## Assignment – 5

## Assignment on Inheritance, Overriding and Abstraction

1. Inheritance

To add a `sing` method to the `Bird` class and modify the main method to print the specified lines, here is the code:

```
class Animal {
    void walk() {
        System.out.println("I am walking");
    }
}

class Bird extends Animal {
    void fly() {
        System.out.println("I am flying");
    }

    void sing() {
        System.out.println("I am singing");
    }
}

public class Solution {
    public static void main(String[] args) {
        Bird bird = new Bird();
        bird.walk();
        bird.fly();
        bird.sing();
    }
}
```

2. Abstraction

To create the `MyBook` class that extends the abstract `Book` class, here is the code:

```
abstract class Book {
    String title;
    abstract void setTitle(String s);
    String getTitle() {
        return title;
    }
}

class MyBook extends Book {
    @Override
    void setTitle(String s) {
        title = s;
    }
}

public class Main {
    public static void main(String[] args) {
        MyBook new_novel = new MyBook();
        new_novel.setTitle("A tale of two cities");
        System.out.println("The title is: " + new_novel.getTitle());
    }
}
```

3. Overloading

To override the `getNumberOfTeamMembers` method in the `Soccer` class, here is the code:

```
class Sports {
    String getName() {
        return "Generic Sports";
    }
    void getNumberOfTeamMembers() {
        System.out.println("Each team has n players in " + getName());
    }
}

class Soccer extends Sports {
    @Override
    String getName() {
        return "Soccer Class";
    }

    @Override
    void getNumberOfTeamMembers() {
        System.out.println("Each team has 11 players in " + getName());
    }
}

public class Main {
    public static void main(String[] args) {
        Sports sport = new Sports();
        System.out.println(sport.getName());
        sport.getNumberOfTeamMembers();

        Soccer soccer = new Soccer();
        System.out.println(soccer.getName());
        soccer.getNumberOfTeamMembers();
    }
}
```

4. Overriding (Duplicate Task)

The fourth task is identical to the third one. Therefore, the same solution applies:

```
class Sports {
   String getName() {
      return "Generic Sports";
   }

   void getNumberOfTeamMembers() {
      System.out.println("Each team has n players in " + getName());
   }
}

class Soccer extends Sports {
   @Override
   String getName() {
      return "Soccer Class";
   }

   @Override
   void getNumberOfTeamMembers() {
      System.out.println("Each team has 11 players in " + getName());
   }
}

public class Main {
   public static void main(String[] args) {
      Sports sport = new Sports();
      System.out.println(sport.getName());
      sport.getNumberOfTeamMembers();

      Soccer soccer = new Soccer();
      System.out.println(soccer.getName());
      soccer.getNumberOfTeamMembers();
```

```
    }
}
```

With these implementations, each task should now meet the specified requirements and produce the expected output.

## 4 Shape Calculator

Task: Create a Java class hierarchy representing different shapes (e.g., Circle, Rectangle) and implement methods to calculate their area.

Requirements:

Define a base class Shape with an abstract method calculateArea() that returns the area of the shape (double).

Create subclasses Circle and Rectangle that extend Shape.

Implement the calculateArea() method in each subclass specific to the shape's formula (e.g., PI * radius^2 for circle, length * width for rectangle).

In the main method, create instances of Circle and Rectangle with user-provided dimensions (e.g., radius, length, and width).

Call the calculateArea() method on each shape object and display the results.

Bonus:

