Tentative OOPSJ LAB IA programs /Lab record programs

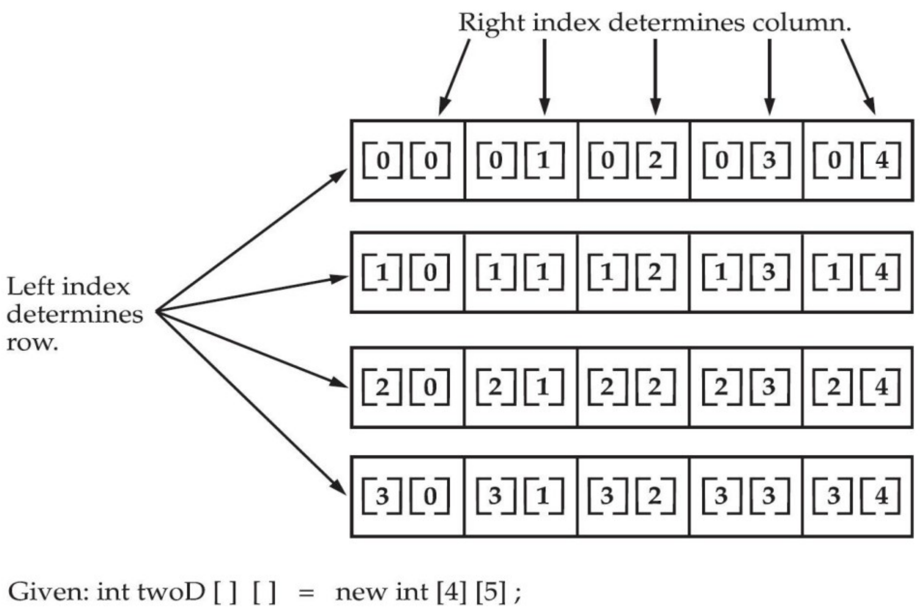
Experiment 1

Q1. /*Program 01 : Matrix Addition
Develop a JAVA program to add TWO matrices of suitable order mxn and rxp
The input values should be read from command line arguments) and print the resultant matrix on console. */

Multidimensional Arrays

In Java, multidimensional arrays are implemented as arrays of arrays. To declare a multidimensional array variable, specify each additional index using another set of square brackets. For example, the following declares a two-dimensional array variable called twoD:

This allocates a 4 by 5 array and assigns it to twoD. Internally, this matrix is implemented as an array of arrays of int. Conceptually, this array will look like the one shown in Figure



```
// defining package
package first;

// importing Scanner class for getting an input through keyboard
import java.util.Scanner;

// starting main method

public class MatrixAddition {
   public static void main(String[] args)
   {
        // creating an object sc for Scanner class
        Scanner sc= new Scanner(System.in);

     //  declaring variables m,n
     int m,n ;

     System.out.println(" Enter the order of first matrix m and n.");

     // getting m and n as integer numbers  through keyboard
     m=sc.nextInt();
     n=sc.nextInt();

 //  declaring variables r,p;
     int r,p ;

     System.out.println(" Enter the order of second matrix r and p.");

  // getting m and n as integer numbers  through keyboard
     r=sc.nextInt();
     p=sc.nextInt();


     // Check if  a positive integer
     if (r <= 0 || p<=0 || m<=0 || n<=0) {
        System.out.println("Please provide a valid positive integer for the order of the matrix");
        return;
     }

     // Check if addition is possible

     if (m!=r  || n!=p ) {
        System.out.println("Please provide  same order for the both matrices.");
        return;
     }

     // Declaring three matrices with right order
     int[][] matrix1 = new int[m][n];
     int[][] matrix2 = new int[r][p];
     int[][] resultMatrix = new int[m][n];

     System.out.println("\nEnter the elements of Matrix1 :");
     // Calling getMatix to get the elements of matrix1
     getMatrix(matrix1, m,n);
     System.out.println("\nEnter the elements of Matrix2 :");
     // Calling getMatix to get the elements of matrix2
     getMatrix(matrix2, r,p);

     // Print the matrices
```

```
        System.out.println("Matrix 1:");
        printMatrix(matrix1,m,n);

        System.out.println("\nMatrix 2:");
        printMatrix(matrix2,r,p);

        // Add the matrices
        addMatrices(matrix1, matrix2,m,n, resultMatrix);

        // Print the resultant matrix
        System.out.println("\nResultant Matrix (Matrix1 + Matrix2):");
        printMatrix(resultMatrix,m,n);
    }

    // Helper method to get a matrix  with input values through keyboard
    private static void getMatrix(int[][] matrix, int m, int n) {
             int x;
             Scanner sc= new Scanner(System.in);
        for (int i = 0; i < m; i++) {
            for (int j = 0; j <n; j++) {
                matrix[i][j] = sc.nextInt();
            }
        }
    }

    // Helper method to add two matrices
    private static int[][] addMatrices(int[][] matrix1, int[][] matrix2, int m, int n, int[][] resultMatrix) {

        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                resultMatrix[i][j] = matrix1[i][j] + matrix2[i][j];
            }
        }

        return resultMatrix;
    }

    // Helper method to print a matrix
    private static void printMatrix(int[][] matrix, int m, int n) {
             for (int i = 0; i < m; i++) {
             for (int j = 0; j < n; j++) {
                         System.out.print(matrix[i][j]+" ");
             }
             System.out.println();
         }
        }
    }
```

i/p and o/p

```
Enter the order of first matrix m and n.
3 2
 Enter the order of second matrix r and p.
3 2

Enter the elements of Matrix1 :
1
2
3
4
5
6

Enter the elements of Matrix2 :
1
1
1
1
1
1
Matrix 1:
1 2
3 4
5 6

Matrix 2:
1 1
1 1
1 1

Resultant Matrix (Matrix1 + Matrix2):
2 3
4 5
6 7
```

Q2. Matrix Multiplication

Develop a JAVA program to multiply TWO matrices of suitable order m x n and r x p
The input values should be read from command line arguments) and print the resultant matrix on console.

[Hint : Matrices mx n and rxp

Check n should be equal to r for Matrix multiplication or should report

Perform the matrix multiplication as follows

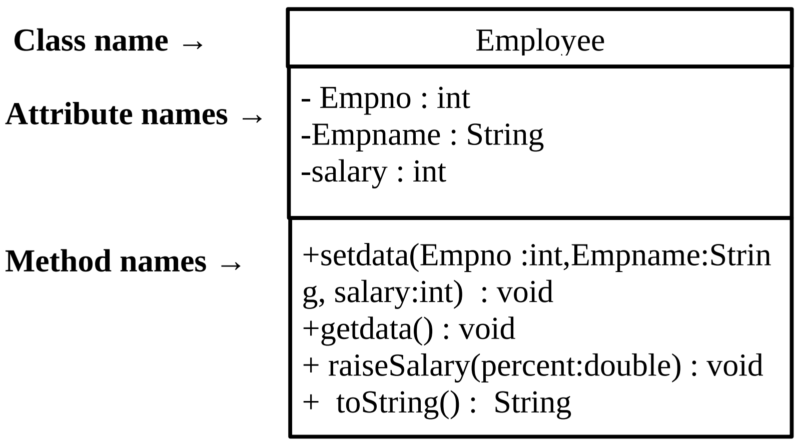
As the result matrix order is m x p

```
for ( i .. for m )
  for( j…for p)
      for (k... for n or p)
            resultmatrix[i][j]=resultmatrix[i][j]+matrix1[i][k]*matrix2[k][j];

]
```

Experiment 2

Q3. Employee Class
A class called Employee, which models an employee with an ID, name and salary, is designed as follows. The method raiseSalary (percent) increases the salary by the given percentage. Develop the Employee class and suitable main method for demonstration.

