Assignment-7 zhuang

```
import java.util.List;
import java.util.ArrayList;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
public class Driver implements Serializable {
   public static void main(String[] args) {
       // Create a list to store shape objects
       List<Shapes> mylist = new ArrayList<>();
       // Create a circle with radius 2.00 and add to the list
       Circle circle = new Circle(2.00);
       mylist.add(circle);
       // Create two rectangles with different dimensions and add to the
list
       Rectangle rectangle01 = new Rectangle(2.11, 3.11);
       mylist.add(rectangle01);
       Rectangle rectangle02 = new Rectangle(1.24, 4.21);
       mylist.add(rectangle02);
       // Calculate the area for each shape in the list
       for (int i = 0; i < mylist.size(); i++) {
           mylist.get(i).calculateArea();
       }
       // Serialize the list of shapes to a file named 'Shape.dat'
       try {
           FileOutputStream fileStream = new
FileOutputStream("Shape.dat");
           ObjectOutputStream os = new ObjectOutputStream(fileStream);
           os.writeObject(mylist);
           os.close();
       } catch (Exception e) {
           e.printStackTrace();
       }
       // Deserialize the 'Shape.dat' file to retrieve the list of
shapes
       try {
           FileInputStream fileStream = new
FileInputStream("Shape.dat");
           ObjectInputStream os = new ObjectInputStream(fileStream);
           List<Shapes> list = (List<Shapes>) os.readObject();
           // Print out the details of each shape in the deserialized
list
```

```
            for (Shapes shape : list) {
                System.out.println(shape.toString());
            }
            os.close();
            // Print the largest shape in the list
            System.out.println("The biggest one is " +
largestShape(list).toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    // Method to find the largest shape in a given list of shapes
    public static Shapes largestShape(List<Shapes> list) {
        Shapes largest = list.get(0);
        // Iterate through the list to find the shape with the largest
area
        for (Shapes shape : list) {
            if (largest.compareShapes(shape) < 0) {
                largest = shape;
            }
        }
        return largest;
    }
}



import java.io.Serializable;
public abstract class Shapes implements ShapesRelate, Serializable {
    protected double area; // Protected variable to store the area of the
shape
    // Default constructor initializing the area to 0.00
    public Shapes() {
        this.area = 0.00;
    }
    // Method to return the area of the shape
    public double getArea() {
        return area;
    }
    // Abstract method to calculate the area, to be implemented in
subclasses
    abstract void calculateArea();
    // Method to compare this shape with another shape based on their
areas
    public int compareShapes(ShapesRelate shapesRelate) {
        Shapes shape = (Shapes) shapesRelate;
```

```
        // Returning 1 if this shape's area is greater, 0 if equal, and
-1 if lesser
        if (this.area > shape.getArea()) {
            return 1;
        } else if (this.area == shape.getArea()) {
            return 0;
        } else {
            return -1;
        }
    }
}


public class Rectangle extends Shapes {

    private double length; // Variable to store the length of the
rectangle
    private double width; // Variable to store the width of the rectangle

    // Default constructor initializing length and width to 0.00
    public Rectangle() {
        this.length = 0.00;
        this.width = 0.00;
    }
    // Constructor with parameters to set the length and width of the
rectangle
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Setter method for width
    public void setWidth(double width) {
        this.width = width;
    }

    // Getter method for width
    public double getWidth() {
        return width;
    }

    // Setter method for length
    public void setLength(double length) {
        this.length = length;
    }
```

```
    // Getter method for length
    public double getLength() {
        return length;
    }

    // Overridden toString method to display the area of the rectangle
    public String toString() {
        return "Rectangle Area is " + area;
    }

    // Method to calculate and update the area of the rectangle
    public void calculateArea() {
        area = width * length;
    }
}


public class Circle extends Shapes {

    private double radius; // Private variable to store the radius of the
circle
    private final double PI = 3.14; // Constant for the value of PI

    // Default constructor initializing the radius to 0.00
    public Circle() {
        radius = 0.00;
    }
    // Constructor with a parameter to set the radius of the circle
    public Circle(double radius) {
        this.radius = radius;
    }

    // Setter method for the radius
    public void setRadius(double radius) {
        this.radius = radius;
    }

    // Getter method for the radius
    public double getRadius() {
        return radius;
    }

    // Overridden method from Shapes to calculate the area of the circle
    public void calculateArea() {
        area = PI * (radius * radius);
    }
```

```
    // Overridden toString method to display the area of the circle
    public String toString() {
        return "Circle area is " + area;
    }
}



public interface ShapesRelate {
      int compareShapes(ShapesRelate shapesRelate);

      }
```

outPut: