Ex- no: 4 Name: M.Rohith
26-10-2022 3122 21 5001 085

SSN College of Engineering

Department of Computer Science and Engineering

UCS2313 – Object Oriented Programming Lab

II Year CSE - B Section (III Semester)

Academic Year 2022-23

Batch: 2021- 2025

Faculty Incharge :S.Rajalakshmi

Exercise – 4 – Abstract class and Interfaces

Objective:

1. To test the following Inheritance type: multiple inheritance.

2. To test the Polymorphism through Interface / abstract classes by method overriding.

Sample Learning Outcome:

1. Need of interface and it's implementation in Java
2. Need of abstract class and it's implementation in Java
3. Multiple inheritance
4. Accessing the derived class objects through base class/interface reference – Dynamic method dispatch/Dynamic binding

Best Practices:

1. Class Diagram usage
2. Naming convention – for file names, variables
3. Comment usage at proper places
4. Prompt messages during reading input and displaying output
5. Incremental program development
6. Modularity
7. All possible test cases in output

Ex- no: 4 Name: M.Rohith
26-10-2022 3122 21 5001 085

1. **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**

Name: M.Rohith
3122 21 5001 085

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** **+calSalary():float**

Design an Interface Student:

<<Student>>
+getMarks():float [] **+calcGPA():float**

Ex- no: 4
26-10-2022

Name: M.Rohith
3122 21 5001 085

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

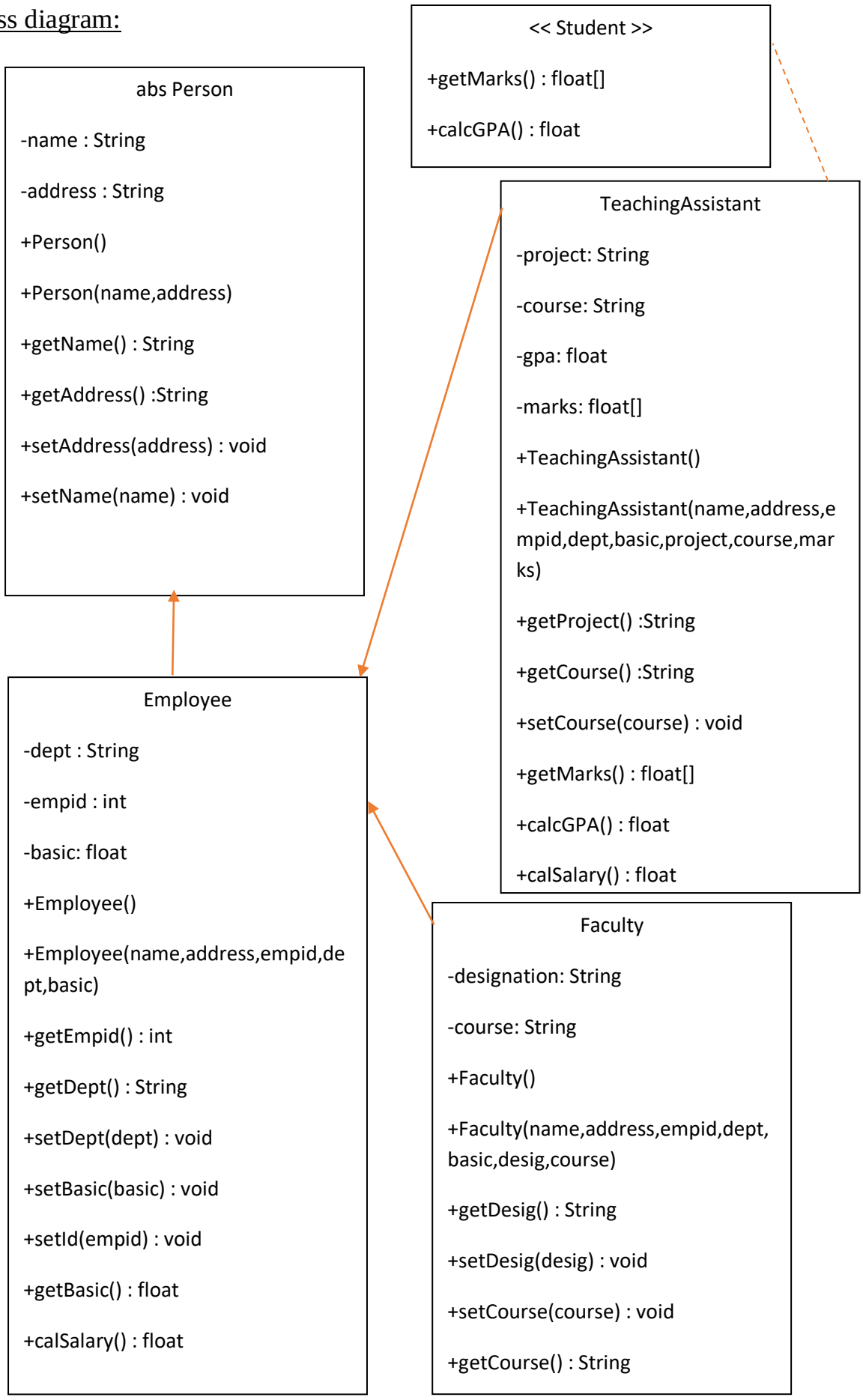
TeachingAssitant
-project:String **-course:String** **-marks:float []**
+TeachingAssitant(name,address,empid,dept,basic,project,course,marks) **+getProject():String** **+getCourse():String** **+setCourse(course):void** **+getMarks():float []** **+calcGPA():float** **+calSalary():float**