Class diagram of Employee Class



+ means public
- means private

```
package Emp;

class Employee
{
   // class member variable
   private int Empno;
   private String Empname;
   private int salary;

 // setter method to set data values
   public void setdata(int Empno,String Empname,int salary)
   {
      this.Empno=Empno;
```

```
        this.Empname=Empname;
        this.salary=salary;
    }
    // getter method to print or return
    public void getdata()
    {
         System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~");
         System.out.println("Employee's Report");
         System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~");
        System.out.println("Empno      : "+Empno);
        System.out.println("Empname : "+Empname);
        System.out.println("Salary       : "+salary);
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~");
    }
 // method for raising salary of employee
    public void raiseSalary(double percent) {
        if (percent > 0) {
            double raiseAmount = salary * (percent / 100);
            salary += raiseAmount;
            System.out.println(Empname + "'s salary raised by " + percent + "%. New salary details is");
        } else {
            System.out.println("Invalid percentage. Salary remains unchanged.");
        }
    }

// method to print a string when the object is printed in the main using toString() method
    public String toString()
    {
            return Empno+" "+Empname+" "+salary;
    }


}
// Main class.
public class  Getter_Setter
{
    // main method
    public static void main(String argvs[])
    {
        // Creating an object of the Employee class
        final Employee e = new Employee();

        // the employee details are getting set using the setter method.
        e.setdata(101,"Ram Kumar",34000);

        // Displaying the details of the employee details using the getter method before salary hike
        e.getdata();

        // Displaying the details of the employee details using the getter method after 10 % salary hike
```

```
        e.raiseSalary(5);

        // Displaying after raising salary using toString()

        System.out.println(e);

    }
}
```

o/p

```
~~~~~~~~~~~~~~~~~~~~~~~~
Employee's Report
~~~~~~~~~~~~~~~~~~~~~~~~
Empno   : 101
Empname : Ram Kumar
Salary  : 34000
~~~~~~~~~~~~~~~~~~~~~~~~
Ram Kumar's salary raised by 5.0%. New salary details is
101 Ram Kumar 35700
```

Q4. Student Class

A class called Student with data members Rollno, Sname , Mark1, Mark2, Mark3, Mark4, Mark5 & Mark6, Total, Result. The method calculate() is defined to calculate total and result and getdata() is defined to print the following details of Student Report using the main class called MarkReport

When the object s is created and set as s.setdata(14,"Ram Kumar", 45,56,67,56,80,50);

it has to be printed as follows when s.calculate() is used to calcuate total and result of the exam such as “passed “ or “failed” to store in the string variable Result and s.getdata() are invoked to print the following

```
~~~~~~~~~~~~~~~~~~~~~~~~
Student's Report
~~~~~~~~~~~~~~~~~~~~~~~~
Rollno          : 14
Student name  : Ram Kumar
Mark1           : 45
Mark2           : 56
Mark3           : 67
Mark4           : 56
Mark5           : 80
Mark6           : 50
~~~~~~~~~~~~~~~~~~~~~~~~
```

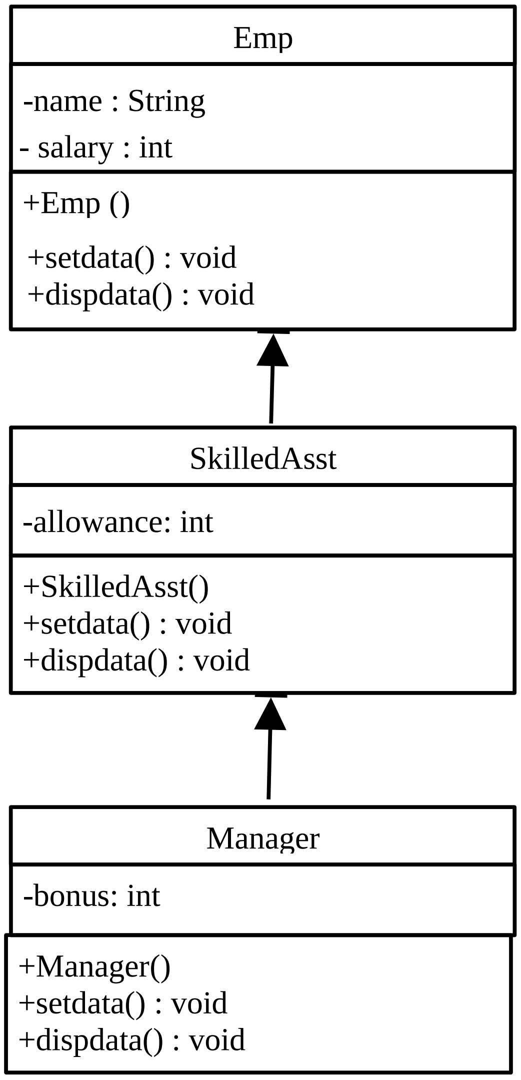
and finally the result has to be printed using toString() using System.out.println(obj) to print the following

```
Ram Kumar passed with total of 354.
```

Experiment 3

Q5. Develop a class Emp which includes data fields namely name, salary and a member function called setdata() to pass the parameters name and salary from main to class and print name and salary through the function dispdata() and Include a non-parameterized constructor to initialize the name to “” and salary to zero. Similarly inherit the class **SkilledAssist** from **Emp** with an extra attribute **allowance** and inherit Class **Manager** from SkilledAsst with an extra attribute **bonus. Develop a main Class EmpTest to create objects of Emp, SkilledAsst and Manager to pass the data fields and print them on the console.**

Class diagram of multilevel inheritance



Emp.java

```
package basic;

public class Emp {

	private String name;
	private int salary;

	Emp()
	{
		name=" ";
		salary=0;
	}

	public void setdata(String name, int salary)
	{
		this.name=name;
		this.salary=salary;
			}



	public void dispdata()
	{
		 System.out.println("Employee name : "+name);
		 System.out.println("Employee salary : "+salary);
	}

}
```

SkilledAsst.java

```
package basic;

public class SkilledAsst extends Emp{
		private int allowance;

	SkilledAsst()
	{
		super();
		allowance=0;
	}

	public void setdata(String name, int salary, int allowance)
	{
		super.setdata(name,salary);
		this.allowance=allowance;
			}
```

```
        public void dispdata()
        {
                 super.dispdata();
                 System.out.println("Employee's allowance : "+allowance);
        }
}
```

Manager.java

```
package basic;
public class Manager extends SkilledAsst  {

                private int bonus;
                Manager ()
                        {
                                super();
                                bonus=0;
                        }
        public void setdata(String name, int salary, int allowance, int bonus)
                        {
                        super.getdata(name,salary,allowance);
                        this.bonus=bonus;
                        }

        public void dispdata()
                {
                 super.dispdata();
                 System.out.println("Employee's bonus : "+bonus);
                }

        }
```

EmpTest.java

```
package basic;

import java.io.*;
public class EmpTest {
          public static void main(String args[]){

            Emp e = new Emp();
            SkilledAsst s = new SkilledAsst();
            Manager m= new Manager();

            System.out.println("Details immediately after declaring objects : ");

            e.dispdata();
            s.dispdata();
```

```
                m.dispdata();

                System.out.println("Details  after setting the data : ");
                e.setdata("Ram",1200);
                s.setdata("Ravi",1400,1000);
                m.setdata("Raja",1800,1500,2000);

              System.out.println("~~~~~~~~~~~~Employee's details~~~~~~~~~~~~ ");
                e.dispdata();
             System.out.println("~~~~~~~~~~~~SkilledAsst's details ~~~~~~~~~~~~ ");
                s.dispdata();
             System.out.println("~~~~~~~~~~~~Manager's details~~~~~~~~~~~~ ");
                m.dispdata();

              }
```

o/p

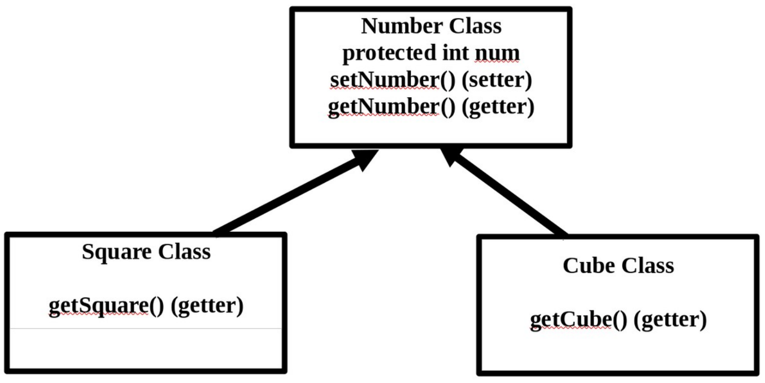
```
Details immediately after declaring objects :
Employee name :
Employee salary : 0
Employee name :
Employee salary : 0
Employee's allowance : 0
Employee name :
Employee salary : 0
Employee's allowance : 0
Employee's bonus : 0
Details  after setting the data :


~~~~~~~~~~~~Employee's details~~~~~~~~~~~~
Employee name : Ram
Employee salary : 1200
~~~~~~~~~~~~SkilledAsst's details ~~~~~~~~~~~~
Employee name : Ravi
Employee salary : 1400
Employee's allowance : 1000
~~~~~~~~~~~~Manager's details~~~~~~~~~~~~
Employee name : Raja
Employee salary : 1800
Employee's allowance : 1500
Employee's bonus : 2000
}
```

o/p

```
Details immediately after declaring objects :
Employee name :
```

```
Employee salary : 0
Employee name :
Employee salary : 0
Employee's allowance : 0
Employee name :
Employee salary : 0
Employee's allowance : 0
Employee's bonus : 0
Details  after setting the data :
~~~~~~~~~~~~~Employee's details~~~~~~~~~~~~~
Employee name : Ram
Employee salary : 1200
~~~~~~~~~~~~~SkilledAsst's details ~~~~~~~~~~~~~
Employee name : Ravi
Employee salary : 1400
Employee's allowance : 1000
~~~~~~~~~~~~~Manager's details~~~~~~~~~~~~~
Employee name : Raja
Employee salary : 1800
Employee's allowance : 1500
Employee's bonus : 2000
```

Q6. Java program to demonstrate example of hierarchical inheritance to get square and cube of a number as follows



Procedure :

First create the class Number with protected data member num (scope must be protected) and member functions setNumber() and getNumber() to set data and return the values to main class.
Create a classes Square with getSquare() to return the result of square of the number and Cube with getCube() to return the result of the Cube

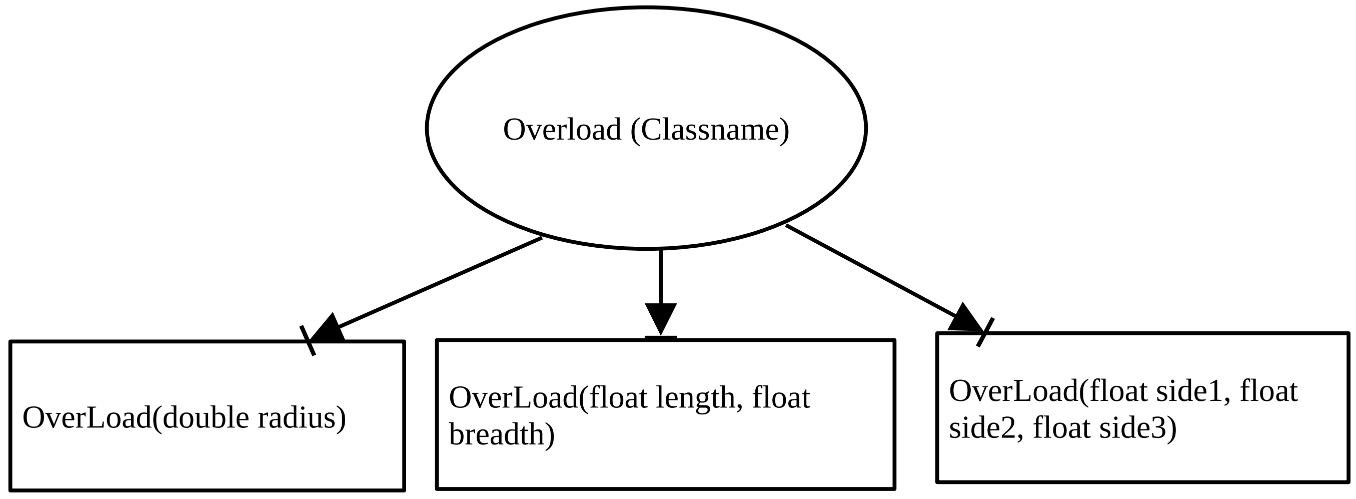
Create a main class Main_Class to create objects for the classes Number, Square, Cube to set data member from main class to the class Number through setNumber() to get an input through key board and getNumber() to return the calculated results through getSquare() and getCube() from Square and Cube classes to main

Input and Output

```
Enter an integer number: 3
The square of the number 3 is 9
The cube of the number 3 is 27
```

Experiment 4

Q7. Build a Java programming for finding areas of circle,rectangle,triangle(three sides given) using constructor overloading and method overloading techniques.The data members have to be passed from main to class and hide the data members in the class Develop three objects using parameterized constructor overloading and print the results using toString() .


