Design a class called Person as described below:

Person

-name:String

-address:String

+Person(name,address)

+getName():String

+getAddress():String

+setAddress(address):void

A sub-class Employee of class Person is designed as shown below:

Employee

-empid:String

-dept:String

-basic:int

+Employee(name,address,empid,dept,basic)

+getEmpid():int

+getDept():String

+setDept(dept):void

+setBasic(basic):void

+getBasic():int

+calSalary():float

A sub-class Faculty of class Employee is designed as shown below:

Faculty

-designation:String

-course:String

+Faculty(name,address,empid,dept,basic,desig,course)

+getDesig():String

+setDesig(desig):void

+setCourse(course):void

+getCourse():float

+cal Salary():float

Design an Interface Student:

<<Student>>

+getMarks():float []

+calcGPA():float

Design a sub-class ResearchAssistant of class Employee, implements <<Student>>

ResearchAssistant

-project:String

-course:String

+ResearchAssitant(name,address,empid,dept,basic,project,course)

+getProject():String

+getCourse():String

+setCourse(course):void

+getMarks():float []

+cal c GPA():float

+cal Salary():float

import java.util.Scanner;

class Person{

private String name;

private String address;

public Person(String name,String address){

this.name=name;

this.address=address;

}

public String getName(){ return name; }

public String getAddress(){return address;}

public void setAddress(String address){

this.address=address;

}

}

class Employee extends Person{

private String empid;

private String dept;

private int basic;

public Employee(String name,String address,String empid,String dept,int basic){

super(name,address);

this.empid=empid;

this.dept=dept;

this.basic=basic;

}

public String getEmpid(){ return empid; }

public String getDept(){ return dept; }

public void setDept(String dept){

this.dept=dept;

}

public void setBasic(int basic){

this.basic=basic;

}

public double getBasic(){ return basic; }

public double calSalary(){

return 1.535\*getBasic();

}

}

class Faculty extends Employee{

String designation;

String course;

public Faculty(String name,String address,String empid,String dept,int basic,String desig,String course){

super(name,address,empid,dept,basic);

this.designation=desig;

this.course=course;

}

public String getDesig(){ return designation; }

public void setDesig(String desig){

this.designation=desig;

}

public void setCourse(String course){

this.course=course;

}

public String getCourse(){ return course; }

public double calSalary(){

return 1.535\*getBasic();

}

}

interface Student{

double[] getMarks();

double calcGPA();

}

class ResearchAssisstant extends Employee implements Student{

private String project;

private String course;

public ResearchAssisstant(String name,String address,String empid,String dept,int basic, String project,String course ){

super(name,address,empid,dept,basic);

this.project=project;

this.course=course;

}

public String getProject(){ return project; }

public String getCourse(){ return course; }

public void setCourse(String course){

this.course=course;

}

public double[] getMarks(){

Scanner in=new Scanner(System.in);

double marks[]=new double[3];

System.out.println("Enter the marks ");

for(int i=0;i<3;i++)

marks[i]=in.nextDouble();

return marks;

}

public double calcGPA(){

double marks[]=new double[3];

marks=getMarks();

double sum=0;

for (int i=0;i<3;i++)

sum+=marks[i];

double avg=sum/3;

return avg/100;

}

public double calSalary(){

return 1.535\*getBasic();

}

}

public class TestPerson{

public static void main(String args[]){

String name,address,empid,dept,project,course;

int basic;

double GPA;

Scanner in=new Scanner(System.in);

System.out.println("Enter the Research Assisstant details ");

System.out.print("Enter name: ");name=in.nextLine();

System.out.print("Enter address: ");address=in.nextLine();

System.out.print("Enter empid: ");empid=in.nextLine();

System.out.print("Enter department: ");dept=in.nextLine();

System.out.print("Enter project: ");project=in.nextLine();

System.out.print("Enter course: ");course=in.nextLine();

System.out.print("Enter basic: ");basic=in.nextInt();

ResearchAssisstant ra=new ResearchAssisstant(name,address,empid,dept,basic,project,course);

GPA=ra.calcGPA();

System.out.println("\n\nResearch Assisstant details ");

System.out.println("Name: "+ra.getName());

System.out.println("Address: "+ra.getAddress());

System.out.println("Employee ID: "+ra.getEmpid());

System.out.println("Department: "+ra.getDept());

System.out.println("Project: "+ra.getProject());

System.out.println("Course: "+ra.getCourse());

System.out.println("Salary: "+ra.calSalary());

System.out.println("GPA: "+GPA);

}

}

/\*

Output:

Enter the Research Assisstant details

Enter name: a

Enter address: a

Enter empid: 1

Enter department: cse

Enter project: asdfgf

Enter course: ;lkjhj

Enter basic: 10000

Enter the marks

95 84 93

Research Assisstant details

Name: a

Address: a

Employee ID: 1

Department: cse

Project: asdfgf

Course: ;lkjhj

Salary: 15350.0

GPA: 0.9066666666666667

\*/

Create a class hierarchy for the following using Interface / Abstract class:

Design Shape as described below:

Shape

#color:String=”red”

+Shape()

+Shape(color)

+getColor():String

+setColor(color):void

abs getArea():float

abs getPerimeter():float

Where abs – abstract method

A sub-class Circle of class Shape is designed as shown below:

Circle#radius:float=1.0

+Circle()