Write a TestDriver function to get input for Faculty and TeachingAssistant and display their details. Find the class that can be kept as abstract.

Ex- no: 4
26-10-2022

Name: M.Rohith
3122 21 5001 085

Class diagram:



Ex- no: 4
26-10-2022

Name: M.Rohith
3122 21 5001 085

Program code:

```
import java.util.Scanner;

abstract class Person
{
    //class variables
    private String name,address;

    //default constructor
    Person()
    {
    }

    //argument constructor
    public Person(String name,String address)
    {
        this.name=name;
        this.address=address;
    }

    //getters
    public String getName()
    {
        return this.name;
    }

    //getter
    public String getAddress()
    {
        return this.address;
    }

    //setter
    public void setAddress(String address)
    {
        this.address=address;
    }

    //setter
    public void setName(String name)
    {
        this.name=name;
    }
}

//Derived class
class Employee extends Person
{
```

Name: M.Rohith
3122 21 5001 085

```
    //class variables
    private String dept;
    private int empid;
    private float basic;

    //default constructor
    Employee()
    {
    }

    //argument constructor
    Employee(String name,String address,int empid,String dept,float basic)
    {
        super(name,address);
        this.empid=empid;
        this.dept=dept;
        this.basic=basic;
    }

    //getters
    public int getEmpid()
    {
        return this.empid;
    }

    public String getDept()
    {
        return this.dept;
    }


    //setters
    public void setDept(String dept)
    {
        this.dept=dept;
    }

    public void setBasic(float basic)
    {
        this.basic=basic;
    }

    public void setId(int empid)
    {
        this.empid=empid;
    }

    //getters
```