Overload (Classname)
OverLoad(double radius)
OverLoad(float length, float breadth)
OverLoad(float side1, float side2, float side3)

```
package method_overloading;
//Method and constructor overloading
class OverLoad {
private double radius,length,breadth,side1,side2,side3,s,area;
private String name;
// using constructor overloading
OverLoad(double radius)
{
this.radius=radius;
area=Math.PI*this.radius*this.radius;
name="circle";
}
OverLoad(float length, float breadth)
{
```

```
this.length=length;
this.breadth=breadth;
area=this.length*this.breadth;
name="rectangle";
}
OverLoad(float side1, float side2, float side3) {
this.side1=side1;
this.side2=side2;
this.side3=side3;
float s=(side1+ side2+ side3)/2;
name="triangle";
area=Math.sqrt(s*(s-side1)*(s-side2)*(s-side3));
}
public String toString()
{
return "Area of "+name+" is "+ String.format("%.2f",area);
}
}
public class method_overloading1{
public static void main(String args[]) {
OverLoad c = new OverLoad(5.1f);OverLoad r = new OverLoad(5.1f, 8.12f);
OverLoad t = new OverLoad(6.2f, 12.5f,16.5f);
System.out.println(c);
System.out.println(r);
System.out.println(t);
}
}
```

```
Area of circle is 81.71
Area of rectangle is 41.41
Area of triangle is 33.55
```

Q8. Build a Java programming for finding greatest of three, four and five numbers by creating a class namely Greatest using constructor overloading and method overloading techniques. The data members (one,two, three, four and five) have to be passed from main class (namely Main_Class) to the class Greatest and hide all the data members in the class. Develop three objects namely g3, g4 and g5 using parameterized constructor overloading and print the results using toString().

Experiment 5

Q9. Build a Java program for finding interest for the investment from different banks which offer different rate interests such as SBI (8%), ICICI(9%) and AXIS (10%) using the concept of method overriding. Develop a parent class BANK with data members Investment_amount and rate_of_interest initialized to zero. Print results of interests from different banks using different subclasses namely SBI, ICICI and AZIX.The data members (Investment_amount has to be passed from main to class and hide the data member in the class. Print the results by invoking getInterest() with respective objects in Main class named Override

```
package Polymorphism;
//Java Program to demonstrate the real scenario of Java Method Overriding
//where three classes are overriding the method of a parent class.
//Creating a parent class.
class Bank{
            int amount;
            double rate_of_interest;
            Bank(int amount, double rate_of_interest)
            {
                  this.rate_of_interest= rate_of_interest;
                           this.amount=amount;
            }
                  public double getInterest(){
                                    return this.rate_of_interest*this.amount/100.0;
                                    }
         }

//Creating child classes.
class SBI extends Bank{

         SBI(int amount, double rate_of_interest)
         {
         super(amount,rate_of_interest);
         }
         public double getInterest(){
         return super.getInterest();
         }
}

class ICICI extends Bank{

         ICICI(int amount, double rate_of_interest)
         {
                  super(amount,rate_of_interest);
         }
         public double getInterest(){
                  return super.getInterest();
                  }
         }
```

```
class AXIS extends Bank{
            AXIS(int amount, double rate_of_interest)
            {
                         super(amount,rate_of_interest);
            }
            public double getInterest(int amount){
                        return super.getInterest();
                        }
            }
//Test class to create objects and call the methods
class Override{
                        public static void main(String args[]){
                                    Bank   b= new Bank(0,0);
                                    SBI    s=new SBI(10000,8.0);
                                    ICICI  i=new ICICI(10000,9.0);
                                    AXIS   a=new AXIS(10000,10.0);
            System.out.println("Bank Interest amount is :"+b.getInterest());
            System.out.println("SBI Interest amount is    :"+s.getInterest());
            System.out.println("ICICI Interest amount is :"+i.getInterest());
            System.out.println("AXIS Interest amount is             :"+a.getInterest());
                        }
}
```

o/p

```
Bank Interest amount is :0.0
SBI Interest amount is    :800.0
ICICI Interest amount is  :900.0
AXIS Interest amount is  :1000.0
```

Q10. Develop a class Mobile with parameterized constructor with parameters Manufacturer, OS, Model, Cost to set the data from main class called Overriding_Demo and define a member function dispdata() to display all the parameters in the class Mobile. Develop the subclasses namely Apple,Android and Blackberry inherited from Mobile. Develop parameterized constructors in the derived Classes Apple,Anroid and Blackberry using super(Manufacturer, OS, Model, Cost) and print the following details using the member functions in the derived Classes Apple,Anroid and Blackberry with super.dispdata() .

o/p

```
manunfacturer     :Apple
operating_system :Appleios
model                   :Delux
cost                     :75000
*********************************************************************************
operating_system :Android
model                   :Grand
cost                     :30000
*********************************************************************************
manunfacturer    :BlackBerry
operating_system :RIM
model                   :Curve
cost                     :20000
******************************************************************************
```

Experiment 6

Q 11. Build a Java program for finding interest for the investment from different banks which offer different rate interests such as SBI (8%), ICICI(9%) and AXIS (10%) using the concept of abstract class. Develop a parent class BANK with data members amount and rate_of_interest with protected visibility and define an abstract method getInterest(int amount,double rate_of_interest). Print results of interests from different banks using different subclasses namely SBI, ICICI and AZIX. The data members amount and rate_of_intererst have to be passed from main to class and hide the data member in the sub classes. Print the results by invoking getInterest() with respective objects in Main class named Abstract_Class

```
package Abstract_Class;
//Java Program to demonstrate the real scenario of Java abstract class
//where three classes are making use of a method which is declared as an abstract in the parent class.

//Creating a parent class with keyword abstract and  define an abstract method without any
implementation in it.
          abstract class Bank{
                              protected int amount;
                              protected double rate_of_interest;
                                        // defining an abstract method without any implementation.
                              abstract public  double getInterest(int amount,double rate_of_interest);
                              }
//Creating child classes.
          class SBI extends Bank{
          public double getInterest(int amount,double rate_of_interest){
                                        this.amount=amount;
                                        this.rate_of_interest=rate_of_interest;
                                        return (amount*rate_of_interest/100.0);}
                    }

class ICICI extends Bank{
          public double getInterest(int amount,double rate_of_interest){
                    this.amount=amount;
                    this.rate_of_interest=rate_of_interest;
                    return (amount*rate_of_interest/100.0);}
}


class AXIS extends Bank{
          public double getInterest(int amount,double rate_of_interest){
                    this.amount=amount;
                    this.rate_of_interest=rate_of_interest;
                    return (amount*rate_of_interest/100.0);}
}


//Test class to create objects and call the methods
class Abstract_Class{
```

```
                public static void main(String args[]){
                        SBI s=new SBI();
                        ICICI i=new ICICI();
                        AXIS a=new AXIS();

        System.out.println("SBI Interest is: "+s.getInterest(10000,8));
        System.out.println("ICICI Interest is: "+i.getInterest(10000,9));
        System.out.println("AXIS Interest is: "+a.getInterest(10000,10));
                }
}
```

```
o/p
SBI Interest is: 800.0
ICICI Interest is: 900.0
AXIS Interest is: 1000.0
```

