**6.1** Addition and Multiplication using **Inheritance** by creating a simple calculator application that performs addition and multiplication operations.

import java.util.Scanner;

class **Calculation**

{

protected int num1, num2;

// Constructor to initialize the two integers

public **Calculation**(int num1, int num2)

{

this.num1 = num1;

this.num2 = num2;

}

// Method to perform addition and return the result

public int **addition**()

{

return num1 + num2;

}

}

class My\_Calculation **extends** Calculation

{

// Constructor that calls the parent constructor

public **My\_Calculation**(int num1, int num2)

{

super(num1, num2);

}

// Method to perform multiplication and return the result

public int **multiplication**()

{

return num1 \* num2;

}

}

public class **MainCalculation**

{

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

{

Scanner scanner = new Scanner(System.in);

System.out.println("Enter Two Integer Numbers");

int num1 = scanner.nextInt();

int num2 = scanner.nextInt();

My\_Calculation myCalculation = new My\_Calculation(num1, num2);

int sum = myCalculation.addition();

int product = myCalculation.multiplication();

System.out.println("Sum of "+num1+" & "+num2+":"+sum);

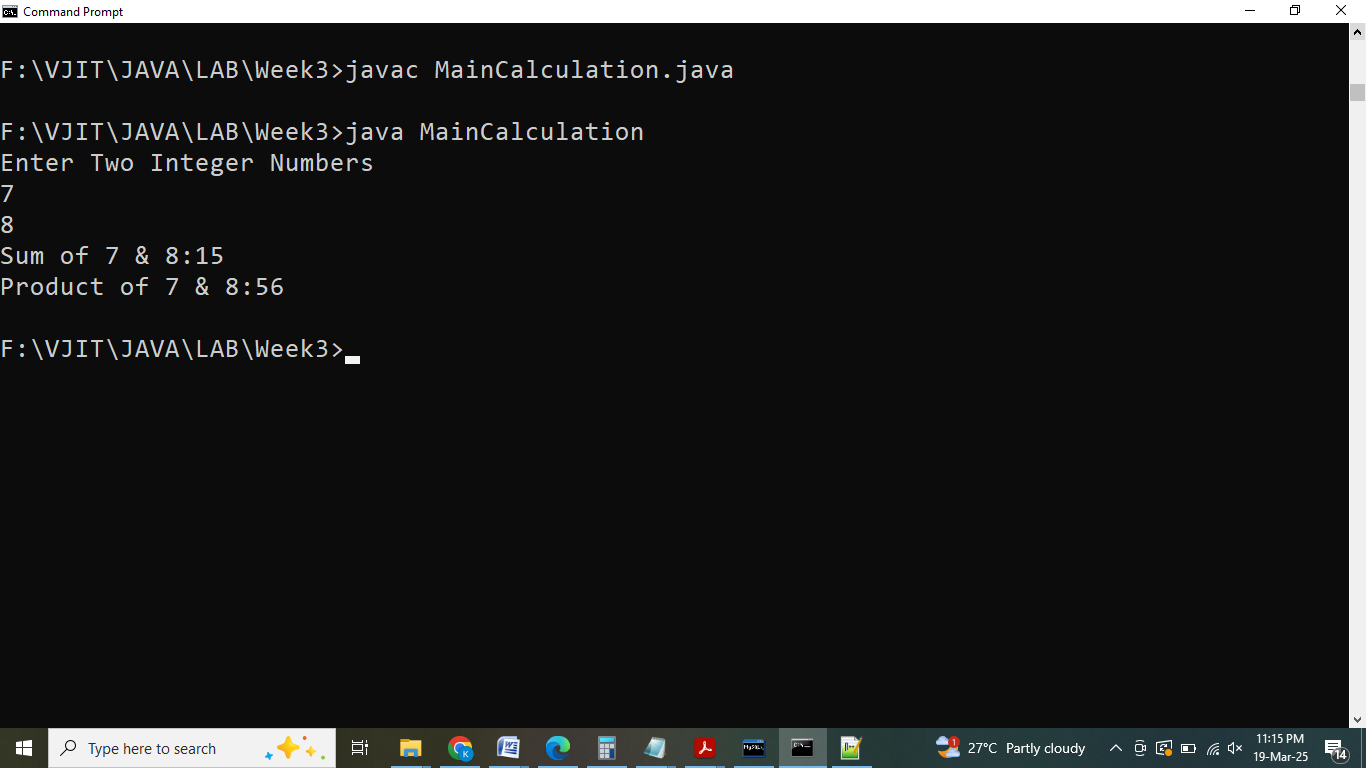
System.out.println("Product of "+num1+" & "+num2+":"+product);

scanner.close();

}

}

**Output:**



**6.2** Write a Java program to model a **multi-level inheritance** scenario involving classes Student, Exam, and Result.

