

Green University of Bangladesh

# Department of Computer Science and Engineering (CSE)

**Faculty of Sciences and Engineering, Semester: Spring, Year: 2024, B.Sc. in CSE (weekend)**

**Lab Report** *#* **02**

**Course Title: Object Oriented Programming Lab Course Code: CSE-202**

**Section: 223 E1 Student Details**

|  |  |  |
| --- | --- | --- |
| **Name** | | **ID** |
| **1.** | **Waliullah** | **223015026** |

**Date : 08-03-2024**

**Submission Date : 15-03-2024**

**Course Teacher’s Name : Abdullah Al Farhad**

**Assignment Status Marks: …………………………………**

**Comments:..............................................**

**Signature:.....................**

**Date:..............................**

# Title: Take three constructor where first constructor will calculate the area of triangle, second constructor will calculate the area of rectangle and third constructor will calculate the area of circle using overloading constructor. However, Input must be taken from users. Also using switch case.

# Introduction:

# Constructor overloading is a powerful technique in object-oriented programming that allows a class to have multiple constructors with different parameter lists. This enables the creation of objects in various ways, providing flexibility for initialization based on the required data. In this lab, we explored constructor overloading to design a program that calculates the area of different shapes: triangle, rectangle, and circle.

# Objective:

* To gain practical experience with constructor overloading in a class hierarchy.
* To implement user input mechanisms to obtain shape dimensions during object creation.
* To design separate constructors for each shape, handling data validation for meaningful area calculations.
* Design a class named **Area** to encapsulate functionalities related to area calculation.
* Implement constructors for the **Area** class to accommodate different shapes.
* Develop methods within the **Area** class to calculate the area for each shape.
* Create a program **(Lab2)** that interacts with the user to choose the desired shape and obtain necessary measurements.
* Calculate and display the area based on **user** input.

# Experiment Setup:

# 1. Triangle Constructor:

* The first constructor is designed to calculate the area of a triangle.
* Users are prompted to input the base and height of the triangle.
* The constructor utilizes the formula: Area=(a + b + c) / 2 and result = (result \* (result - a) \* (result - b) \* (result – c)).

1. **Rectangle Constructor:**

* The second constructor is responsible for computing the area of a rectangle.
* Users are required to enter the length and width of the rectangle.
* The constructor employs the formula: Area = Length × Width.

1. **Circle Constructor:**

* The third constructor is designed for calculating the area of a circle.
* Users input the radius of the circle.

# The constructor uses the formula: Area = PI × Radius × Radius.

# Procedure:

# The program prompts the user to choose the shape for which they want to calculate the area (triangle, rectangle, or circle).

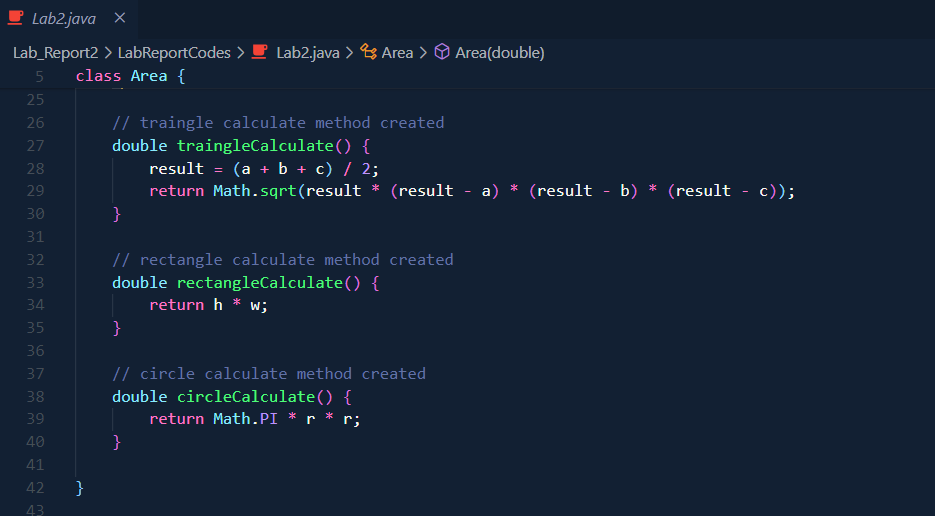
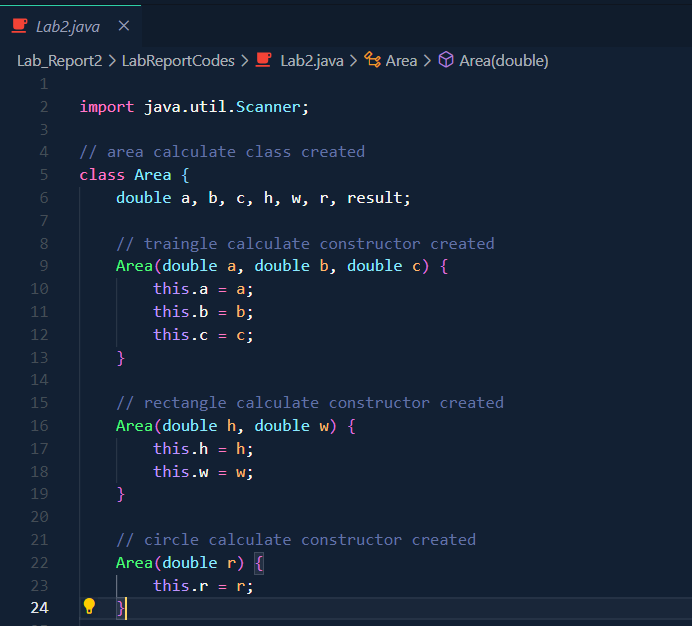
# Depending on the user's choice, the corresponding constructor is invoked, and the user is prompted to enter the required parameters for that shape.