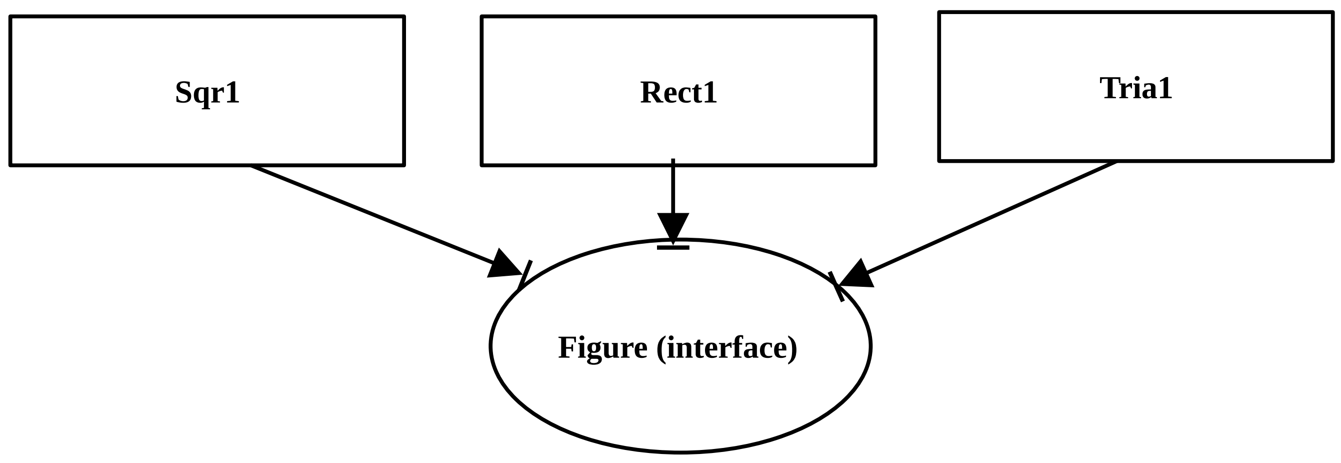
Q12. Develop an abstract class called Mobile with an abstract method dispdata() and data members Manufacturer, OS, Model, Cost to display all the parameters in the class as follows. Develop the subclasses namely Apple,Android and Blackberry inherited from Mobile to implement abstract method dispdata() in the respective class.

```
manunfacturer    :Appleios
operating_system   :Apple
model              :Delux
cost              :75000
******************************************************************************
manunfacturer    :Samsung
operating_system   :Android
model              :Grand
cost              :30000
******************************************************************************
manunfacturer    :BlackB
operating_system   :RIM
model              :Curve
cost              :20000
******************************************************************************
```

Experiment 7

Q13. Build a Java program for finding the areas of different shapes like square, rectangle, triangle(3 sides given) with respective parameters. Implement a common interface called figure which implements three classes namely Sqr1, Rect1 and Tria1 with parameterized constructors of respective dimensions. Create a reference to the interface and store the objects of the classes into the reference pointer of the interface in the main class.

Interface diagram



```
package Interface2;

interface figure {
            double Area();
    }

// subclass implementing interface
class Rect1 implements figure{
            int  length,breadth;
            Rect1(int length,int breadth)
            {
                        this.length=length;
                        this.breadth=breadth;
            }
            public double Area()
            {

                        return length*breadth;
            }
}
```

```
//subclass implementing interface
class Tria1 implements figure{
        int side1,side2,side3;
        Tria1(int side1,int side2,int side3)
        {
                this.side1=side1;
                this.side2=side2;
                this.side3=side3;
        }

        public double Area()
                {
                        float s=(side1+side2+side3)/2;
                        return Math.sqrt(s*(s-side1)*(s-side1)*(s-side1));
                }
        }

//subclass implementing interface
class Sqr1 implements figure{
        int side;
        Sqr1(int side)
        {
                this.side=side;
        }
        public double Area()
        {
                return side*side;
        }
}

// main class
public class  Interface_Construct   {

        public static void main (String[] args)
      {
                 // creating a reference to the interface
                   figure f;
                 // creating an object for the class
                   Rect1 r= new Rect1(5,8);
                 // storing the object into the reference pointer
                   f=r;

                   System.out.println("The area of rectangle is "+f.Area());

                   Tria1 t= new Tria1(5,6,7);
                   f=t;

                   System.out.println("The area of triangle is  "+f.Area());
                   Sqr1 s= new Sqr1(5);
                   f=s;

                   System.out.println("The area of square is    "+f.Area());                              }
                }
```

o/p

```
The area of rectangle is 40.0
The area of triangle is  24.0
The area of square is    25.0
```

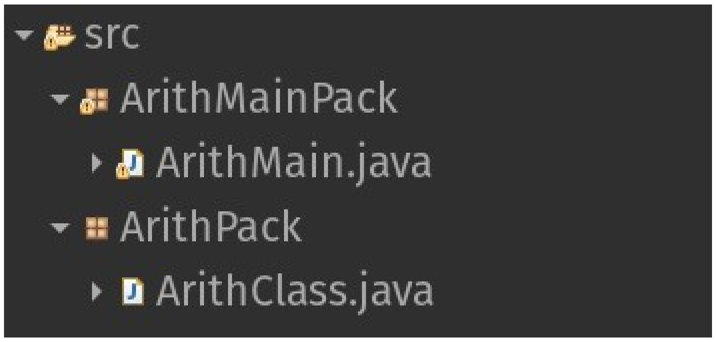
Q14. Write a Java programming to create a banking system with two classes - SavingsAccount, and CurrentAccount with attributes namely balance and interestRate which implement an interface named Account with the methods named deposit(double amount), withdraw(double amount), applyInterest() and getBalance() to deposit, withdraw, calculate interest, and view balances respectively. Include the constructors in the aforementioned two classes to initialize the attributes Initial balance and InterestRate. Display the transaction details as shown below.

o/p

```
Checking Initial Balance of SavingsAccount
Current balance in SavingsAccount is 10000.0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Withdrawing 1000 from SavingsAccount
Checking the current Balance of SavingsAccount
Current balance in SavingsAccount is 9000.0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Applying interest to the SavingsAccount
Current balance in SavingsAccount is 9900.0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Checking Initial Balance of CurrentAccount
Current balance inCurrentAccount  is 10000.0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Withdrawing 1000 from CurrentAccount
Checking the current Balance of CurrentAccount
Current balance in CurrentAccount  is 9000.0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Applying interest to the CurrentAccount
Current balance in CurrentAccount  is 10170.0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Experiment 8

Q15. Develop a Java program for performing arithmetic operations of two numbers x and y got through key board. There are two packages as shown below.



The below programs show how a class from different package can be accessed.