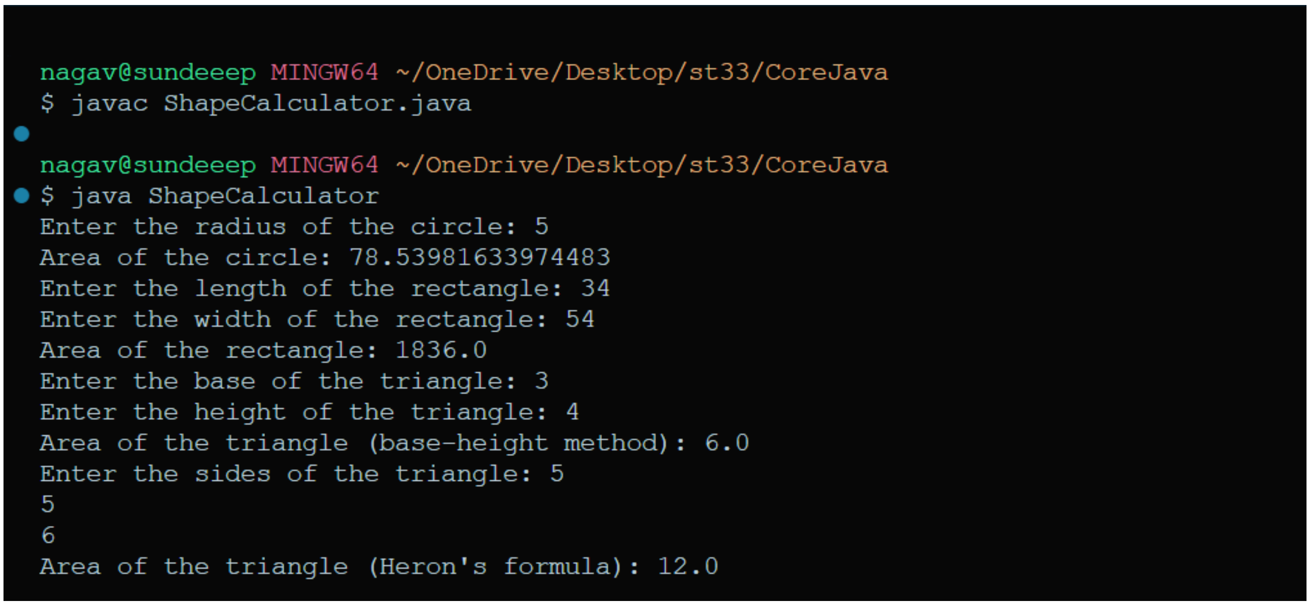
Add a Triangle subclass with its calculateArea() implementation.

Implement method overloading in the Shape class for a calculateArea(double... sides) method that can handle shapes with variable sides (e.g., triangle).

**Answer:**

```
import java.util.Scanner;

abstract class Shape {
    // Abstract method to calculate the area of the shape
    public abstract double calculateArea();
    // Method overloading for shapes with variable sides
    public double calculateArea(double... sides) {
        return 0.0; // Default implementation, to be overridden by subclasses if
needed
    }
}

class Circle extends Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double calculateArea() {
        return Math.PI * Math.pow(radius, 2);
    }
}

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 calculateArea() {
        return length * width;
    }
}

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 calculateArea() {
        return 0.5 * base * height;
    }

    // Overloaded method to calculate the area using Heron's formula
    @Override
    public double calculateArea(double... sides) {
        if (sides.length != 3) {
            throw new IllegalArgumentException("Triangle must have 3 sides.");
        }
        double a = sides[0];
        double b = sides[1];
        double c = sides[2];
        double s = (a + b + c) / 2; // Semi-perimeter
        return Math.sqrt(s * (s - a) * (s - b) * (s - c));
    }
}

public class ShapeCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Circle
        System.out.print("Enter the radius of the circle: ");
        double radius = scanner.nextDouble();
        Circle circle = new Circle(radius);
        System.out.println("Area of the circle: " + circle.calculateArea());

        // Rectangle
        System.out.print("Enter the length of the rectangle: ");
        double length = scanner.nextDouble();
        System.out.print("Enter the width of the rectangle: ");
        double width = scanner.nextDouble();
        Rectangle rectangle = new Rectangle(length, width);
        System.out.println("Area of the rectangle: " + rectangle.calculateArea());

        // Triangle
        System.out.print("Enter the base of the triangle: ");
        double base = scanner.nextDouble();
        System.out.print("Enter the height of the triangle: ");
        double height = scanner.nextDouble();
        Triangle triangle = new Triangle(base, height);
        System.out.println("Area of the triangle (base-height method): " +
triangle.calculateArea());

        // Triangle with sides
        System.out.print("Enter the sides of the triangle: ");
        double side1 = scanner.nextDouble();
        double side2 = scanner.nextDouble();
        double side3 = scanner.nextDouble();
        System.out.println("Area of the triangle (Heron's formula): " +
triangle.calculateArea(side1, side2, side3));

        scanner.close();
    }
}
```

**Output:**



```
nagav@sundeeep MINGW64 ~/OneDrive/Desktop/st33/CoreJava
$ javac ShapeCalculator.java

nagav@sundeeep MINGW64 ~/OneDrive/Desktop/st33/CoreJava
$ java ShapeCalculator
Enter the radius of the circle: 5
Area of the circle: 78.53981633974483
Enter the length of the rectangle: 34
Enter the width of the rectangle: 54
Area of the rectangle: 1836.0
Enter the base of the triangle: 3
Enter the height of the triangle: 4
Area of the triangle (base-height method): 6.0
Enter the sides of the triangle: 5
5
6
Area of the triangle (Heron's formula): 12.0
```