**ADVANCED JAVA PROGRAMMING – ASSIGNMENT 1**

This document contains the following

* Assignment and student details
* Completion Plan
* Core concepts used in the project
* Reflection on the concepts learnt
* Links to the GitHub repository
* Source Code
* Instructions to run the code
* Output Screenshots
* Conclusion

# Assignment and student details:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Assignment #1**

**Course:** *Advanced Java Programming - ITE-5215-0NB - Semester 2*

**Last Name:** *Pillarisetty*

**First Name:** *Tejeswi Devi Priya Pillarisetty*

**ID:** *N01654962*

**Section:** *0NB*

*This assignment represents my own work in accordance with Humber Academic Policy.*

***Signature:*** *Tejeswi Devi Priya Pillarisetty*

**Date:** *5th June 2024*

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Assignment details:**

**Description:**

The following assignment lets you practice basic java coding techniques, creating classes, methods, using arrays, inheritance, polymorphism, Exceptional Handling.

**Task – 1:**

Design an abstract class named GeometricObject that contains:

• A private String data field named color (default value “white”)

• A private Boolean data field named filled.

• A no-arg constructor.

• A protected overloaded constructor GeometricObject(String color, boolean filled).

• The accessor and mutator methods for color and filled.

• An abstract method getArea()

• An abstract method getPerimeter()

Design another class named Triangle that contains:

❖ Triangle class must inherit the GeometricObject class.

❖ Three double data fields named side1, side2, and side3 with default values 1.0 to denote three sides of the triangle.

❖ A no-arg constructor that creates a default triangle.

❖ A constructor that creates a triangle with the specified side1, side2, and side3.

❖ The accessor methods for all three data fields.

❖ A method named getArea() that returns the area of this triangle.

❖ A method named getPerimeter() that returns the perimeter of this triangle.

❖ A method named toString() that returns a string description for the triangle.

❖ The formulate to compute the area of the triangle is as follows:

****

➢ Write a test program that prompts the user to enter three sides of the triangle, a color, and a Boolean value to indicate whether the triangle is filled.

➢ The program should create a Triangle object with the given sides and set the color and filled properties using the input.

➢ The program should display the area, perimeter, color, and true or false to indicate whether it is filled or not.

**Task – 2**

Triangle class in Task – 1 (above), defines a triangle with three sides. In a triangle, the sum of any two sides is greater than the other side. The Triangle class must obey/ follow to this rule.

Design a TriangleException class, and modify the constructor of the Triangle class to throw a TriangleException object if triangle is created with sides that violates the rule, asfollow:

/\*\*Construct a triangle with the specified sides \*/

public Triangle(double s1, double s2, double s3) throws TriangleException{

//Implementation

}

Hint: For not having any confusion you can copy your Triangle class from the Task – 1 and rename it in Task – 2.

Write a test program that will create two Triangle class objects one with legal sides and the other will Illegal side.

Note: Students are encouraged to design their own outputs for the both the tasks.

**Assignment Header**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Assignment #

Course:<subject type> - Semester

Last Name:<student last name>

First Name:<student first name>

ID:<student ID>

Section:<section name>

This assignment represents my own work in accordance with Humber Academic Policy.

Signature

Date:<submission date>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Code Submission Criteria:**

Please note that you should have:

• Appropriate indentation.

• Proper file structure

• Follow java naming convention

• Document all the classes properly

• Do Not have any debug/ useless code and/ or files in the assignment

• Do not have everything in the main method.

• Have a separate TestClass with the main method in it.

• Check your inputs if the user is not entering garbage inputs.

• Use exceptional handling or other methods to let the user know if the inputs are incorrect.

**Deliverables and Important Notes:**

All these deliverables are supposed to be uploaded on the blackboard once done.

• You are supposed to create video/ record voice/ detailed document of your running solution. (50%)

o Screen Video captured file should state your last name and id, likeZhian\_123456.mp4 (or whatever the extension of the file is)

Also Submit a separate word file with the screen shots of your program’s output, state your last name and id, like Zhian\_123456.docs

o Detailed document should include screen shots of your output, have your name and id on the top of the file and save the file with your last name and id, likeZhian\_123456.docx (or whatever the extension of the file is)

• A word/ text file which will reflect on learning of your concepts in this Assignment Also include the instructions on how to run your code. (50%)

o Should state your Full name and Id on the top of the file and save the file with your last name and id, like Zhian\_123456.txt

• Remember that you are encouraged to talk to each other, to the instructor, or to anyone else about any of the assignments, but the final solution may not be copied from any source.

# Completion plan:

This assignment completion plan is as follows:

**Task 1:** Done by Shubham Sangwan & Dewrat Raval

**Task 2:** Done by Tejeswi Devi Priya Pillarisetty

# Core concepts used in the project:

* Abstract Classes and Methods
* Inheritance
* Exception Handling
* Encapsulation
* Polymorphism

# Reflection on the concepts learnt:

1. **Abstract Classes and Methods**

Abstract classes are like templates for other classes to inherit from. They contain the abstract methods. This allows code reusability which allowed me to create a flexible class hierarchy.

