

| Name | Samya Sunibir Das |
| --- | --- |
| ID | 1911563042 |
| Course Code | CSE215 |
| Section | 19 |
| Assignment No. | 03 |
| Semester | Fall 2022 (223) |

**Answer to Question 1: (Please scroll down to page 8 for question 2)**

(Compiled in VS Code), in sequence in this report: TestGeometricObject, GeometricObject.java, Triangle.java

Rectangle.java, Circle.java not added here since no changes were made.

**TestGeometricObject.java**

public class TestGeometricObject {

*/\*\* Main method \*/*

public static void main(String[] *args*) {

*//create the geometric objects needed*

GeometricObject geoObject1 = new Circle(5);

GeometricObject geoObject2 = new Rectangle(5, 3);

GeometricObject geoObject3 = new Triangle(5, 3, 4);

System.out.println("Do the three objects have the same area? " + equalArea(geoObject1, geoObject2, geoObject3));

*// Display circle*

showGeometricObject(geoObject1);

*// Display rectangle*

showGeometricObject(geoObject2);

*//Display triangle*

showGeometricObject(geoObject3);

}

*/\*\* A method for comparing the areas of three geometric objects \*/*

public static boolean equalArea(GeometricObject *o1*, GeometricObject *o2*, GeometricObject *o3*) {

return *o1*.getArea() == *o2*.getArea() && *o1*.getArea() == *o3*.getArea(); *//returns true if equal*

}

*//show geometric object by accessing their area, perimeter*

public static void showGeometricObject(GeometricObject *object*) {

System.out.println();

System.out.println("The area is " + *object*.getArea());

System.out.println("The perimeter is " + *object*.getPerimeter());

}

}

**GeometricObject.java**

public abstract class GeometricObject {

private String color = "white";

private boolean filled;

private java.util.Date dateCreated;

*/\*\* Construct a default geometric object \*/*

protected GeometricObject() {

dateCreated = new java.util.Date();

}

*/\*\* Construct a geometric object with color and filled value \*/*

protected GeometricObject(String *color*, boolean *filled*) {

dateCreated = new java.util.Date();

this.color = *color*;

this.filled = *filled*;

}

*/\*\* Return color \*/*

public String getColor() {

return color;

}

*/\*\* Set a new color \*/*

public void setColor(String *color*) {

this.color = *color*;

}

*/\*\* Return filled. Since filled is boolean,*

*\* the get method is named isFilled \*/*

public boolean isFilled() {

return filled;

}

*/\*\* Set a new filled \*/*

public void setFilled(boolean *filled*) {

this.filled = *filled*;

}

*/\*\* Get dateCreated \*/*

public java.util.Date getDateCreated() {

return dateCreated;

}

*/\*\* Return a string representation of this object \*/*

public String toString() {

return "created on " + dateCreated + "\ncolor: " + color +

" and filled: " + filled;

}

*/\*\* Abstract method getArea \*/*

public abstract double getArea();

*/\*\* Abstract method getPerimeter \*/*

public abstract double getPerimeter();

}

**Triangle.java**

public class Triangle extends GeometricObject {

private double s1 = 1.0;

private double s2 = 1.0;

private double s3 = 1.0;

public Triangle() {

}

public Triangle(double *s1*, double *s2*, double *s3*) {

this.s1 = *s1*;

this.s2 = *s2*;

this.s3 = *s3*;

}

*/\*\* Return sides \*/*

public double getS1() {

return s1;

}

public double getS2() {

return s2;

}

public double getS3() {

return s3;

}

public void setS1(double *s1*) {

this.s1 = *s1*;

}

public void setS2(double *s2*) {

this.s2 = *s2*;

}

public void setS3(double *s3*) {

this.s3 = *s3*;

}

*//calculate perimeter*

public double getPerimeter() {

return s1 + s2 + s3;

