed in decimal)).

The `InetAddress` class has no visible constructors. In order to create an `InetAddress` object, you have to use one of the available factory methods. Factory methods are simply static methods that return an instance of the class they reside in. In this case, `InetAddress` has three such methods - whose structures are stated below:

```
public static InetAddress getLocalHost ( );
public static InetAddress getByName (String host );
public static InetAddress [ ] getAllByName (String host );
```

The `getLocalHost()` method simply returns the `InetAddress` object that represents the local host (i.e of your own machine). The `getByName()` method returns an `InetAddress` object for a host name passed in. On the Internet, it is quite possible for a single name to be used to represent several machines. The `getAllByName()` factory method returns an array of `InetAddress`es that a particular name resolve to. These methods throw `UnknownHostException` in case of an error.

The `InetAddress` class also has a few non-static methods, which can be used on the objects returned by the methods are mentioned below :

public string getHostName( ) - returns a string that represents the host name associated with the `InetAddress` object.

public byte[ ] getAddress ( ) - returns a four elements byte array that represents the object `InetAddress` in " Network byte order ".

public String toString ( ) - returns a string that lists the host name.

Communication using UDP

Java implements the low level communication with TCP/ IP UDP protocol using two classes : `DatagramPacket` and `DatagramSocket` . The `DatagramPacket` object is the data container, while the `DatagramSocket` is the mechanism used to send or receive the `DatagramPacket`s.

`DatagramPackets` can be created using one of the two constructors as stated below :

public DatagramPacket (byte ibuf [ ], int length);
public DatagramPacket (byte ibuf [ ], int length, InetAddress iaddr, int iport);

The first constructor uses only a `byte` buffer and a length. It is used for receiving data over a `DatagramSocket`. The second constructor adds target address and port number, which are used by `DatagramSocket` to determine where the data in the packet will be sent.

There are several methods in this class `DatagramPacket` for accessing the internal state. They can be called to get the destination address and port number of a packet, as well as the raw data and its length. Following is a summary of each of them.

public InetAddress getAddress ( ) - returns the destination InetAddress. It is typically used for sending.

public int getport ( ) - returns the integer destination port number. It is typically used for sending.

public byte [ ] getData ( ) - returns the byte array of data contained in the datagram. It is used to retrieve data from the datagram after it has been received.

public int getLength ( ) - returns the length of the valid data contained in the byte array that would be returned from the `getData()` method.

As one can note that the there is no method in `DatagramPacket` class to send or receive any data grams; this functionality is the responsibility of a companion class DatagramSocket. It has following two constructors:

public DatagramSocket ( );
public DatagramSocket (int port);

The first constructors is used to make communication between two ports in the local machine whereas the second constructor is useful to do communication between two non-localized ports. This class has the following methods :

public void send (DatagramPacket pkt) - takes a `DatagramPacket` object and sends the datagram's data to the previously defined host and port address.

public synchronized void received (DatagramPacket pkt) - takes a DatagramPacket object as a recipient for data gram to be received.

public synchronized void close( ) - to close the DatagramSocket that established when the communication is completed.

Following is an example to exercise the Data , Data gram Packet and Data gram socket classes.

Example 1

```
// Communication using UDP //

import java.net.*;         // Include the java.net package
class CommnUDP {
       public static int server port = 666;           // A port 666 is decided as sever
port
       public static int client port = 999;           // Another port 999 is decided as
client port .
       public static int size = 1024 ;         // For the buffer limit
       public static DatagramSocket ds;        // Socket instance for communication
       public static byte buffer [ ] = new byte [size] ;    //  buffer  is  to  store   a
content
       public static void theServer( ) {
              int i =0;
              while (true)                     // run till the end of session
                     int c= System.in.read ( );              // read from keyboard
              switch ( c) {
                     case -1 :
                            System.out.println (" Server quits : ");
                            return;

                     case '\n'  :        // Message entered for transmission
                            ds.send       (new       DatagramPacket       (buffer,       i,
                            InetAddress.getLocalHost( ),

                            clientPort ));
                            i =  0;
                            break;
                     default :
                            buffer [i++ ] = byte( c );
                     }
              }
       }
       public static void theClient ( ) {
              while(true) {
                     DatagramPacket pkt = new DatagramPacket (buffer, buffer.length);
                     ds.receive (pkt) ;
                     System.out.println   (new   String   (pkt.getData   (   ),   0,   0,
pkt.getLength( ))));
              }
       }
       public static void main (String args [ ] )    {
              if (args.length == 1) {
                     ds = new DatagramSocket (serverPort );
                     theServer (  );                  // sending message from server
              } else {
                     ds = new DatagramSocket (clientPort );
```

```
                    theClient ( );            // Receiving message in client
                }
        }
}
```

This example illustrates the DatagramSocket constructor to perform the communication between two ports on the local machine. To use this program, run Java commnUDP in one window; this will be the client. Then run Java commnUDP abc; this will be the server. Anything that is typed in the server window will be sent to the client window after a new line is entered.

Next Example 2 is to use UDP protocol to make a dialog between two distant machines in the Internet.

Example 2

```
// Communication using UDP outside  //

import java.net.*;
class DialogUDP {
        public static int hostPort = 999;             // Port No. of the distant machine
is assumed
        public static void main (String args [ ] )
                DatagramSocket      ds;
                DatagramPacket      pkt;
                InternetAddress hostAddress;
                byte buffer [  ] = new byte [ 1024];
                ds = new DatagramSocket ( );            //  Establish  a  socket  in   your
        machine
                hostAddress = InetAddress getByName (“ www.iitkgp.com” );
                                        // If you are ready to send a message for me.
        //   Phase  I :  To transmit a message
                // Code for loading some message into buffer array   //

                pkt = new DatagramPacket (buffer, buffer.length, hostAddress, hostPort);
                                        // Data gram packet for sending  message
                ds.send ( pkt) ;                //  message  is  transmitted  for  the  host
machine.

        // Phase II : To receive some message from host.
                pkt = new DatagramPacket (buffer, buffer.length );
                ds. receive (pkt);              // data gram  as a receiving packet
                String message = new String (pkt.getData ( ), 0);
                System.out.printin (“ Message : ”  + message );
                ds.close ( );
        }
}
```

This program is a two phased dialogue. In the first phase a message from the user’s machine will be delivered to the host machine “www. iitkgp.com”, and if the host send some message that will be received in the second please.

Communication Using TCP

TCP communication based on the concept of Client/Server model. It allows bi-directional point-to-point, stream based connection between two machines one of which is termed as sever and other being the client. This is implemented using two core classes : Socket and ServerSocket. These two classes are to establish socket connections between client and server. Sockets in Java are end points of communication links between processes (which are under run in two machines). The Socket class is to create a socket for client and the ServerSocket is for server. Let us visit these two classes before proceeding further :

Socket class is having two constructor as below :

public Socket (String host , int port) - creates a socket connecting the local host to the named host and port.
public Socket (InetAddress address, int port) - creates a socket using pre-exiting InetAddress object and a port.

The Socket class also contains methods to examine addresses and port information associated with it at any time.

public InetAddress getInetAddress( ) - returns the InetAddress associated with the socket object .
public int get.port ( ) - returns the remote port with which socket object is connected to.
public int getLocalPort () - returns the local port of the socket object is connected to.

Once the Socket object is created, it then can gain access to the input and output streams associated with it. Each of these next methods in Socket class are to do this :

public InputStream getInputStream( ) - returns the InputStream associate with this socket.
public void close ( ) - close both the InputStream and OutputStream.

Similar to the Socket class, ServerSocket class is to create a socket connection for server. Two constructors are available here as :

public ServerSocket (int port) - creates a ServerSocket on the specified port.
public ServerSocket (int port, int count) - creates a server socket on the specified port waiting "count" milliseconds if the port is in use.

It has the methods getInetAddress(), getLocalPort() and close() very similar to the method in Socket class. Another extra method namely public Socket accept() is very important which initiates communication for waiting clients and then returns with a normal socket.