import java.util.Scanner;

class **Student**

{

protected int rollNumber;

protected String name;

protected String branch;

public **Student**(int rollNumber, String name, String branch)

{

this.rollNumber = rollNumber;

this.name = name;

this.branch = branch;

}

public void **displayStudentDetails**(

{

System.out.println("Roll Number: " + rollNumber);

System.out.println("Name: " + name);

System.out.println("Branch: " + branch);

}

}

class Exam **extends** Student

{

protected int[] subjectMarks;

// Constructor to initialize student and subject marks

public **Exam**(int rollNumber, String name, String branch, int[] subjectMarks)

{

super(rollNumber, name, branch);

this.subjectMarks = subjectMarks;

}

public void **displayExamDetails**()

{

displayStudentDetails();

System.out.println("Subject Marks:");

for (int i = 0; i < subjectMarks.length; i++)

{

System.out.println("Subject " + (i+1) + ": " + subjectMarks[i]);

}

}

}

class Result **extends** Exam

{

private int totalMarks;

private String result;

// Constructor to initialize student details, subject marks, and calculate the result

public **Result**(int rollNumber, String name, String branch, int[] subjectMarks)

{

super(rollNumber, name, branch, subjectMarks);

calculateResult();

}

// Method to calculate total marks and determine the result

private void **calculateResult**()

{

totalMarks = 0;

for (int marks : subjectMarks)

{

totalMarks += marks;

}

// Determine result based on total marks

result = totalMarks >= 150 ? "Pass" : "Fail"; // Assuming 150 is the passing criteria

}

public void **displayResult**()

{

System.out.println("Result:");

System.out.println("--------------");

displayExamDetails();

System.out.println("Total Marks: " + totalMarks);

System.out.println("Result: " + result);

}

}

public class **TestStudentExamResult**

{

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

{

Scanner scanner = new Scanner(System.in);

System.out.print("Roll Number: ");

int rollNumber = scanner.nextInt();

scanner.nextLine();

System.out.print("Name: ");

String name = scanner.nextLine();

System.out.print("Branch: ");

String branch = scanner.nextLine();

int[] subjectMarks = new int[3];

System.out.println("Marks:");

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

{

System.out.print("Subject " + (i+1) + ": ");

subjectMarks[i] = scanner.nextInt();

}

Result studentResult = new Result(rollNumber, name, branch, subjectMarks);

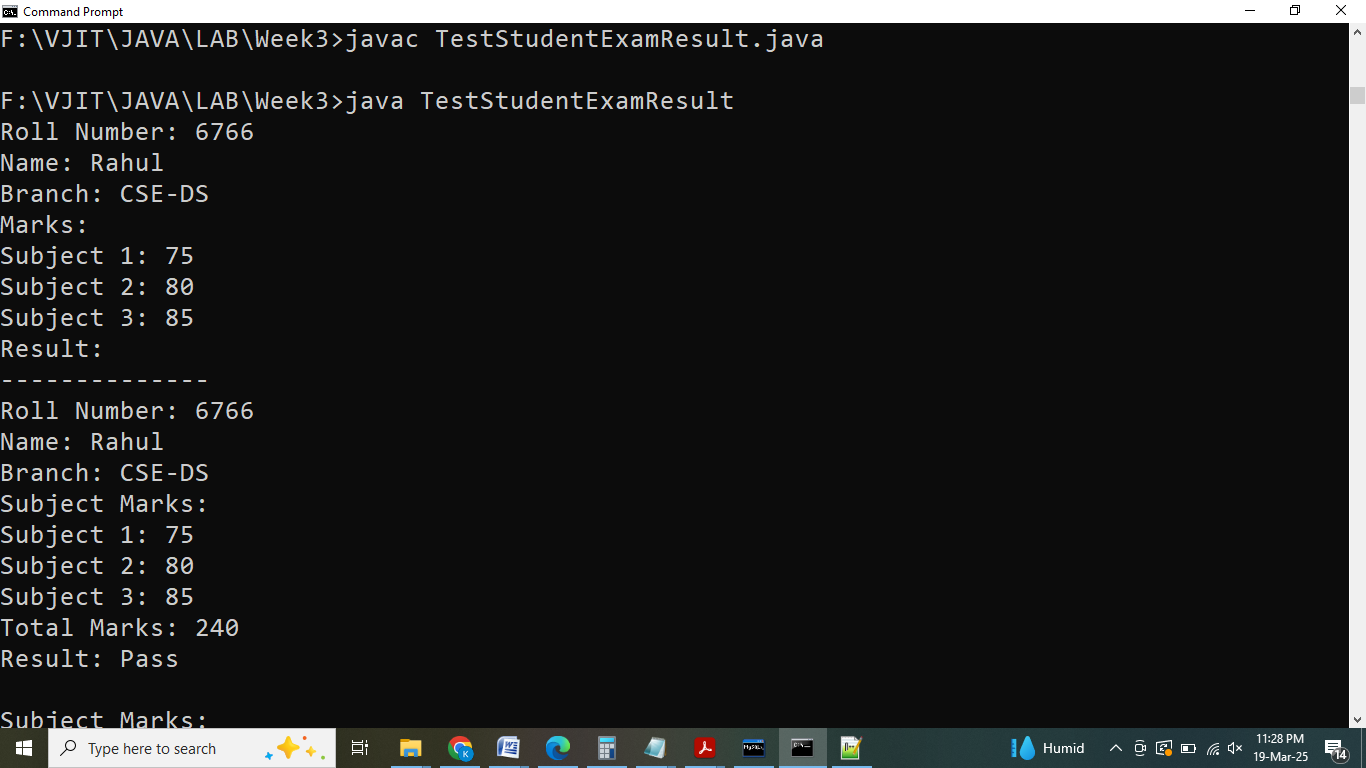
studentResult.displayResult();