}

*//calculate area*

public double getArea() {

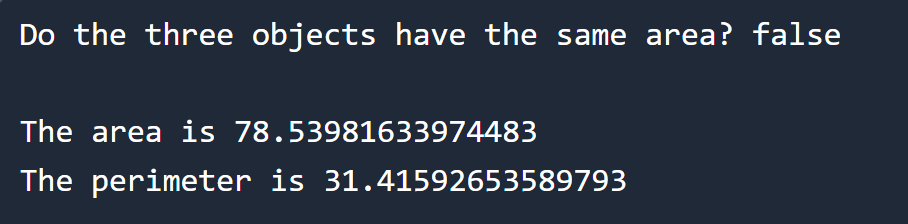
double x = getPerimeter() / 2;

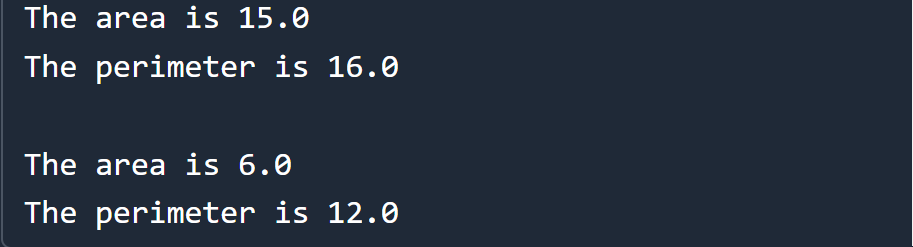
return Math.sqrt(x \* ((x - s1) \* (x - s2) \* (x - s3)));

}

}

Output after compiling TestGeometricObject.java (as it has main method)





**Answer to Question 2**

The four characteristics of OOP are: 1. Encapsulation, 2. Abstraction, 3. Inheritance, 4. Polymorphism.

1. **Encapsulation** in code: In Java, we can create a fully encapsulated class by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it. There were numerous examples of encapsulation used in all of the 5 classes. For example, in Triangle class, we saw the sides being declared as private variables early on, then used setter and getter to get the data.

Declaring the private variables

private double s1 = 1.0;

private double s2 = 1.0;

private double s3 = 1.0;

Sample getter setters for a side

public double getS1() {

return s1;}

public void setS1(double *s1*) {

this.s1 = *s1*;

}

1. **Abstraction** in code: In Java, Abstraction is a process of hiding the implementation details and showing only functionality to the user, abstraction can be done for classes and methods both. In GeometricObject.java, the class was created as abstract then two abstract methods were defined with the implementations hidden.

*/\*\* Abstract method getArea \*/*

public abstract double getArea();

*/\*\* Abstract method getPerimeter \*/*

public abstract double getPerimeter();

1. **Inheritance** in code: In Java, inheritance is a mechanism in which one object acquires all the properties and behaviors of a parent object. There were numerous examples of inheritance all throughout the code, Triangle, Rectangle, Circle all of the three classes inherited all of the available attributes and methods from GeometricObject

public class Triangle extends GeometricObject {

1. **Polymorphism** in code: in Java, Polymorphism is an idea by which we can perform a single action in different ways. All of the Triangle, Circle, Rectangle classes had getArea(), getPerimeter() methods with the same return type and same method name. Then later they were compared in main program to check whether the area is not equal for all three or not, calling methods with same name but with different functionality from separate classes.

Triangle getArea():

public double getArea() {

double x = getPerimeter() / 2;

return Math.sqrt(x \* ((x - s1) \* (x - s2) \* (x - s3)));

}

Circle getArea():

public double getArea() {

return radius \* radius \* Math.PI;

}

Rectangle getArea():

public double getArea() {

return width \* height;

}

Comparison in main, for three different objects calling three separate methods with same name but different formulae from three different classes

*/\*\* A method for comparing the areas of three geometric objects \*/*

public static boolean equalArea(GeometricObject *o1*, GeometricObject *o2*, GeometricObject *o3*) {

return *o1*.getArea() == *o2*.getArea() && *o1*.getArea() == *o3*.getArea(); *//returns true if equal*

}

*The End*