In this assignment, the ‘GeometricObject’ class is an abstract class containing the abstract methods ‘getArea()’ and ‘getPerimeter()’.

1. **Inheritance**

Inheritance allows a subclass to inherit attributes and method from the superclass. Subclasses can extend and override superclass methods to create methods according to the specific needs of the subclass.

In this assignment, the ‘Triangle’ class inherits from the ‘GeometricObject’ class, inheriting its color and filled attributes, also implementing its abstract methods.

1. **Exception Handling**

Exception handling allows programs to gracefully handle errors or exceptional conditions that may occur during runtime. It provides mechanisms to detect and handle errors which in turn prevents the program crashes.

In this assignment, exception handling is used in ‘TriangleException’ class to validate triangle sides according to the triangle inequality theorem. Also, I have created another exception where any or all of the sides is less than or equal to 0.

1. **Encapsulation**

Encapsulation combines the data and methods into a single class keeping the internal state of an object hidden and exposing only needed functionalities through public methods.

In this assignment, encapsulation concept is used in ‘GeometricObject’ class where color and filled are private data fields and are modified through public methods like ‘getColor()’ and ‘setColor()’.

1. **Polymorphism**

Polymorphism allows different objects to be treated as if they are of the same type. It enables methods to be used on objects without needing to know their specific types in advance, making code more flexible and adaptable.

In this assignment, polymorphism is demonstrated by the fact that the ‘getArea()’ and ‘getPerimeter()’ methods are implemented differently in the ‘Triangle’ subclass, but they can be used in the same way as methods in the ‘GeometricObject’ superclass. This means that we can call these methods on any ‘GeometricObject’ object, whether it's ‘Triangle’ or another subclass.

# Links to the GitHub repository:

# <https://github.com/Priya-2705/Assignment1.git>

# Project Hierarchy

A screen shot of a computer program

Description automatically generated

# Source Code and Output Screenshots:

The following are the steps followed in completing this assignment:

**Creating a java project in Eclipse**

* Open Eclipse IDE.
* Go to File 🡪 New 🡪 Others 🡪 Java Project.
* Give the project name ‘Assignment1’ and click Finish.
* Create a package named ‘com.humber.java.assignment1 in src folder of the project.
* Then, create the following class files in it.

**GeometricObject.java**

package com.humber.java.assignment1;

public abstract class GeometricObject {

private String color = "white";

private boolean filled;

public GeometricObject() {

this.color = "white";

this.filled = false;

}

protected GeometricObject(String color, boolean filled) {

this.color = color;

this.filled = filled;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

public boolean isFilled() {

return filled;

}

public void setFilled(boolean filled) {

this.filled = filled;

}

public abstract double getArea();

public abstract double getPerimeter();

}

**Triangle.java**

package com.humber.java.assignment1;

public class Triangle extends GeometricObject {

private double side1 = 1.0;

private double side2 = 1.0;

private double side3 = 1.0;

public Triangle() {

super();

}

public Triangle(double side1, double side2, double side3) {

super();

this.side1 = side1;

this.side2 = side2;

this.side3 = side3;

}

public double getSide1() {

return side1;

}

public double getSide2() {

return side2;

}

public double getSide3() {

return side3;

}

*@Override*

public double getArea() {

double s = (side1 + side2 + side3) / 2.0;

return Math.*sqrt*(s \* (s - side1) \* (s - side2) \* (s - side3));

}

*@Override*

public double getPerimeter() {

return side1 + side2 + side3;

}

*@Override*

public String toString() {

return "Triangle: side1 = " + side1 + " side2 = " + side2 + " side3 = " + side3;

}

}

**TestTriangle.java**

package com.humber.java.assignment1;

import java.util.Scanner;

public class TestTriangle{

public static void main(String[] args) {

Scanner sc = new Scanner(System.***in***);

System.***out***.println("Enter side1 of triangle: ");

double side1 = sc.nextDouble();

System.***out***.println("Enter side2 of triangle: ");

double side2 = sc.nextDouble();

System.***out***.println("Enter side3 of triangle: ");

double side3 = sc.nextDouble();

System.***out***.println("Enter the color of the triangle: ");

String color = sc.next();

System.***out***.println("Is the triangle filled? True/False");

boolean isFilled = sc.nextBoolean();

Triangle triangle = new Triangle(side1, side2, side3);

triangle.setColor(color);

triangle.setFilled(isFilled);

System.***out***.println(color + " triangle is created");

System.***out***.println("Specifications of the triangle are as follows: ");

System.***out***.println("Area of the Triangle: " + triangle.getArea());