Name: M.Rohith
3122 21 5001 085

```
    public float getBasic()
    {
        return this.basic;
    }

    public float calSalary()
    {
        return (float) (((0.6 * getBasic()) + (0.1 * getBasic())) -((0.085 *
getBasic()) + (0.08 * getBasic())));
    }
}

class Faculty extends Employee
{
    //class variables
    private String designation,course;

    //default constructor
    Faculty()
    {
    }

    //argument constructor
    Faculty(String name,String address,int empid,String dept,int basic,String
desig,String course)
    {
        super(name,address,empid,dept,basic);
        this.designation=desig;
        this.course=course;
    }

    //getters
    public String getDesig()
    {
        return this.designation;
    }

    //setters
    public void setDesig(String desig)
    {
        this.designation=desig;
    }

    public void setCourse(String course)
    {
        this.course=course;
    }
```

```
    //getters
    public String getCourse()
    {
        return this.course;
    }
}

interface Student
{
    float[] getMarks();
    float calcGPA();
}

class TeachingAssistant extends Employee implements Student
{
    //class variables
    private String project,course;
    private float gpa;
    private float[] marks=new float[5];

    //default constructor
    TeachingAssistant()
    {
    }

    //argument constructor
    TeachingAssistant(String name,String address,int empid,String dept,int
basic,String project,String course,float marks)
    {
        super(name,address,empid,dept,basic);
        this.project=project;
        this.course=course;
        //marks
    }

    //getters
    public String getProject()
    {
        return this.project;
    }

    public String getCourse()
    {
        return this.course;
    }

    //setters
    public void setCourse(String course)
```

```
    {
        this.course=course;
    }

    //getters
    public float[] getMarks()
    {
        int sum=0;
        Scanner scanner=new Scanner(System.in);
        for(int i=0;i<5;i++)
        {
            System.out.println("Enter the mark for subject "+(i+1)+": ");
            this.marks[i]=scanner.nextFloat();
            sum+=this.marks[i];
        }

        System.out.println("Sum = "+sum);
        this.gpa=(float)((sum/500)*10);
        System.out.println("gpa = "+gpa);
        return marks;
    }

    public float calcGPA()
    {
        return this.gpa;
    }

    public float calSalary()
    {
        return (float) (((0.6 * getBasic()) + (0.1 * getBasic())) -((0.085 *
getBasic()) + (0.08 * getBasic())));
    }
}

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

        Faculty faculty=new Faculty();
        setDetails(faculty);
        System.out.println();
        displayDetails(faculty);
        System.out.println();

        TeachingAssistant teachingassistant=new
TeachingAssistant("Vimal","No.5, West Tambaram, OMR Road, Chennai-
```

```
8",211055,"Computer Science and Engineering",50000,"Assistant Professor","OOP
Java",100);
        float[] receivedmarks=teachingassistant.getMarks();
        System.out.println("The marks obtained are: ");
        for(int i=0;i<5;i++)
            System.out.println(receivedmarks[i]+" ");
        System.out.println();
        displayDetails(teachingassistant);
    }

    public static void setDetails(Faculty faculty)
    {
        Scanner scanner=new Scanner(System.in);

        System.out.print("Enter the faculty name: ");
        String name=scanner.nextLine();
        System.out.print("Enter the faculty address: ");
        String address=scanner.nextLine();
        System.out.print("Enter the faculty employee ID: ");
        int empid=scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter the faculty department: ");
        String dept=scanner.nextLine();
        System.out.print("Enter the faculty basic pay: ");
        float basic=scanner.nextFloat();
        scanner.nextLine();
        System.out.print("Enter the faculty designation: ");
        String desig=scanner.nextLine();
        System.out.print("Enter the faculty course: ");
        String course=scanner.nextLine();

        faculty.setName(name);
        faculty.setAddress(address);
        faculty.setId(empid);
        faculty.setDept(dept);
        faculty.setBasic(basic);
        faculty.setDesig(desig);
        faculty.setCourse(course);
    }

    public static void displayDetails(Faculty faculty)
    {
        System.out.println("Faculty name: "+faculty.getName());
        System.out.println("Faculty address: "+faculty.getAddress());
        System.out.println("Faculty employee ID: "+faculty.getEmpid());
        System.out.println("Faculty department: "+faculty.getDept());
        System.out.println("Faculty designation: "+faculty.getDesig());
        System.out.println("Faculty course: "+faculty.getCourse());
```

```
        System.out.println("Faculty salary: "+faculty.calSalary());
    }

    public static void displayDetails(TeachingAssistant teachingAssistant)
    {
        System.out.println("Teaching assistant name:
"+teachingAssistant.getName());
        System.out.println("Teaching assistant address:
"+teachingAssistant.getAddress());
        System.out.println("Teaching assistant employee ID:
"+teachingAssistant.getEmpid());
        System.out.println("Teaching assistant department:
"+teachingAssistant.getDept());
        System.out.println("Teaching assistant project:
"+teachingAssistant.getProject());
        System.out.println("Teaching assistant course:
"+teachingAssistant.getCourse());
        System.out.println("Teaching assistant GPA:
"+teachingAssistant.calcGPA());
        System.out.println("Teaching assistant salary:
"+teachingAssistant.calSalary());
    }
}
```