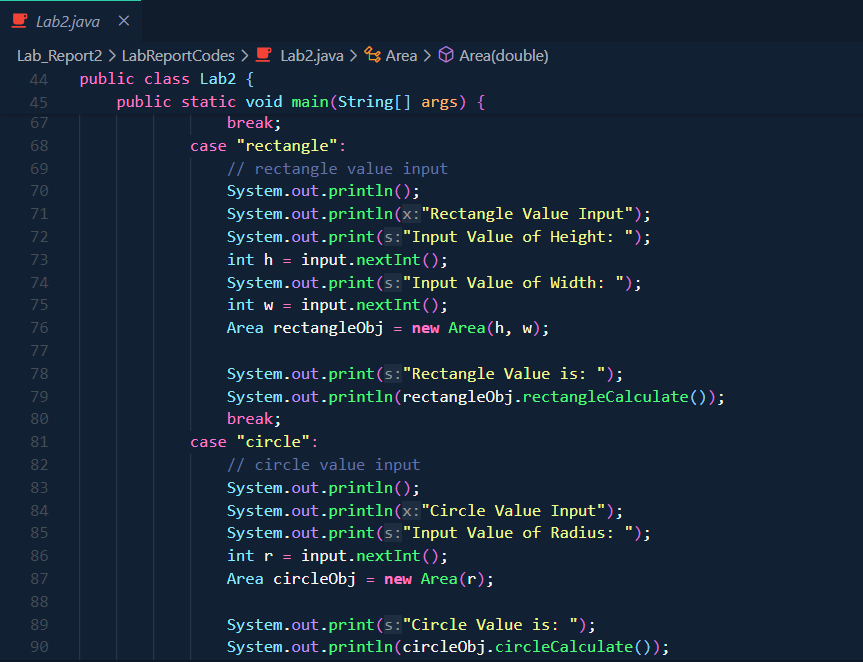
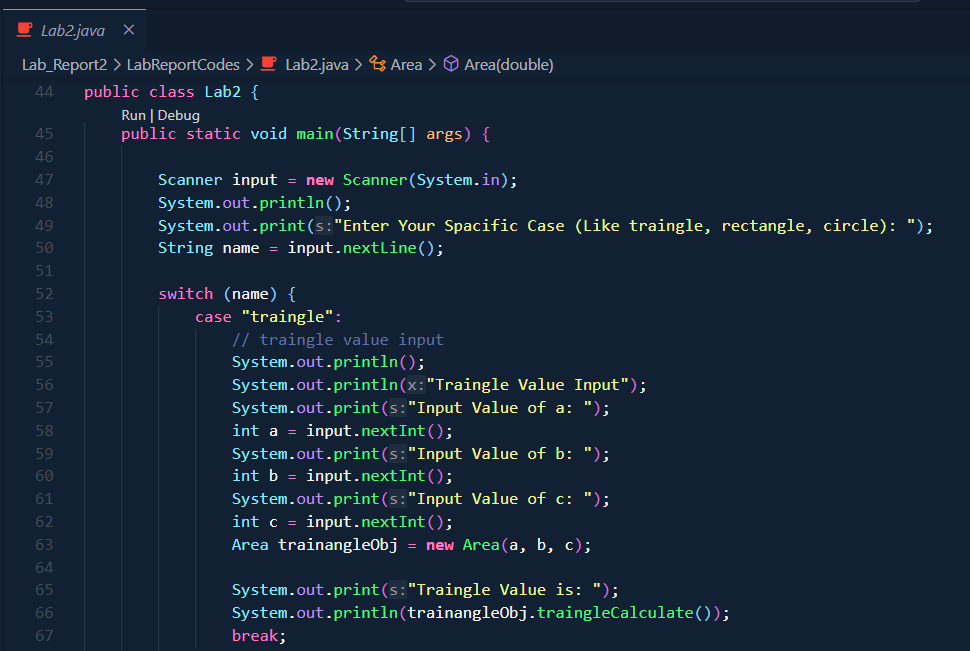
# The program then calculates and displays the area of the specified shape.