scanner.close();

}

}

**Output:**



**6.3** Write a Java program to demonstrate **hierarchical inheritance** involving a base class Number and two derived classes Square and Cube.

import java.util.Scanner;

// Base class Number

class **Number**

{

double number;

// Constructor to initialize the number

public **Number**(double number)

{

this.number = number;

}

}

// Derived class 1 - Square

class Square **extends** Number

{

// Constructor to initialize the number for Square

public **Square**(double number)

{

super(number); // Call the base class constructor

}

// Method to calculate and return the square

public double **calculateSquare**()

{

return number \* number;

}

}

// Derived class 2 - Cube

class Cube **extends** Number

{

// Constructor to initialize the number for Cube

public **Cube**(double number)

{

super(number); // Call the base class constructor

}

// Method to calculate and return the cube

public double **calculateCube**()

{

return number \* number \* number;

}

}

// Main class

public class **HierarchicalInheritance**

{

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

{

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

double inputNumber = scanner.nextDouble();

// Create objects of derived classes

Square squareObject = new Square(inputNumber);

Cube cubeObject = new Cube(inputNumber);

// Display the number, square, and cube

System.out.println("Number: " + inputNumber);

System.out.println("Square: " + squareObject.calculateSquare());

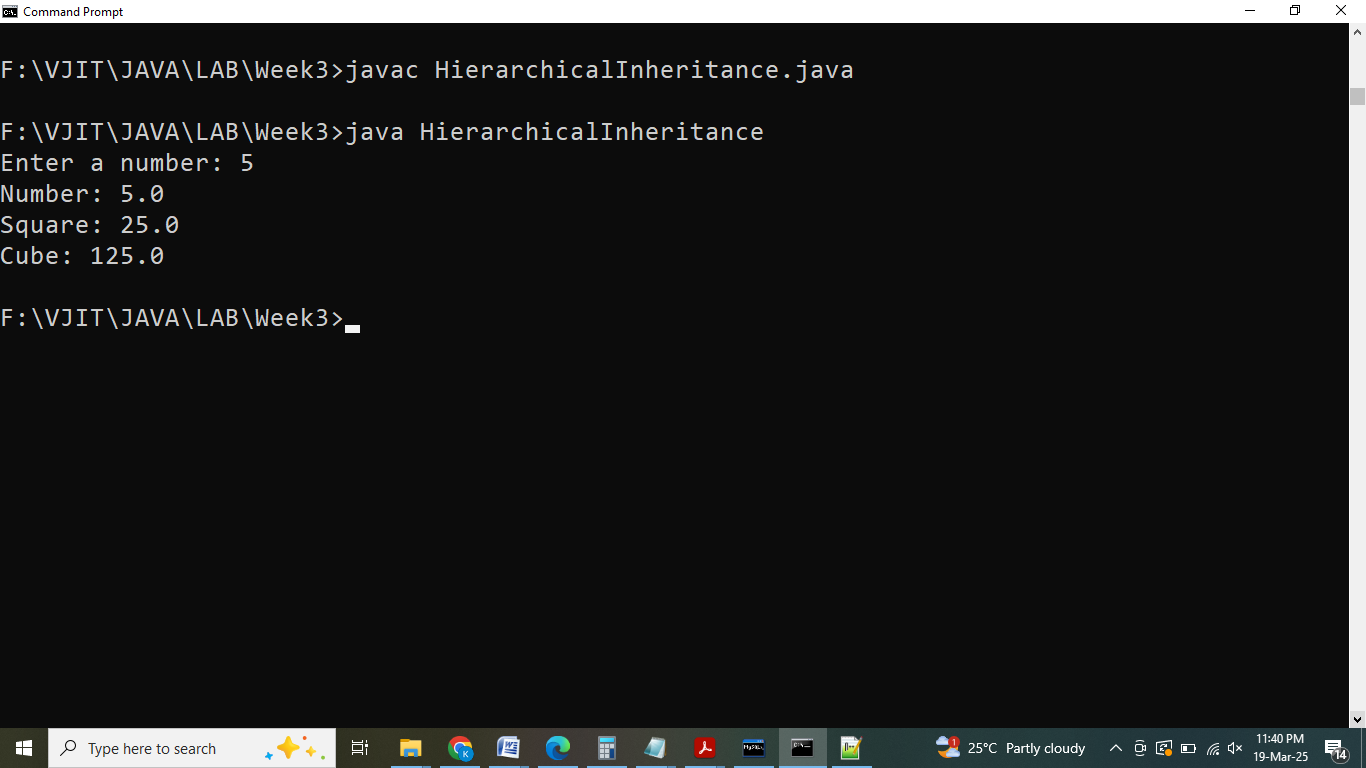
System.out.println("Cube: " + cubeObject.calculateCube());