```
package ArithMainPack;

//ArithMain.java which is in the package ArithMainPack


import java.util.Scanner;
//creating object for ArithClass which is in the package called ArithPack
import ArithPack.ArithClass;


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

                        Scanner sc = new Scanner(System.in);

                         // creating object for ArithClass which is in the package called ArithPack
                         ArithClass a = new ArithClass();

while(true)
{

        System.out.println("Arithmatic operations");
        System.out.println("Please enter 1-Add,2-Sub,3-Mul,4-Div 5-Moddiv 6-exit ");
        int choice = sc.nextInt();
        System.out.println("Please enter two nos to do arithmatic operaions");
                         int x=sc.nextInt();
                         int y=sc.nextInt();
```

```
    if (choice==6)
    {
            System.out.println("\n exiting");
            break;
    }

            switch(choice)
                    {
                    case 1 :
                            System.out.println("\n Sum is "+a.Add(x,y));
                            break;
                    case 2 :
                            System.out.println("\n Difference is "+a.Sub(x,y));
                            break;
                    case 3 :
                            System.out.println("\n Multiplication is "+a.Mul(x,y));
                            break;
                    case 4 :
                            System.out.println("\n Division is "+a.Div(x,y));
                            break;
                    case 5 :
                            System.out.println("\n Mod Division is "+a.ModDiv(x,y));
                            break;
                    }

            }
     }
}
// accessing ArithClass module from ArithPack

package ArithPack;

public class ArithClass {

          public int Add(int a, int b)
          {
             return a+b;
          }
          public int Sub(int a,int b)
          {
             return a-b;
          }

          public int Mul(int a,int b)
          {
             return a*b;
          }

          public int Div(int a,int b)
          {
             return a/b;
          }
          public int ModDiv(int a,int b)
          {
             return a%b;
          }

           }
```
o/p

```
Arithmatic operations
Please enter 1-Add,2-Sub,3-Mul,4-Div 5-Moddiv 6-exit
1
Please enter two nos to do arithmatic operaions
4 5

Sum is 9
Arithmetic operations
Please enter 1-Add,2-Sub,3-Mul,4-Div 5-Moddiv 6-exit
2
Please enter two nos to do arithmatic operaions
5 6

 Difference is -1
Arithmetic operations
Please enter 1-Add,2-Sub,3-Mul,4-Div 5-Moddiv 6-exit
3
Please enter two nos to do arithmatic operaions
5 6

 Multiplication is 30
Arithmetic operations
Please enter 1-Add,2-Sub,3-Mul,4-Div 5-Moddiv 6-exit
4
Please enter two nos to do arithmatic operaions
5 6

 Division is 0
Arithmetic operations
Please enter 1-Add,2-Sub,3-Mul,4-Div 5-Moddiv 6-exit
5
Please enter two nos to do arithmatic operaions
16 5

 Mod Division is 1
Arithmetic operations
Please enter 1-Add,2-Sub,3-Mul,4-Div 5-Moddiv 6-exit
6
Please enter two nos to do arithmatic operaions
```

```
5 6

exiting
```

Q16. Develop a Java program for performing relational operations of two numbers x and y got through key board using modules RelateMain.java from Relateclass package as shown below

o/p

```
Relational operations
1.CheckGreater
2. CheckSmaller
3. CheckEqual
4. Exit

Please enter the operation which you prefer

3

Please enter two nos to relate

4 5

false

Please enter the operation which you prefer
Relational operations
1.CheckGreater
2. CheckSmaller
3. CheckEqual
4. Exit

1

Please enter two nos to relate

4 3

true

Please enter the operation which you prefer
Relational operations
1.CheckGreater
2. CheckSmaller
3. CheckEqual
4. Exit

3
```

```
Please enter two nos to relate

5 5

true
Please enter the operation which you prefer
Relational operations
1.CheckGreater
2. CheckSmaller
3. CheckEqual
4. Exit

4
bye ! Thanks for using me!
```

Experiment 9

Q17. Develop a stack class to hold a maximum of 10 integers with suitable methods. Develop a JAVA main method to illustrate Stack operations.

```
import java.util.Scanner;

public class Stack {
    private static final int MAX_SIZE = 10;
    private int[] stackArray;
    private int top;

    public Stack() {
        stackArray = new int[MAX_SIZE];
        top = -1;
    }

    public void push(int value) {
        if (top < MAX_SIZE - 1) {
            stackArray[++top] = value;
            System.out.println("Pushed: " + value);
        } else {
            System.out.println("Stack Overflow! Cannot push " + value + ".");
        }
    }

    public int pop() {
        if (top >= 0) {
            int poppedValue = stackArray[top--];
            System.out.println("Popped: " + poppedValue);
            return poppedValue;
        } else {
            System.out.println("Stack Underflow! Cannot pop from an empty stack.");
            return -1; // Return a default value for simplicity
        }
    }

    public int peek() {
        if (top >= 0) {
            System.out.println("Peeked: " + stackArray[top]);
            return stackArray[top];
        } else {
            System.out.println("Stack is empty. Cannot peek.");
            return -1; // Return a default value for simplicity
        }
    }

    public void display() {
```

```
    if (top >= 0) {
        System.out.print("Stack Contents: ");
        for (int i = 0; i <= top; i++) {
            System.out.print(stackArray[i] + " ");
        }
        System.out.println();
    } else {
        System.out.println("Stack is empty.");
    }
}

public boolean isEmpty() {
    return top == -1;
}

public boolean isFull() {
    return top == MAX_SIZE - 1;
}

public static void main(String[] args) {
    Stack stack = new Stack();
    Scanner scanner = new Scanner(System.in);

    int choice;

    do {
        System.out.println("\nStack Menu:");
        System.out.println("1. Push");
        System.out.println("2. Pop");
        System.out.println("3. Peek");
        System.out.println("4. Display Stack Contents");
        System.out.println("5. Check if the stack is empty");
        System.out.println("6. Check if the stack is full");
        System.out.println("0. Exit");

        System.out.print("Enter your choice: ");
        choice = scanner.nextInt();

        switch (choice) {
            case 1:
                System.out.print("Enter the value to push: ");
                int valueToPush = scanner.nextInt();
                stack.push(valueToPush);
                break;
            case 2:
                stack.pop();
                break;
            case 3:
                stack.peek();
```

```
                    break;
                case 4:
                    stack.display();
                    break;
                case 5:
                    System.out.println("Is the stack empty? " + stack.isEmpty());
                    break;
                case 6:
                    System.out.println("Is the stack full? " + stack.isFull());
                    break;
                case 0:
                    System.out.println("Exiting the program. Goodbye!");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 0);

        scanner.close();
    }
}
```

Output

```
$ java Stack

Stack Menu:
1. Push
2. Pop
3. Peek
4. Display Stack Contents
5. Check if the stack is empty
6. Check if the stack is full
0. Exit
Enter your choice: 4
Stack is empty.