Name: M.Rohith
3122 21 5001 085

Output:

```
Enter the faculty name: Ravi Varma
Enter the faculty address: No.45, Millerpuram, Madurai-8
Enter the faculty employee ID: 211415
Enter the faculty department: Computer Science and Engineering
Enter the faculty basic pay: 100000
Enter the faculty designation: HOD
Enter the faculty course: IoT

Faculty name: Ravi Varma
Faculty address: No.45, Millerpuram, Madurai-8
Faculty employee ID: 211415
Faculty department: Computer Science and Engineering
Faculty designation: HOD
Faculty course: IoT
Faculty salary: 53500.0

Enter the mark for subject 1:
100
Enter the mark for subject 2:
99
Enter the mark for subject 3:
98
Enter the mark for subject 4:
99
Enter the mark for subject 5:
100
Sum = 496
gpa = 0.0
The marks obtained are:
100.0
99.0
98.0
99.0
100.0
```

```
Teaching assistant name: Vimal
Teaching assistant address: No.5, West Tambaram, OMR Road, Chennai-8
Teaching assistant employee ID: 211055
Teaching assistant department: Computer Science and Engineering
Teaching assistant project: Assistant Professor
Teaching assistant course: OOP Java
Teaching assistant GPA: 0.0
Teaching assistant salary: 26750.0
```

==========

Name: M.Rohith
3122 21 5001 085

II) 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**

Name: M.Rohith
3122 21 5001 085

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**

Name: M.Rohith
3122 21 5001 085

A sub-class Square of class *rectangle* designed as shown below (Square is one where the length and width of rectangle are same):

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 implements 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.

Ex- no: 4
26-10-2022

Name: M.Rohith
3122 21 5001 085

Class diagram:

Ex- no: 4
26-10-2022

Name: M.Rohith
3122 21 5001 085

Program code:

```
import java.util.Scanner;

abstract class Shape        //abstract class
{
    protected String color="red";   //protected class members

    public Shape()  //default constructor
    {

    }

    public Shape(String color)  //single argument constructor
    {
        this.color=color;
    }

    public String getColor()    //getter
    {
        return this.color;
    }

    public void setColor(String color)      //setter
    {
        this.color=color;
    }

    abstract float getArea();           //abstract methods
    abstract float getPerimeter();
}

interface calculations      //interface
{
    float getArea();
    float getPerimeter();
}

class Circle extends Shape implements calculations
{
    protected float radius=1;       //protected class variable

    public Circle()         //default constructor
    {

    }

    public Circle(float radius)         //single argument constructor
```

```
    {
        this.radius=radius;
    }

    public Circle(float radius,String color)          //multiple argument
constructor
    {
        super(color);
        this.radius=radius;
    }

    public float getRadius()        //getter
    {
        return this.radius;
    }

    public void setRadius(float radius)         //setter
    {
        this.radius=radius;
    }

    public float getArea()          //method overriding
    {
        return (float)(Math.PI*radius*radius);
    }

    public float getPerimeter()         //method overriding
    {
        return (float) (2*Math.PI*radius);
    }
}

class Rectangle extends Shape implements calculations
{
    protected float width=1;        //protected class members
    protected float length=1;

    public Rectangle()      //default constructor
    {

    }

    public Rectangle(float width,float length)      //multiple argument
constructors
    {
        this.length=length;
        this.width=width;
    }
```