scanner.close();

}

}

**Output:**



**6.4** Write a Java Program to demonstrate Interface concept **(Implementation Multiple inheritance using interface),**

Implement necessary methods to calculate the Area of circle, Perimeter of the circle and getColor method to return the color and the main method has been provided.

import java.util.\*;

interface **Shape**

{

double calculateArea();

double calculatePerimeter();

}

interface **Color**

{

String getColor();

}

class Circle **implements** Shape, Color

{

private double radius;

private String color;

//write your code here

// Constructor to initialize radius and color

public **Circle**(double radius, String color)

{

this.radius = radius;

this.color = color;

}

// Implementing calculateArea method from Shape interface

@Override

public double **calculateArea**()

{

return Math.PI \* radius \* radius; // Area of a circle = π \* r²

}

// Implementing calculatePerimeter method from Shape interface

@Override

public double **calculatePerimeter**()

{

return 2 \* Math.PI \* radius; // Perimeter (circumference) of a circle = 2 \* π \* r

}

// Implementing getColor method from Color interface

@Override

public String **getColor**()

{

return color;

}

}

public class **MultipleInheritance**

{

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

{

Scanner scanner = new Scanner(System.in);

System.out.print("Radius of the circle: ");

double radius = scanner.nextDouble();

scanner.nextLine();

System.out.print("Color of the circle: ");

String color = scanner.nextLine();

Circle circle = new Circle(radius, color);

System.out.println("---------------");

// Formatting double values to 2 decimal places

System.out.printf("Circle Area: %.2f%n", circle.calculateArea());

System.out.printf("Circle Perimeter: %.2f%n", circle.calculatePerimeter());

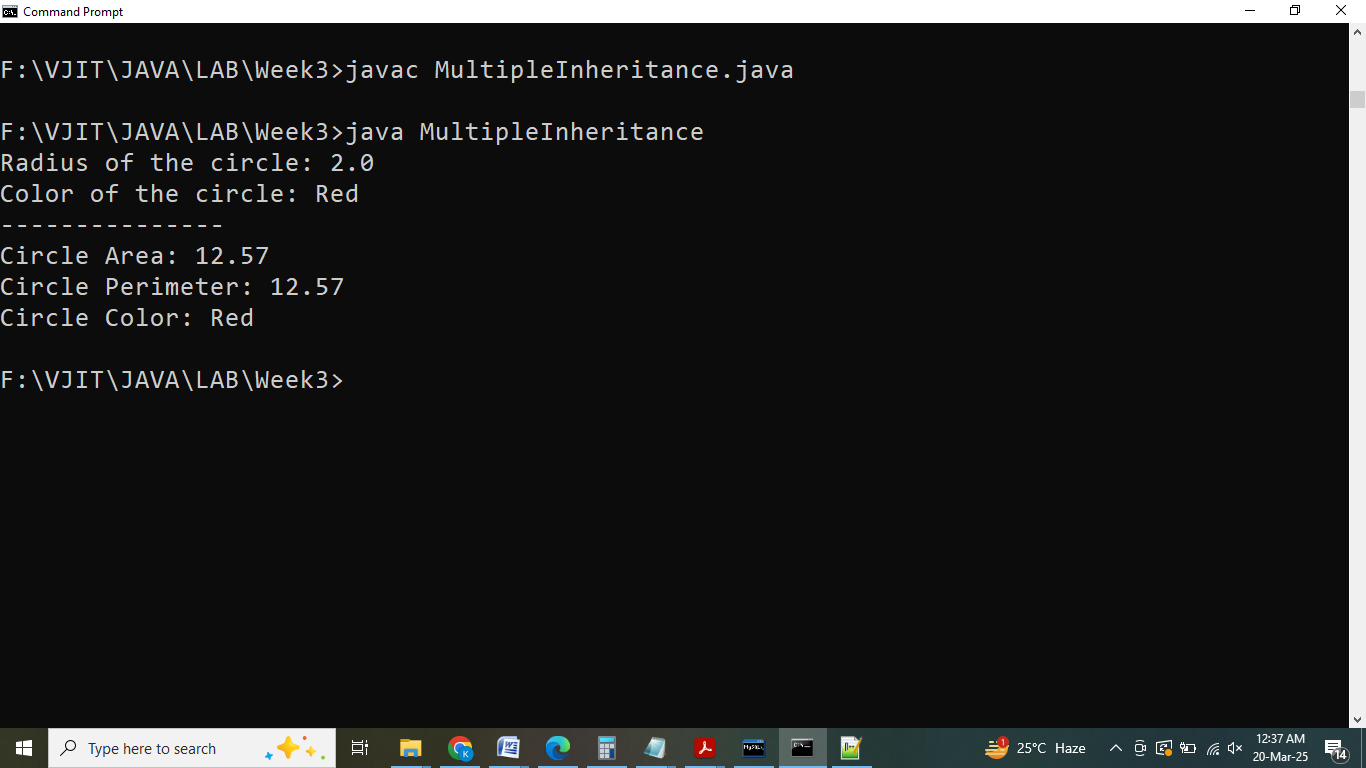
System.out.println("Circle Color: " + circle.getColor());

scanner.close();

}

}

**Output:**



**6.5** Write a Java program to demonstrate the Interface concept by implementing **Hybrid Inheritance** using interfaces.

Create a Triangle class that implements the Shape and Color interfaces. Implement the necessary methods to calculate the Area of the triangle, the Perimeter of the triangle, and return the Color of the triangle.

import java.util.Scanner;

interface **Shape**

{

double calculateArea(); // Method to calculate area

double calculatePerimeter(); // Method to calculate perimeter

}

interface **Color**

{

String getColor(); // Method to get the color of the shape

}

class Triangle **implements** Shape, Color

{

private double sideA;

private double sideB;

private double sideC;

private String color;

// Constructor to initialize the sides of the triangle and its color

public **Triangle**(double sideA, double sideB, double sideC, String color)

{

this.sideA = sideA;

this.sideB = sideB;

this.sideC = sideC;

this.color = color;

}

// Implementing calculateArea method from Shape interface using Heron's formula

@Override

public double **calculateArea**()

{

double s = (sideA + sideB + sideC) / 2; // Semi-perimeter

// Area calculation using Heron's formula

return Math.sqrt(s \* (s - sideA) \* (s - sideB) \* (s - sideC));

}

// Implementing calculatePerimeter method from Shape interface

@Override

public double **calculatePerimeter**()

{

return sideA + sideB + sideC; // Perimeter = sum of all sides

}

// Implementing getColor method from Color interface

@Override

public String **getColor**()

{

return color; // Return the color of the triangle

}

}

public class **HybridInheritance**

{

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

{

Scanner scanner = new Scanner(System.in);

// Input: sides of the triangle and its color

System.out.print("Side A: ");

double sideA = scanner.nextDouble();

System.out.print("Side B: ");

double sideB = scanner.nextDouble();

System.out.print("Side C: ");

double sideC = scanner.nextDouble();

scanner.nextLine(); // Consume newline left-over

System.out.print("Color: ");

String color = scanner.nextLine();

// Create Triangle object

Triangle triangle = new Triangle(sideA, sideB, sideC, color);

System.out.println("-------------------");

// Output the area, perimeter, and color of the triangle

System.out.printf("Area: %.2f\n", triangle.calculateArea());

System.out.printf("Perimeter: %.2f\n", triangle.calculatePerimeter());

