**Experiment No. 06**

**Experiment Name: Experiment with OOP Features**

*Course title: Programming Language II(Java) Lab*

*Course code:*

*Spring 2025*

**Date of Submission**:



**Submitted to-**

###### **Md. Rafsan Jani**

*Assistant Professor*

*Department of Computer Science and Engineering*

|  |  |  |
| --- | --- | --- |
| **Sl** | Class Roll | Name |
| 01 | 2023000010001 | Md Samaul Islam |
|  |  |  |

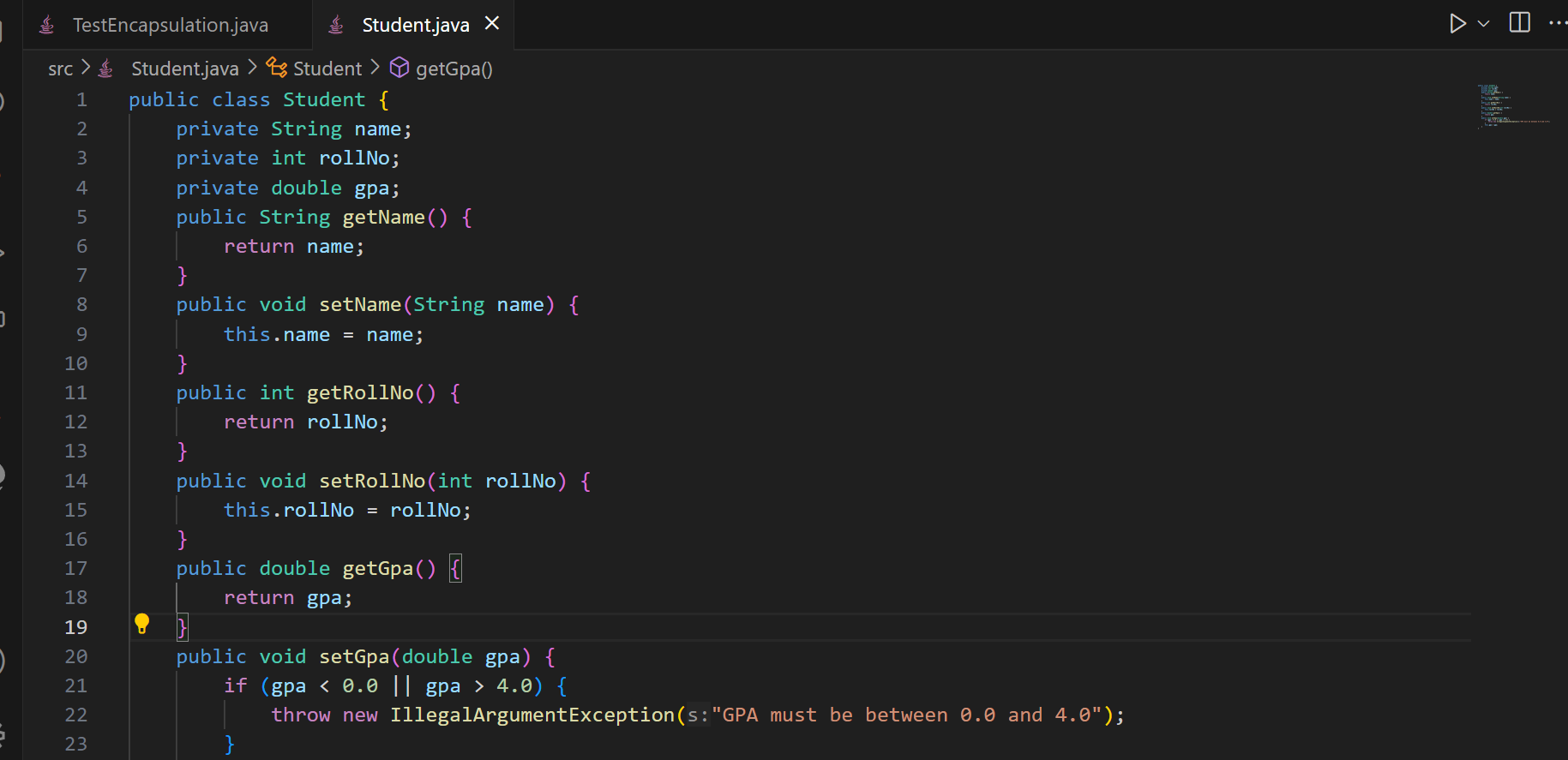
Hw 1: Create a Student class with private fields:

name,

rollNo,

gpa

Provide public getters and setters.