Stack Menu:
1. Push
2. Pop
3. Peek
4. Display Stack Contents
5. Check if the stack is empty
6. Check if the stack is full
0. Exit
Enter your choice: 5
Is the stack empty? true
```

```
Stack Menu:
1. Push
2. Pop
3. Peek
4. Display Stack Contents
5. Check if the stack is empty
6. Check if the stack is full
0. Exit
Enter your choice: 6
Is the stack full? false

Stack Menu:
1. Push
2. Pop
3. Peek
4. Display Stack Contents
5. Check if the stack is empty
6. Check if the stack is full
0. Exit
Enter your choice: 1
Enter the value to push: 10
Pushed: 10

Stack Menu:
1. Push
2. Pop
3. Peek
4. Display Stack Contents
5. Check if the stack is empty
6. Check if the stack is full
0. Exit
Enter your choice: 1
Enter the value to push: 20
Pushed: 20

Stack Menu:
1. Push
2. Pop
3. Peek
4. Display Stack Contents
5. Check if the stack is empty
6. Check if the stack is full
0. Exit
Enter your choice: 4
Stack Contents: 10 20

Stack Menu:
1. Push
2. Pop
```

```
3. Peek
4. Display Stack Contents
5. Check if the stack is empty
6. Check if the stack is full
0. Exit
Enter your choice: 3
Peeked: 20

Stack Menu:
1. Push
2. Pop
3. Peek
4. Display Stack Contents
5. Check if the stack is empty
6. Check if the stack is full
0. Exit
Enter your choice: 1
Enter the value to push: 30
Pushed: 30

Stack Menu:
1. Push
2. Pop
3. Peek
4. Display Stack Contents
5. Check if the stack is empty
6. Check if the stack is full
0. Exit
Enter your choice: 4
Stack Contents: 10 20 30

Stack Menu:
1. Push
2. Pop
3. Peek
4. Display Stack Contents
5. Check if the stack is empty
6. Check if the stack is full
0. Exit
Enter your choice: 2
Popped: 30

Stack Menu:
1. Push
2. Pop
3. Peek
4. Display Stack Contents
5. Check if the stack is empty
6. Check if the stack is full
```

```
0. Exit
Enter your choice: 3
Peeked: 20

Stack Menu:
1. Push
2. Pop
3. Peek
4. Display Stack Contents
5. Check if the stack is empty
6. Check if the stack is full
0. Exit
Enter your choice: 4
Stack Contents: 10 20

Stack Menu:
1. Push
2. Pop
3. Peek
4. Display Stack Contents
5. Check if the stack is empty
6. Check if the stack is full
0. Exit
Enter your choice: 0
Exiting the program. Goodbye!
```

Experiment 10

Q18. Resizable interface
Develop a JAVA program to create an interface Resizable with methods resizeWidth(int width) and resizeHeight(int height) that allow an object to be resized. Create a class Rectangle that implements the Resizable interface and implements the resize methods.

Java Code

```
// Resizable interface
interface Resizable {
    void resizeWidth(int width);
    void resizeHeight(int height);
}

// Rectangle class implementing Resizable interface
class Rectangle implements Resizable {
    private int width;
    private int height;

    public Rectangle(int width, int height) {
        this.width = width;
        this.height = height;
    }

    // Implementation of Resizable interface
    @Override
    public void resizeWidth(int width) {
        this.width = width;
        System.out.println("Resized width to: " + width);
    }

    @Override
    public void resizeHeight(int height) {
        this.height = height;
        System.out.println("Resized height to: " + height);
    }

    // Additional methods for Rectangle class
    public int getWidth() {
        return width;
    }

    public int getHeight() {
        return height;
    }

    public void displayInfo() {
        System.out.println("Rectangle: Width = " + width + ", Height = " + height);
    }
}

// Main class to test the implementation
public class ResizeDemo {
    public static void main(String[] args) {
        // Creating a Rectangle object
        Rectangle rectangle = new Rectangle(10, 5);

        // Displaying the original information
        System.out.println("Original Rectangle Info:");
        rectangle.displayInfo();

        // Resizing the rectangle
        rectangle.resizeWidth(15);
        rectangle.resizeHeight(8);

        // Displaying the updated information
        System.out.println("\nUpdated Rectangle Info:");
        rectangle.displayInfo();
    }
}
```

In this program, the Resizable interface declares the methods resizeWidth and resizeHeight. The Rectangle class implements this interface and provides the specific implementation for resizing the width and height. The main method in the ResizeDemo class creates a Rectangle object, displays its original information, resizes it, and then displays the updated information.

Output

```
Original Rectangle Info:
Rectangle: Width = 10, Height = 5
Resized width to: 15
Resized height to: 8
```

Experiment 11

Q 19 : Build a Customized exception in the name of DivisionByZeroException which will be invoked when you try to divide a number by zero