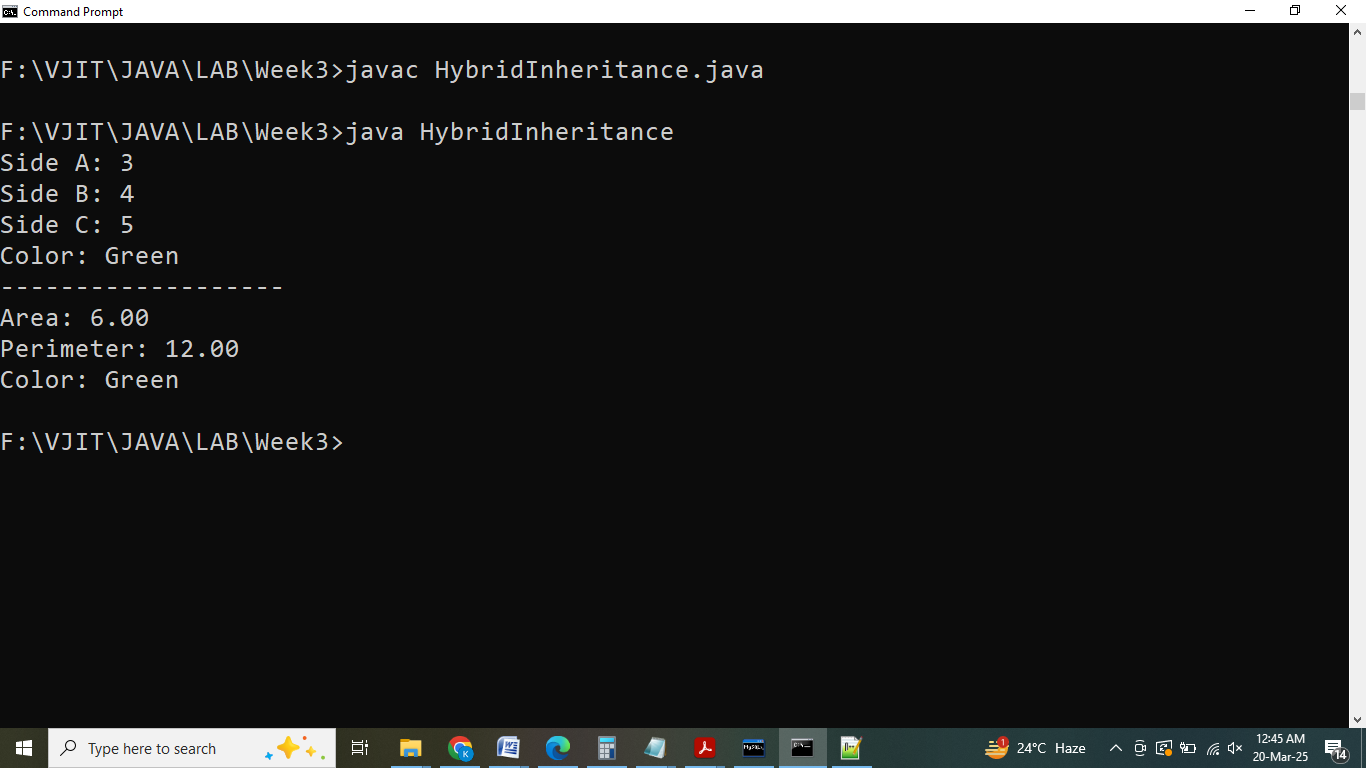
System.out.println("Color: " + triangle.getColor());

scanner.close();

}

}

**Output:**



**7. Employee - Method Overriding**

You are developing a program to simulate different roles in a software development team. Design a class named TeamMember with a method performTask() that prints "Performing a task as a team member".

Now, create two subclasses, Developer and ProductOwner, which extend the TeamMember class's functionality by overriding the performTask() method.

import java.util.Scanner;

class **TeamMember**

{

// write your code here

public void **performTask**()

{

System.out.println("Performing a task as a team member");

}

}

class Developer **extends** TeamMember

{

// write your code here

@Override

public void **performTask**()

{

Scanner scanner = new Scanner(System.in);

// Prompt for the programming language

System.out.print("Enter the programming language: ");

String programmingLanguage = scanner.nextLine();

// Output message indicating the developer and their programming language

System.out.println("I am a developer coding in " + programmingLanguage);

}

}

class ProductOwner **extends** TeamMember

{

// write your code here

@Override

public void **performTask**()

{

Scanner scanner = new Scanner(System.in);

// Prompt for the project being managed

System.out.print("Enter the project: ");

String project = scanner.nextLine();

// Output message indicating the product owner and their project

System.out.println("I am a product owner managing the " + project);

}

}

public class **TeamTest**

{

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

{

// Creating an instance of Developer

Developer developer = new Developer();

developer.performTask();