The communication process is shown in Figure.

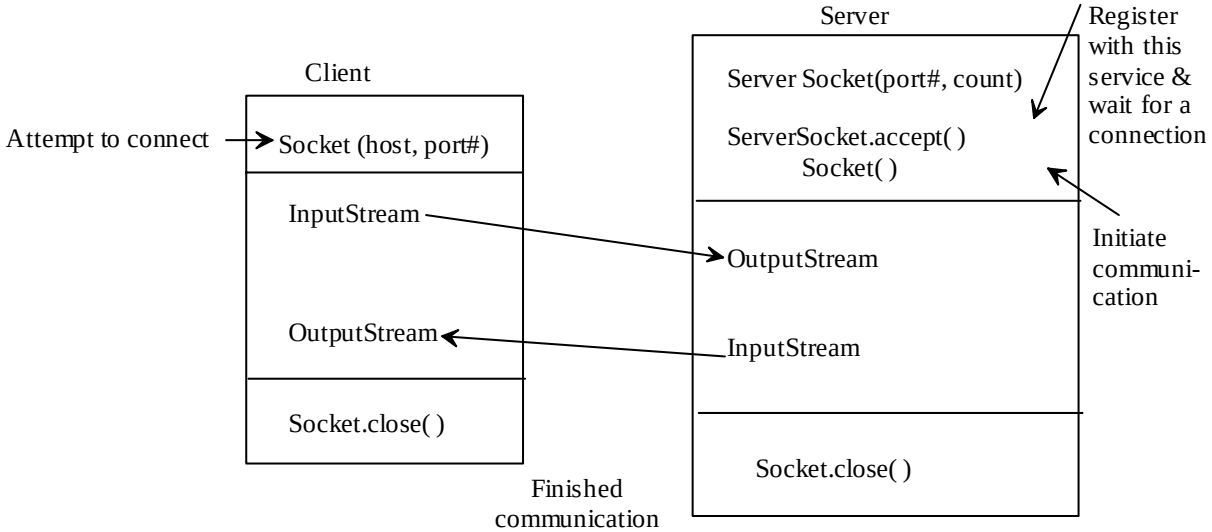


Figure: Client-Server model for TCP communication

With this discussion, we can establish a TCP connection. Suppose, we want to write a server system as SimpleServer. Using any one constructors in ServerSocket we can create a new socket (listening) on our machine that can accept incoming connections from clients across the network. The port number we have to select between 0 - 65535 uniquely (0 -1023 is reserved for standard Internet ports). Suppose, we select a port

number for our SimpleServer as 9876 (we have chosen arbitrarily for testing purpose). This is sufficient to create a new ServerSocket object in our machine. Once a new ServerSocket object is created, it is then ready to listen on its port form client requests to arrive. This can be initiated by invoking the accept ( ) method on the SeverSocket object.

Following is the Example 3 to create a SimpleServer in our machine. This server will send a message "Hello Net world!" whenever a client connects to it.

Example 3

```
// Minimal TCP/ IP Sever //

import java.net.*;
import java.io.*;
class SimpleServer {
      public static void main (String args [ ] )    {
      ServerSocket server;
      Socket socket;
      String msg = "Hello Net world !";
      OutputStream oStream;
      DataOutputStream dataOut;


      try {                      // To create a ServerSocket object
            Server = new ServerSocket (9876);
      } catch (IOException e )  {  }
                  //    Run the Server to attend the clients for ever
      while (true) {
            try {
                  socket = server.accept (  );     //  Wait  here  and  listen  for  a
connection.
                  oStream  =  socket  getOutputStream(  ); //Get  a  communication  stream
for output
                  dataOut  =  new  DataOutputStream(oStream);    //  DataStream  out  in
binary
                  dataOut.writeUTF(mag);            // Send message to the client
                  dataOut.close ( );          // Close the Stream object when message
is transmitted
                  oStream.close ( );        // Close the output Stream or the current
socket
                  socket.close ( );         // Close the current socket connection
            }  catch (IOException e )  {  }
           }      // listen for other clients if any
      }     // main (  )
}     // Simple Server
```

Note : that this Application uses DataOutputStream to send message in binary bit string format for portability only. The writeUTF(String) is a function defined in DataOutputStream class to write a string object.