```
// Custom exception class
class DivisionByZeroException extends Exception {
public DivisionByZeroException(String message) {
super(message);
}
}
public class CustomExceptionDemo {
// Method to perform division and throw custom exception if denominator is zero
static double divide(int numerator, int denominator) throws DivisionByZeroException {
if (denominator == 0) {
throw new DivisionByZeroException("Cannot divide by zero!");
}
return (double) numerator / denominator;
}
public static void main(String[] args) {
int numerator = 10;
int denominator = 0;
try {
double result = divide(numerator, denominator);
System.out.println("Result of division: " + result);
} catch (DivisionByZeroException e) {
System.out.println("Exception caught: " + e.getMessage());
} finally {
System.out.println("Finally block executed");
}
}
}
```

o/p

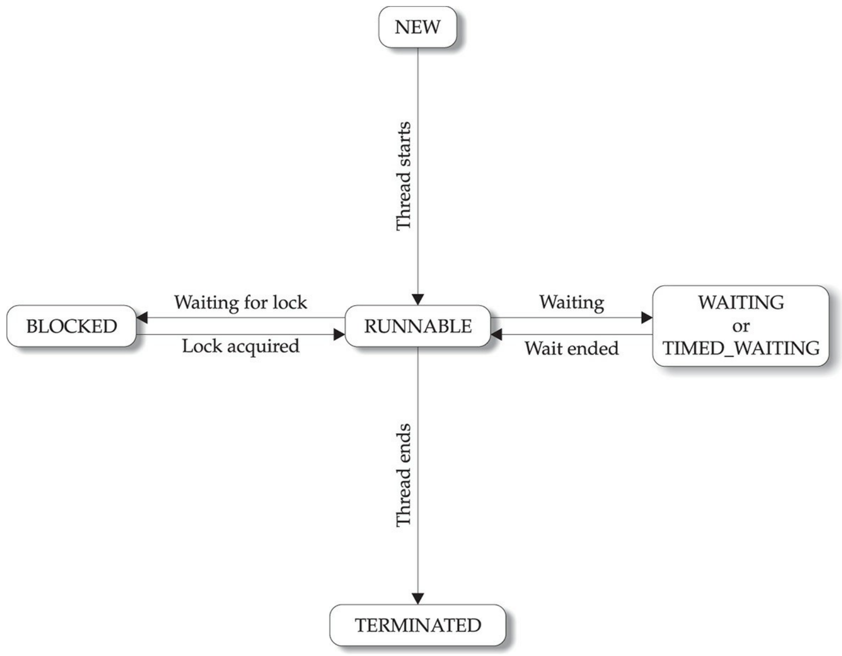
```
Exception caught: Cannot divide by zero!
Finally block executed
```

Exercise

Q20 custom exception: Create a custom exception called **InsufficientBalanceException** to represent a situation where a user tries to withdraw more money than is available in their bank account. (**refer to the Class notes Module 4**)

Experiment 12

Q21. : Write a Java program to illustrate five states of the life cycle of the thread (New, Runnable, Running, Non-Runnable (Blocked/Waiting), Terminated) with an example program.



A thread can be in one of the five states.The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:

1. New
2. Runnable
3. Running
4. Non-Runnable (Blocked/Waiting)
5. Terminated

Java Code

```
package Threads;


class ThreadStateExample extends Thread {

    @Override
    public void run() {
        // Thread is in running state when it enters here.
        System.out.println("Thread is running.");
        try {
            // Making the thread sleep to put it in Non-Runnable (Waiting) state.
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted.");
        }

        System.out.println("Thread is finished running.");
    }

    public static void main(String[] args) {
        // Thread is in New state when created but not started.
        ThreadStateExample thread = new ThreadStateExample();
        System.out.println("Thread state after creation: " + thread.getState());  // NEW

        // Thread is now in Runnable state after start() method is invoked.
        thread.start();
        System.out.println("Thread state after calling start: " + thread.getState());  // RUNNABLE

        try {
            // The main thread sleeps so we can see the RUNNING and WAITING state of the thread.
            Thread.sleep(500);  // Allows time for the thread to run
            System.out.println("Thread state while sleeping: " + thread.getState());  // TIMED_WAITING
(non-runnable state)

            // Wait for thread to die (terminated state).
            thread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        // Thread is in Terminated state after finishing execution.
        System.out.println("Thread state after execution: " + thread.getState());  // TERMINATED
    }
}
```

o/p

```
Thread state after creation: NEW
Thread state after calling start: RUNNABLE
Thread is running.
Thread state while sleeping: TIMED_WAITING
Thread is finished running.
Thread state after execution: TERMINATED
```

Experiment 13

Q22 : Construct a simple Java program that demonstrates the concept of the Multiple threads by creating Main Thread, a single thread, and multiple threads

```
package Threads;
//Main class to demonstrate multiple threads
public class MultiThreadExample  {

 public static void main(String[] args) {
     // Displaying the current thread (main thread)
     System.out.println("Main Thread ID: " + Thread.currentThread().getId());

     // Creating a single thread
     SingleThread singleThread = new SingleThread();
     singleThread.start(); // Starts the single thread

     // Creating multiple threads
     for (int i = 1; i <= 3; i++) {
         MultiThread multiThread = new MultiThread(i);
         multiThread.start(); // Starts each of the multiple threads
     }

     // Main thread continues running
     System.out.println("Main Thread ends.");
 }
}

//Class for creating a single thread by extending the Thread class
class SingleThread extends Thread {
 @Override
 public void run() {
     System.out.println("SingleThread ID: " + this.getId() + " is running.");
     try {
         Thread.sleep(1000); // Simulating some work
     } catch (InterruptedException e) {
         e.printStackTrace();
     }
     System.out.println("SingleThread ID: " + this.getId() + " has finished.");
 }
}

//Class for creating multiple threads by extending the Thread class
class MultiThread extends Thread {
 private int threadNumber;
 // Constructor to assign thread number
 public MultiThread(int threadNumber) {
     this.threadNumber = threadNumber;
 }
 @Override
 public void run() {
     System.out.println("MultiThread #" + threadNumber + " ID: " + this.getId() + " is running.");
     try {
         Thread.sleep(1000); // Simulating some work
     } catch (InterruptedException e) {
         e.printStackTrace();
     }
     System.out.println("MultiThread #" + threadNumber + " ID: " + this.getId() + " has finished.");
 }
}
```

o/p

```
Main Thread ID: 1
SingleThread ID: 14 is running.
MultiThread #1 ID: 15 is running.
MultiThread #2 ID: 16 is running.
Main Thread ends.
MultiThread #3 ID: 17 is running.
SingleThread ID: 14 has finished.
MultiThread #1 ID: 15 has finished.
MultiThread #2 ID: 16 has finished.
MultiThread #3 ID: 17 has finished.
```

Experiment 14

Q23 : passing objects into a class to perform the addition of two objects.

Add two objects of feet,inches by passing objects as parameters into a class called Measure with data members feet, inches and parameterized and non parameterized constructors to initialize the data members feet and inches . Use Calculate() to add the two objects and print the result object.

```
package Second;

//Defining class for adding two objecs of feet and inches
class Measure {
private int feet, inches;

Measure()
{
        this.feet=0;
        this.inches=0;
}

Measure(int feet, int inches) {
  this.feet = feet;
  this.inches = inches;
}

// adding two objects
Measure Calculate(Measure m) {
        Measure r = new Measure();
  r.feet=feet+m.feet;
  r.inches=(inches+m.inches)%12;
  r.feet+=(inches+m.inches)/12;
  return new Measure(r.feet,r.inches);

}

public String toString()
{

        return feet+" feet and "+inches+" inches";
}
}
```

```
public class FeetInchAddition {

	public static void main(String[] args) {
		// initialising data members in the class through parameterized constructor

		Measure m1=new Measure(4,10);
		Measure m2=new Measure(5,10);


		// performing addition of m1 and m2

		System.out.println(m1);
		System.out.println(m2);
		System.out.println(m1.Calculate(m2));

	}

}
```

o/p

```
4 feet and 10 inches
5 feet and 10 inches
10 feet and 8 inches
```

Q24 : Add two objects of Complex numbers C1 (3+4i) and C2 (4+5i) by passing objects as parameters into a class called Complex with data members real and Image and parameterized and non parameterized constructors to initialize the data members real and Image. Use Add() to add the two objects and print the result object.