// Creating an instance of ProductOwner

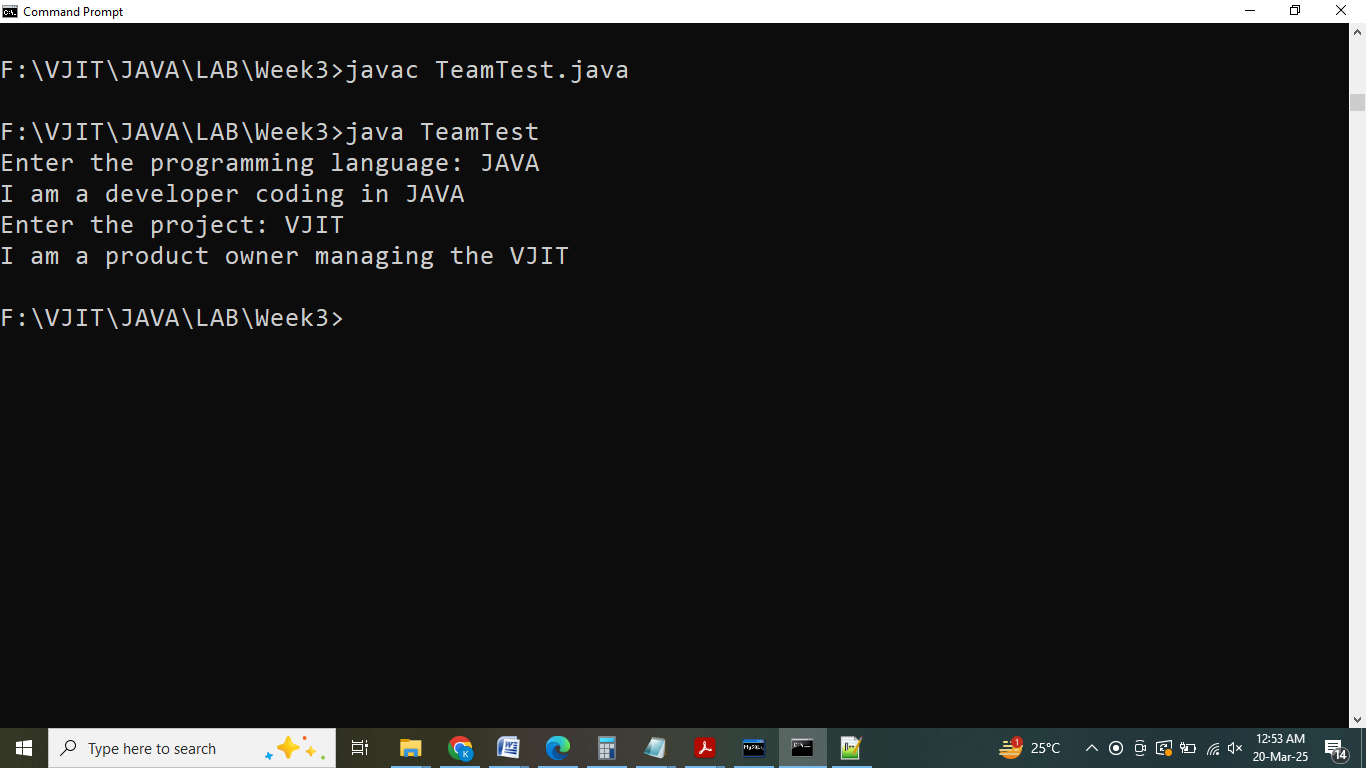
ProductOwner productOwner = new ProductOwner();

productOwner.performTask();

}

}

**Output:**



**8. Dynamic method dispatch**

Implement runtime polymorphism using the dynamic method dispatch for the scenario given:

Create three classes Rectangle (Parent), Square(Child of Rectangle), and Triangle (Child of Rectangle). These classes contain a method “area” to calculate the area of the shape using appropriate expressions.

import java.util.Scanner;

class **Rectangle**

{

public void **area**()

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter length and width of Rectangle:");

double length = sc.nextDouble();

double width = sc.nextDouble();

int area = (int) Math.floor(length \* width);

System.out.println("Area of rectangle:" + area);

}

}

class Square **extends** Rectangle

{

public void **area**()

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter side of square:");

int side = sc.nextInt();

int area = side \* side;

System.out.println("Area of square:" + area);

}

}

class Triangle **extends** Rectangle

{

public void **area**()

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter base and height of Traingle:");

double base = sc.nextDouble();

double height = sc.nextDouble();

int area =(int)Math.floor(0.5 \* base \* height);

System.out.println("Area of triangle:" + area);

}

}

class **Calculation**

{

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

{

Rectangle r = new Rectangle();

r.area();

r = new Square();

r.area();