Following Example 4 is the simple example for a client program.

Example 4

```
// Minimal TCP/IP  client  //

import java.net.*;
import java.io.*;
class SimpleClient  {
      public static void main (String args [ ] ) throws IOException     {
            int c;
            Socket socket;
            InputStream iStream;
            DataInputStream dataIn;
```

```
            socket = new Socket ("New Server", 9876 );
                              // Create socket to connect the server "New Server" at
port 9876
            iStream = socket getInput.Stream ( );
            dataIn  = new dataInputStream ( iStream);         // Get an input stream in
      binary
            String mag = new String (dataIn.readUTF( ) );   //      Read the string in
      Distream
            System.out.println(mgg);
            // When done just close the connection and exit
            dataIn.close ( );
            iStream.close ( );
            socket.close ( );               // Close the connection
      }
}     // SimpleClient program
```

Example 5
Following is an application that opens a connection to a Web server and adds two numbers from the server connection. This is treated as the Web server as iterative, that is, the server processes one client's request at a time.

```
// Client Side Program (Simple Web Client)//
import java.io.*;
import java.net.*;

public class SimpleWebClient {
       static DataOutputStream outbound;
       static DataInputStream inbound;
       static InputStreamReader reader;
       static BufferedReader br;
    public static void main(String args[])
    {
        try
        {
            // Open a client socket connection
            Socket clientSocket1 = new Socket("localhost", 2000);
            System.out.println("Client1: " + clientSocket1);
                    String choice="y";
                    reader = new InputStreamReader(System.in);
                    br = new java.io.BufferedReader(reader);

           outbound = new DataOutputStream(
                clientSocket1.getOutputStream() );

            inbound = new DataInputStream(
                clientSocket1.getInputStream() );

                    while(choice.equals("y"))
                    {
                           int num1, num2, result;

                           System.out.println("Enter first Number");
                           num1 = Integer.parseInt(br.readLine());

                           System.out.println("Enter second Number");
                           num2 = Integer.parseInt(br.readLine());

                           outbound.writeInt(num1);
                           outbound.writeInt(num2);
                                          result = inbound.readInt();
                           System.out.println("Sum = " + result);
```

```
                        System.out.print("Want to add more (y/n) : ");
                        choice=br.readLine();
                    }
                        // Clean up
                        outbound.close();
                        inbound.close();
                        clientSocket1.close();
        }
        catch (UnknownHostException uhe)
        {
            System.out.println("UnknownHostException: " + uhe);
        }
        catch (IOException ioe)
        {
            System.err.println("IOException: " + ioe);
        }
    }
}

// Sever Side Program (Simple Web Sever)//
import java.io.*;
import java.net.*;
import java.lang.*;

class SimpleWebServer
{
       public static void main(String args[])
       {
              ServerSocket serverSocket = null;
              Socket clientSocket = null;
              ServiceClient sock;
              int connects = 0;
              try
              {
                     // Create the server socket
                     serverSocket = new ServerSocket(2000, 5);
                     System.out.println("server started");
                     while (connects < 5)
                     {

                            clientSocket = serverSocket.accept();
                            sock = new ServiceClient(clientSocket);
                            sock.start();
                            connects++;
                     }
                     System.out.println("Closing server");
                     serverSocket.close();
              }

              catch (IOException ioe)
              {
                     System.out.println("Error in SimpleWebServer: " + ioe);
              }

       }

}


class ServiceClient extends Thread
{
              Socket client;
               DataInputStream inbound ;
```

```
            DataOutputStream outbound ;

            ServiceClient(Socket clnt)
            {
                  client=clnt;
            }

         public void run()
         {           inbound = null;
                          outbound = null;
            try
            {
                  // Acquire the streams for IO
                  inbound = new DataInputStream( client.getInputStream());
                  outbound = new DataOutputStream( client.getOutputStream());

                  int num1, num2,result;


                  while(true)
                  {
                        //Accept two numbers
                  num1 = inbound.readInt();
                  num2 = inbound.readInt();
                  result=num1+num2;

                  //Send the result to client
                  outbound.writeInt(result);
                  }

       }
       catch(Exception e)
                  {

                  // Clean up
                  System.out.println("Cleaning up connection: " + client);
                        try
                        {
                  outbound.close();
                  inbound.close();
                  client.close();
                        }
                        catch(Exception r)
                        {
                        }

                  }
               }

}
```