System.***out***.println("Perimeter of the Triangle: " + triangle.getPerimeter());

System.***out***.println("Color of the Triangle: " + triangle.getColor());

System.***out***.println("Is the triangle filled? " + triangle.isFilled());

sc.close();

}

}

**TriangleException.java**

package com.humber.java.assignment1;

public class TriangleException extends Exception {

public TriangleException(String message) {

super(message);

}

}

**TriangleWithException.java**

package com.humber.java.assignment1;

public class TriangleWithException extends GeometricObject {

private double side1 = 1.0;

private double side2 = 1.0;

private double side3 = 1.0;

public TriangleWithException() {

super();

}

public TriangleWithException(double side1, double side2, double side3) throws TriangleException {

super();

if (side1 <= 0 || side2 <= 0 || side3 <= 0) {

throw new TriangleException("Any side should not be 0 or less than zero");

}

else if (side1 + side2 <= side3 || side2 + side3 <= side1 || side3 + side1 <= side2) {

throw new TriangleException("The sum of any two sides must be greater than the other side.");

}

this.side1 = side1;

this.side2 = side2;

this.side3 = side3;

}

public double getSide1() {

return side1;

}

public double getSide2() {

return side2;

}

public double getSide3() {

return side3;

}

*@Override*

public double getArea() {

double s = (side1 + side2 + side3) / 2.0;

return Math.*sqrt*(s \* (s - side1) \* (s - side2) \* (s - side3));

}

*@Override*

public double getPerimeter() {

return side1 + side2 + side3;

}

*@Override*

public String toString() {

return "Triangle: side1 = " + side1 + " side2 = " + side2 + " side3 = " + side3;

}

}

**TestTriangleWithException.java**

package com.humber.java.assignment1;

public class TestTriangleWithException {

public static void main(String[] args) {

// Test case with valid sides

System.***out***.println("Creating a triangle with valid sides: (3, 4, 5)");

try {

TriangleWithException validTriangle = new TriangleWithException(3, 4, 5);

validTriangle.setColor("Blue");

validTriangle.setFilled(true);

System.***out***.println("Triangle is created");

System.***out***.println("Specifications of the triangle are as follows: ");

System.***out***.println("Area of the Triangle: " + validTriangle.getArea());

System.***out***.println("Perimeter of the Triangle: " + validTriangle.getPerimeter());

System.***out***.println("Color of the Triangle: " + validTriangle.getColor());

System.***out***.println("Is the triangle filled? " + validTriangle.isFilled());

} catch (TriangleException e) {

System.***out***.println("Error creating triangle: " + e.getMessage());

}

// Test case with invalid sides

System.***out***.println("\nCreating a triangle with invalid sides: (1, 2, 3)");

try {

TriangleWithException invalidTriangle = new TriangleWithException(1, 2, 3);

invalidTriangle.setColor("Green");

invalidTriangle.setFilled(false);

System.***out***.println("Triangle is created");

System.***out***.println("Specifications of the triangle are as follows: ");

System.***out***.println("Area of the Triangle: " + invalidTriangle.getArea());

System.***out***.println("Perimeter of the Triangle: " + invalidTriangle.getPerimeter());

System.***out***.println("Color of the Triangle: " + invalidTriangle.getColor());

System.***out***.println("Is the triangle filled? " + invalidTriangle.isFilled());

} catch (TriangleException e) {

System.***out***.println("Error creating triangle: " + e.getMessage());

}

// Test case with invalid sides 2

System.***out***.println("\nCreating another triangle with invalid sides: (1, 0, -3)");

try {

TriangleWithException invalidTriangle = new TriangleWithException(1, 0, -3);

invalidTriangle.setColor("Orange");

invalidTriangle.setFilled(true);

System.***out***.println("Triangle is created");

System.***out***.println("Specifications of the triangle are as follows: ");

System.***out***.println("Area of the Triangle: " + invalidTriangle.getArea());

System.***out***.println("Perimeter of the Triangle: " + invalidTriangle.getPerimeter());

System.***out***.println("Color of the Triangle: " + invalidTriangle.getColor());

System.***out***.println("Is the triangle filled? " + invalidTriangle.isFilled());

} catch (TriangleException e) {

System.***out***.println("Error creating triangle: " + e.getMessage());

}

}

}

# Instructions on how to run the code:

* Download the Assignment1.zip file and extract it.
* Open it in Eclipse IDE.
* Right click on Assignment1 java project and hover on Run as and then, click on Java application from the sub menu.
* Give the inputs in console to get the expected output.

# Output Screenshots:

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

# Conclusion:

In this assignment, I learned fundamental concepts of Java programming, including abstract classes, inheritance, exception handling, and documentation. By implementing a triangle class hierarchy and incorporating exception handling for invalid triangle sides, I gained practical experience in applying these concepts to real-world scenarios.