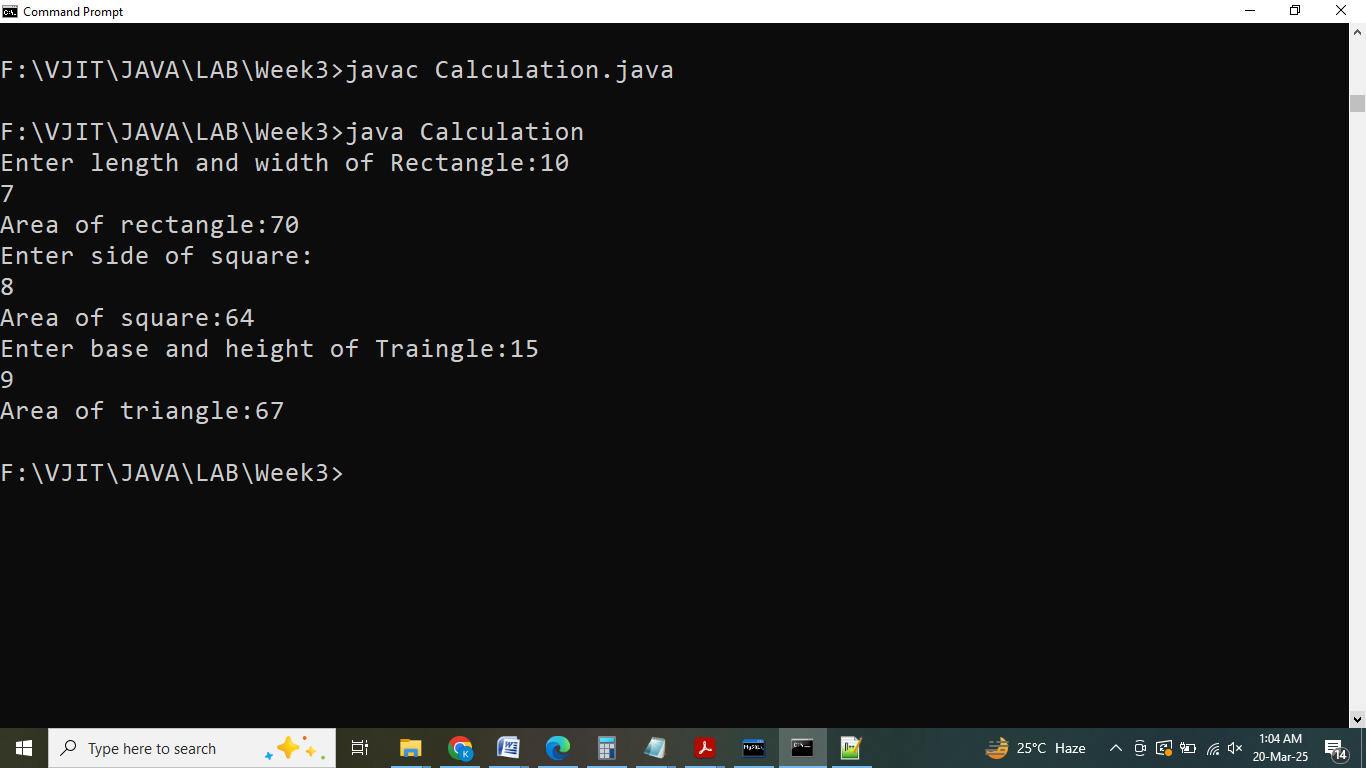
r = new Triangle();

r.area();

}

}

**Output:**



**9.** Write a Java program that creates **a final class** called ImmutablePoint to represent a point in a 2D plane. The class should have two final instance variables x and y to store the coordinates of the point.

**Implement the following:**

1. A constructor ImmutablePoint(double x, double y) that initializes the x and y coordinates.
2. Getter methods getX() and getY() to retrieve the values of x and y, respectively.
3. A method calculateDistance(ImmutablePoint other) that takes another ImmutablePoint object as input and calculates the Euclidean distance between the current point and the given point using the formula sqrt((x2 - x1)^2 + (y2 - y1)^2).

import java.util.Scanner;

public **final** class **ImmutablePoint**

{

// Final instance variables to represent the coordinates of the point

private final double x;

private final double y;

// Constructor to initialize the coordinates

public **ImmutablePoint**(double x, double y)

{

this.x = x;

this.y = y;

}

// Getter methods to retrieve the coordinates

public double **getX**()

{

return x;

}

public double getY()

{

return y;

}

// Method to calculate the distance between two points

public double **calculateDistance**(ImmutablePoint other)

{

double dx = other.getX() - this.x;

double dy = other.getY() - this.y;

return Math.sqrt(dx \* dx + dy \* dy); // Euclidean distance formula

}

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

{

Scanner scanner = new Scanner(System.in);

// Input coordinates for the first point

System.out.print("x-coordinate for first point: ");

double x1 = scanner.nextDouble();

System.out.print("y-coordinate for first point: ");

double y1 = scanner.nextDouble();

ImmutablePoint point1 = new ImmutablePoint(x1, y1);

// Input coordinates for the second point

System.out.print("x-coordinate for second point: ");

double x2 = scanner.nextDouble();

System.out.print("y-coordinate for second point: ");

double y2 = scanner.nextDouble();

ImmutablePoint point2 = new ImmutablePoint(x2, y2);

// Close the scanner

scanner.close();

// Calculate and display the distance between the two points

double distance = point1.calculateDistance(point2);

System.out.println("-------------------");

System.out.println("Distance: " + distance);

}

}

**Output:**