+Circle(radius)

+Circle(radius,color)

+getRadius():float

+setRadius(radius):void

+getArea():float

+getPerimeter():float

A sub-class Rectangle of class Shape is designed as shown below:

Rectangle

#width:float=1.0

#length:float=1.0

+Rectangle()

+Rectangle(width,length)

+Rectangle(width,length,color)

+getWidth():float

+setWidth(width):void

+getLength():float

+setLength(length):void

+getArea():float

+getPerimeter():float

A sub-class Square of class Rectangle designed as shown below:

Square

+Square()

+Square(side)

+Square(side,color)

+getSide():float

+setSide(side):void

+getArea():float

+getPerimeter():float

Note the following:

1. Shape contains the abstract methods.

2. Those abstract methods are to be implemented by the defining classes.

EXERCISE :1. Draw the class diagram of the above class hierarchy.

2. Implement the above class hierarchy by using Interface and Abstract class .

Hint:

To write an Interface:

a. Only abstract methods can be declared inside the Interface.

b. Identify the common behavior of the set of objects and declare that as abstract

methods inside the Interface.

c. The classes that implement the Interface will provide the actual implementation of

those abstract methods.

To write an Abstract class:

a. An abstract class can have constructor(s), abstract or non-abstract method(s).

b. Define the constructors and non-abstract method in the Abstract class Shape.

Declare the common behavior as the abstract method.

c. Let the classes Rectangle, Circle, Square define its own constructors, member

variable and methods.

3. Write a test driver called TestInterface | TestAbstract . Use an array of

objects of type Shape to display the area, perimeter of all the shapes (Circle,

Rectangle, Square).

4. Note down the differences while implementing the Inheritance through Interface

and Abstract class.

5. Note the run-time polymorphism in resolving the method call exhibited by Java

through method overriding.

import java.util.Scanner;

abstract class Shape{

protected String color;

public Shape(){

color="red";

}

public Shape(String color){

this.color=color;

}

public String getColor(){

return color;

}

public void setColor(String color){

this.color=color;

}

abstract double getArea();

abstract double getPerimeter();

}

interface Common{

abstract double getArea();

abstract double getPerimeter();

}

class Circle extends Shape implements Common{

protected double radius;

public Circle(){

radius=1.0;

}

public Circle(double radius){

this.radius=radius;

}

public Circle(double radius, String color){

this.radius=radius;

this.color=color;

}

public double getRadius(){

return radius;

}

public void setRadius(double radius){

this.radius=radius;

}

public double getArea(){

return (float)3.14\*radius\*radius;

}

public double getPerimeter(){

return (float)2\*3.14\*radius;

}

}

class Rectangle extends Shape implements Common{

protected double width;

protected double length;

public Rectangle(){

width=1.0;

length=1.0;

}

public Rectangle(double width,double length){

this.width=width;

this.length=length;

}

public Rectangle(double width,double length,String color){

super(color);

this.width=width;

this.length=length;

}

public Rectangle(String color){

super(color);

}

public double getWidth(){

return width;

}

public void setWidth(double width){

this.width=width;

}

public double getLength(){

return length;

}

public void setLength(){

this.length=length;

}

public double getArea(){

return (float)width\*length;

}

public double getPerimeter(){

return 2\*(width+length);

}

}

class Square extends Rectangle{

protected double side;

public Square(){

side=1.0;

}

public Square(double side){

this.side=side;

}

public Square(double side,String color){

super(color);

this.side=side;

}

public double getSide(){

return side;

}

public void setSide(double side){

this.side=side;

}

public double getArea(){

return side\*side;

}

public double getPerimeter(){

return side\*4;

}

}

public class TestInterface{

public static void main(String args[]){

String color;

double radius;

double width;

double length;

double side;

Scanner in=new Scanner(System.in);

System.out.print("Enter color of Circle: ");

color=in.nextLine();

System.out.print("Enter radius of Circle: ");

radius=in.nextDouble();

in.nextLine();

Circle c=new Circle(radius,color);

System.out.print("Enter color of Rectangle: ");

color=in.nextLine();

System.out.print("Enter width of Rectangle: ");

width=in.nextDouble();

System.out.print("Enter length of Rectangle: ");

length=in.nextDouble();

in.nextLine();

Rectangle r=new Rectangle(width,length,color);

System.out.print("Enter color of Square: ");

color=in.nextLine();

System.out.print("Enter side of Square: ");

side=in.nextDouble();

Square s=new Square(side,color);

System.out.println("Area of Circle :"+c.getArea());

System.out.println("Perimeter of Circle :"+c.getPerimeter());

System.out.println("Area of Rectangle :"+r.getArea());

System.out.println("Perimeter of Rectangle :"+r.getPerimeter());

System.out.println("Area of Square :"+s.getSide()\*s.getSide());

System.out.println("Perimeter of Square :"+4\*s.getSide());

}

}

/\*

Output:

Enter color of Circle: red

Enter radius of Circle: 5

Enter color of Rectangle: blue

Enter width of Rectangle: 5

Enter length of Rectangle: 6

Enter color of Square: green

Enter side of Square: 5

Area of Circle :78.50000262260437

Perimeter of Circle :31.400000000000002

Area of Rectangle :30.0

Perimeter of Rectangle :22.0

Area of Square :25.0

Perimeter of Square :20.0

\*/