Validate that gpa cannot be negative or over 4.0.  
**Student.java**  


public class Student {

    private String name;

    private int rollNo;

    private double gpa;

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public int getRollNo() {

        return rollNo;

    }

    public void setRollNo(int rollNo) {

        this.rollNo = rollNo;

    }

    public double getGpa() {

        return gpa;

    }

    public void setGpa(double gpa) {

        if (gpa < 0.0 || gpa > 4.0) {

            throw new IllegalArgumentException("GPA must be between 0.0 and 4.0");

        }

        this.gpa = gpa;

    }

}

**TestEncapsulation.java**

public class TestEncapsulation {

    public static void main(String[] args) {

        Student obj = new Student();

        obj.setName("Arafat");

        obj.setRollNo(51);

        try {

            obj.setGpa(3.5);

        } catch (IllegalArgumentException e) {

            System.out.println(  e.getMessage());

        }

        System.out.println("Student's name: " + obj.getName());

        System.out.println("Student's rollNo: " + obj.getRollNo());

        System.out.println("Student's gpa: " + obj.getGpa());

    }

}

Hw 2:

Create a base class Shape with method getArea().

Derive classes

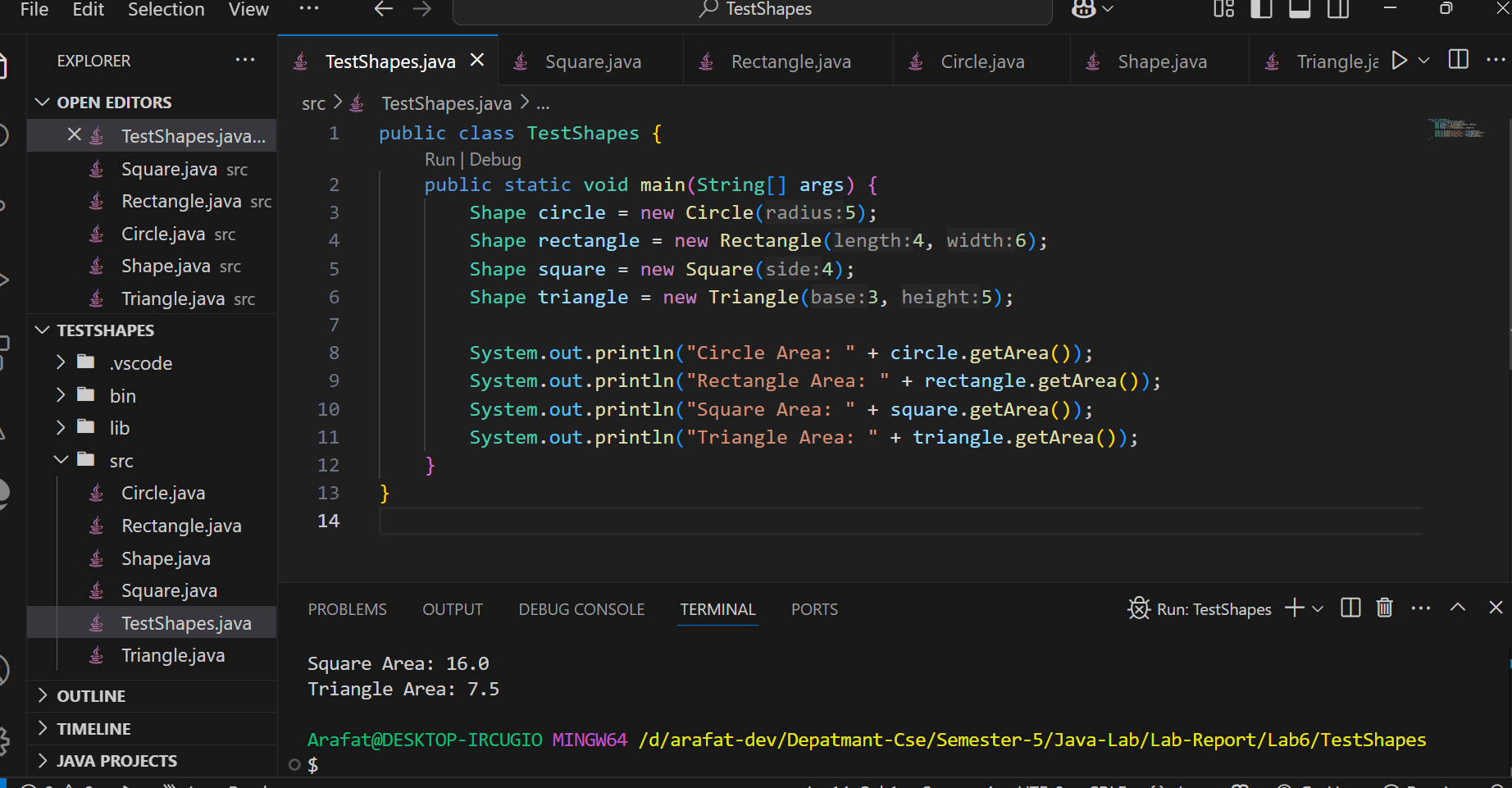
Circle,

Rectangle,

Square, and

Triangle.

Override getArea() appropriately in each subclass.



**Shape.java**

public class Shape {

    public double getArea() {

        return 0.0;

    }

}

**Circle.java**public class Circle extends Shape {

    private double radius;

    public Circle(double radius) {

        this.radius = radius;

    }

    @Override

    public double getArea() {

        return Math.PI \* radius \* radius;

    }

}

**Rectangle.java**

public class Rectangle extends Shape {

    private double length;

    private double width;

    public Rectangle(double length, double width) {

        this.length = length;

        this.width = width;

    }

    @Override

    public double getArea() {

        return length \* width;

    }

}

**Square.java**

public class Square extends Shape {

    private double side;

    public Square(double side) {

        this.side = side;

    }

    @Override

    public double getArea() {

        return side \* side;

    }

}

**Triangle.java**

public class Triangle extends Shape {

    private double base;

    private double height;

    public Triangle(double base, double height) {

        this.base = base;

        this.height = height;

    }

    @Override

    public double getArea() {

        return 0.5 \* base \* height;

    }

}

**TestShapes.java**

public class TestShapes {

    public static void main(String[] args) {

        Shape circle = new Circle(5);

        Shape rectangle = new Rectangle(4, 6);

        Shape square = new Square(4);

        Shape triangle = new Triangle(3, 5);

        System.out.println("Circle Area: " + circle.getArea());

        System.out.println("Rectangle Area: " + rectangle.getArea());

        System.out.println("Square Area: " + square.getArea());