Name: M.Rohith
3122 21 5001 085

```
    public Rectangle(float width,float length,String color)
    {
        super(color);
        this.length=length;
        this.width=width;
    }

    public float getWidth()         //getter
    {
        return this.width;
    }

    public void setWidth(float width)           //setter
    {
        this.width=width;
    }

    public float getLength()            //getter
    {
        return this.length;
    }

    public void setLength(float length)     //setter
    {
        this.length=length;
    }

    public float getArea()              //method overriding
    {
        return (float)(length*width);
    }

    public float getPerimeter()         //method overriding
    {
        return (float)(2*(length+width));
    }
}

class Square extends Rectangle implements calculations
{
    public Square()         //default constructor
    {

    }

    public Square(float side)           //single argument constructor
    {
```

```
        super(side,side);
    }

    public Square(float side,String color)          //multiple argument
constructor
    {
        super(side,side,color);
    }

    public float getSide()      //getter
    {
        return this.length;
    }

    public void setSide(float side)         //setter
    {
        super.setWidth(side);
        super.setLength(side);
    }

    public float getArea()          //method overriding
    {
        return (float)(super.length*super.width);
    }

    public float getPerimeter()         //method overriding
    {
        return (float)(4*super.length);
    }
}

class TestInterface
{
    public static void  main(String[] a)
    {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of shapes: ");
        int n = scanner.nextInt();
        Shape[] shapes = new Shape[n];
        for (int i = 0; i < n; i++)
        {
            System.out.print("Shapes :\n1.Circle \n2.Rectangle \n3.Square \n
Enter the shape number: ");
            int shapeName = scanner.nextInt();
            System.out.println();
            switch(shapeName){
            case 1:
                System.out.print("Enter the radius: ");
```

```
                float radius = scanner.nextFloat();
                System.out.print("\nEnter the color: ");
                String colorCir = scanner.next();
                shapes[i] = new Circle(radius, colorCir);
                System.out.println("\nThe area and perimeter of the " +
shapes[i].getColor() + " circle formed is: " + shapes[i].getArea() + " and " +
shapes[i].getPerimeter());
                break;
            case 2:
                System.out.print("Enter the width: ");
                float width = scanner.nextFloat();
                System.out.print("\nEnter the length: ");
                float length = scanner.nextFloat();
                System.out.print("\nEnter the color: ");
                String colorRect = scanner.next();
                shapes[i] = new  Rectangle(width, length, colorRect);
                System.out.println("\nThe area and perimeter of the " +
shapes[i].getColor() + " rectangle formed is: " + shapes[i].getArea() + " and
" + shapes[i].getPerimeter());
                break;
            case 3:
                System.out.print("Enter the side: ");
                float side = scanner.nextFloat();
                System.out.print("\nEnter the color: ");
                String colorSq = scanner.next();
                shapes[i] = new Square(side, colorSq);
                System.out.println("\nThe area and perimeter of the " +
shapes[i].getColor() + " square formed is: " + shapes[i].getArea() + " and " +
shapes[i].getPerimeter());
                break;
            default:
                System.out.println("Invalid shape...");
            }
        }
        scanner.close();
    }
}
```

Output:

```
Enter the number of shapes: 3
Shapes :
1.Circle
2.Rectangle
3.Square
 Enter the shape number:
1

Enter the radius: 2

Enter the color: orange

The area and perimeter of the orange circle formed is: 12.566371 and 12.566371
Shapes :
1.Circle
2.Rectangle
3.Square
 Enter the shape number: 2

Enter the width: 10

Enter the length: 15

Enter the color: white

The area and perimeter of the white rectangle formed is: 150.0 and 50.0
Shapes :
1.Circle
2.Rectangle
3.Square
 Enter the shape number: 3

Enter the side: 12

Enter the color: green

The area and perimeter of the green square formed is: 144.0 and 48.0
```

Learning outcomes:

Java program to test the multiple inheritance and the polymorphism through abstract classes and interfaces by method overriding has been implemented and executed successfully along with their class diagrams for each given exercise.

############$$$$$$$$$$$$$$$$$####################$$$$$$$$$$$$$$$###