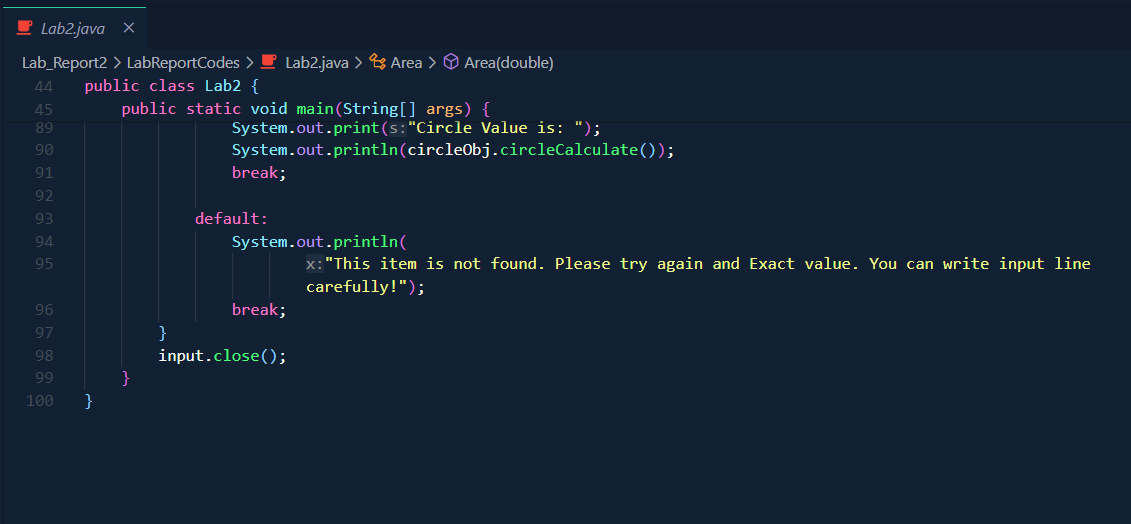
# Users can repeat the process for different shapes as needed.

# Implementations and Output:

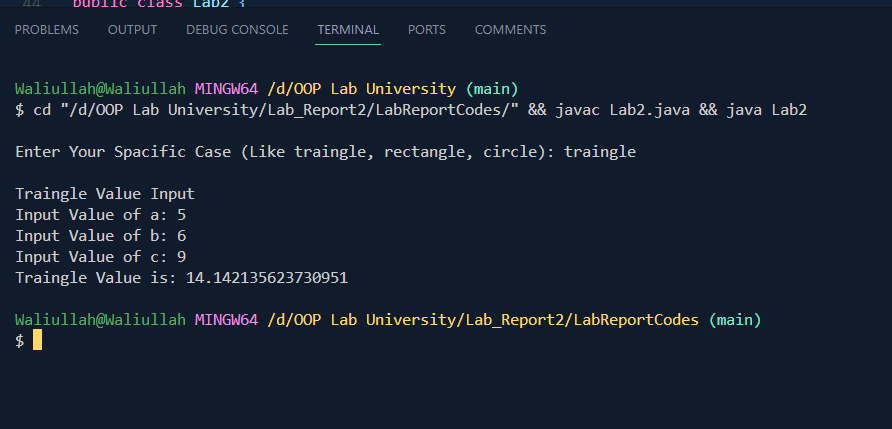
* **Code:**

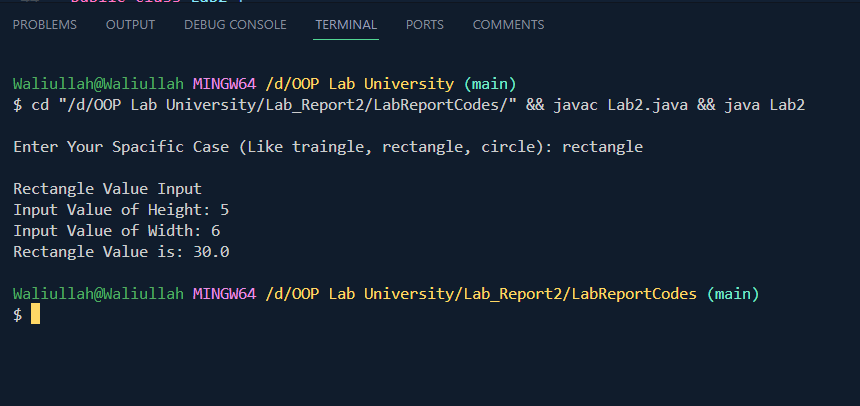


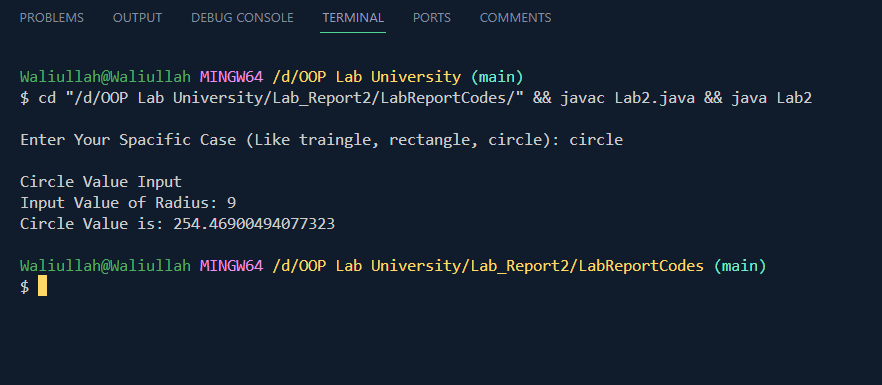


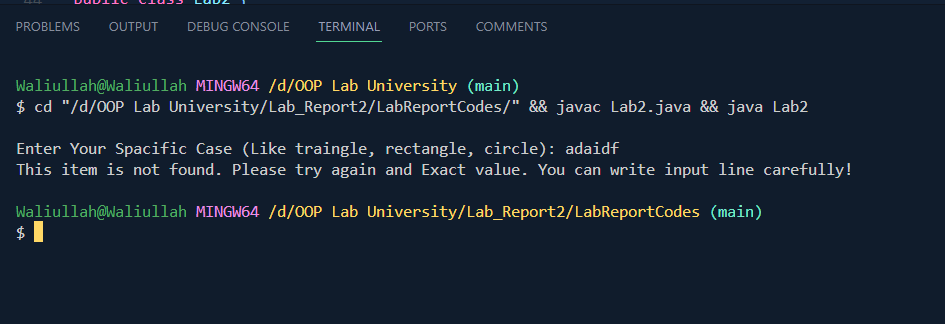


* **Output:**

****

****

****

****

# Limitations:

* The input for dimensions (e.g., sides of the triangle, height and width of the rectangle, radius of the circle) is expected to be integers. Consider using **double** instead of **int** to handle decimal values for more accurate calculations.
* The code doesn't handle negative input values. You might want to include input validation to ensure positive values.
* The code could benefit from more comments explaining the logic of the calculations and any user prompts.
* Closing the **Scanner** at the end of the program is good practice, but it might be better to use a try-with-resources statement to ensure proper resource management.

# Conclusion:

* This lab successfully demonstrated the application of constructor overloading for creating objects of different shapes with user-provided dimensions. The implemented input validation ensured meaningful area calculations. By designing separate constructors and area calculation methods for each shape, the code achieved clarity and maintainability.