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

    }

}

**Hw 3:** Create a class Animal with a method makeSound().

Create subclasses

Dog,

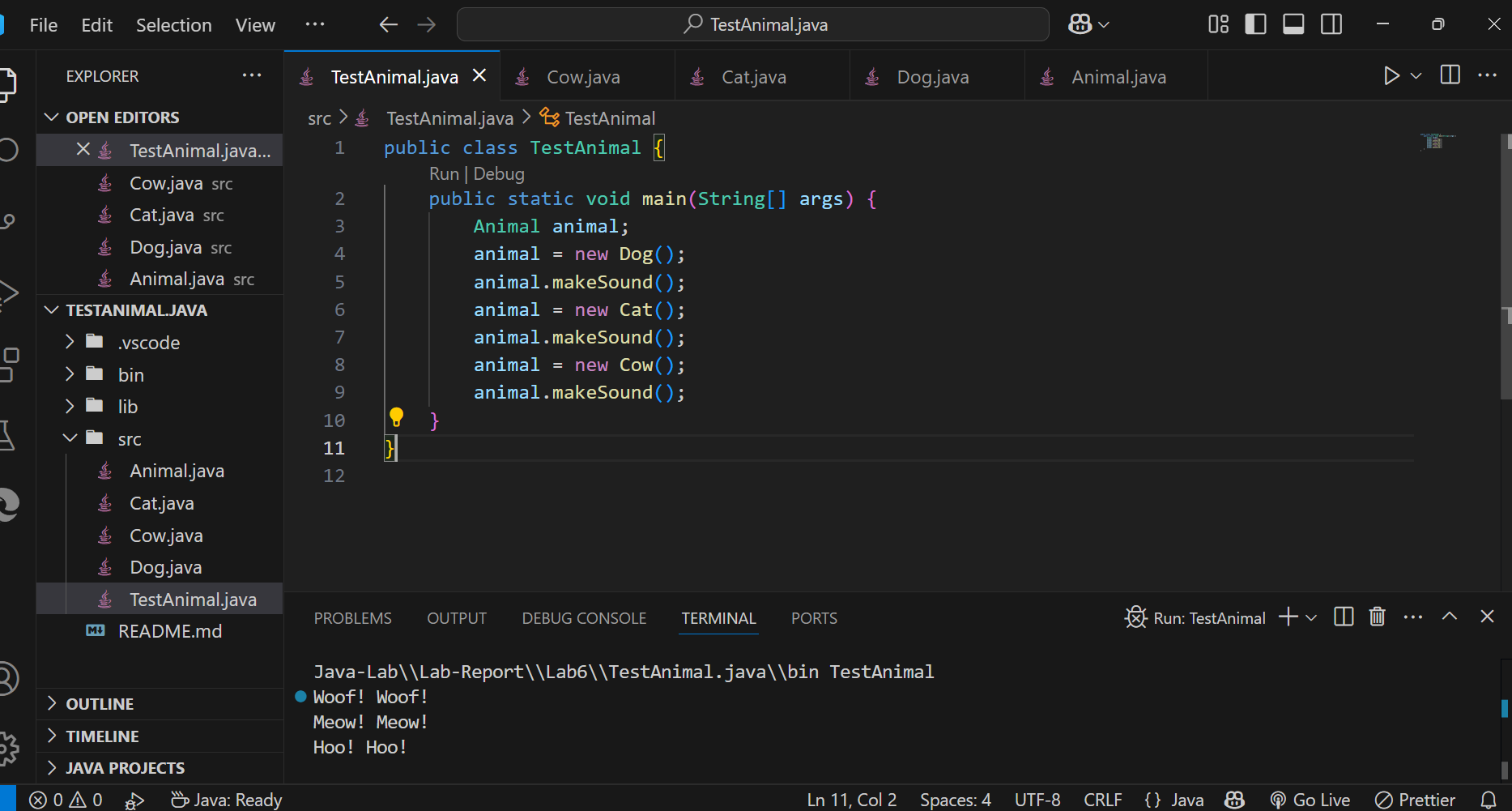
Cat, and

Cow,

override makeSound() to print unique sounds.

Use dynamic method dispatch to demonstrate

polymorphism.



**Animal.java**

public class Animal {

    public void makeSound() {

        System.out.println("Some generic animal sound");

    }

}

**Dog.java**

public class Dog extends Animal {

    @Override

    public void makeSound() {

        System.out.println("Woof! Woof!");

    }

}

**Cat.java**

public class Cat extends Animal {

    @Override

    public void makeSound() {

        System.out.println("Meow! Meow!");

    }

}

**Cow.java**

public class Cow extends Animal {

    @Override

    public void makeSound() {

        System.out.println("Hoo! Hoo!");

    }

}

**TestAnimal.java**

public class TestAnimal {

    public static void main(String[] args) {

        Animal animal;

        animal = new Dog();

        animal.makeSound();

        animal = new Cat();

        animal.makeSound();

        animal = new Cow();

        animal.makeSound();

    }

}