Assignment:
Modify the same application for a concurrent server.

```
/****************** Client-side code *********************/

import java.io.*;
import java.net.*;
```

```
/**
 * An application that opens a connection to a Web server and adds two numbers from the
server connection.
*/

public class SimpleWebClient {
       static DataOutputStream outbound;
       static DataInputStream inbound;
       static InputStreamReader reader;
       static BufferedReader br;
    public static void main(String args[])
    {

        try
        {
            // Open a client socket connection
            Socket clientSocket1 = new Socket("localhost", 2000);
            System.out.println("Client1: " + clientSocket1);
                    String choice="y";
                    reader = new InputStreamReader(System.in);
                    br = new java.io.BufferedReader(reader);

           outbound = new DataOutputStream(
                clientSocket1.getOutputStream() );

            inbound = new DataInputStream(
                clientSocket1.getInputStream() );

                    while(choice.equals("y"))
                    {


                           int num1, num2, result;

                           System.out.println("Enter first Number");
                           num1 = Integer.parseInt(br.readLine());

                           System.out.println("Enter second Number");
                           num2 = Integer.parseInt(br.readLine());

                           outbound.writeInt(num1);
                           outbound.writeInt(num2);
                    result = inbound.readInt();
                           System.out.println("Sum = " + result);



                           System.out.print("Want to add more (y/n) : ");
                           choice=br.readLine();
                    }
                           // Clean up
                           outbound.close();
                           inbound.close();
                           clientSocket1.close();
        }
        catch (UnknownHostException uhe)
        {
            System.out.println("UnknownHostException: " + uhe);
        }
        catch (IOException ioe)
        {
            System.err.println("IOException: " + ioe);
        }
```

```
    }

}


/****************** Client-side code *********************/

import java.io.*;
import java.net.*;
import java.lang.*;

class SimpleWebServer
{
       public static void main(String args[])
       {
              ServerSocket serverSocket = null;
              Socket clientSocket = null;
              ServiceClient sock;
              int connects = 0;
              try
              {
                     // Create the server socket
                     serverSocket = new ServerSocket(2000, 5);
                     System.out.println("server started");
                     while (connects < 5)
                     {

                            clientSocket = serverSocket.accept();
                            sock = new ServiceClient(clientSocket);
                            sock.start();
                            connects++;
                     }
                     System.out.println("Closing server");
                     serverSocket.close();
              }

              catch (IOException ioe)
              {
                     System.out.println("Error in SimpleWebServer: " + ioe);
              }

       }

}


class ServiceClient extends Thread
{
              Socket client;
               DataInputStream inbound ;
              DataOutputStream outbound ;

              ServiceClient(Socket clnt)
              {
                     client=clnt;
              }

         public void run()
         {           inbound = null;
                          outbound = null;
              try
```

```
            {
                  // Acquire the streams for IO
                  inbound = new DataInputStream( client.getInputStream());
                  outbound = new DataOutputStream( client.getOutputStream());

                  int num1, num2,result;


                  while(true)
                  {
                        //Accept two numbers
                  num1 = inbound.readInt();
                  num2 = inbound.readInt();
                  result=num1+num2;

                  //Send the result to client
                  outbound.writeInt(result);
                  }

      }
      catch(Exception e)
                  {

                  // Clean up
                  System.out.println("Cleaning up connection: " + client);
                        try
                        {
                  outbound.close();
                  inbound.close();
                  client.close();
                        }
                        catch(Exception r)
                        {
                        }

                  }

       }

}
```