Programming I - Assignment 6

Student Name: Evelyn Toledo Lally
Student Number: 22102020

Question 1

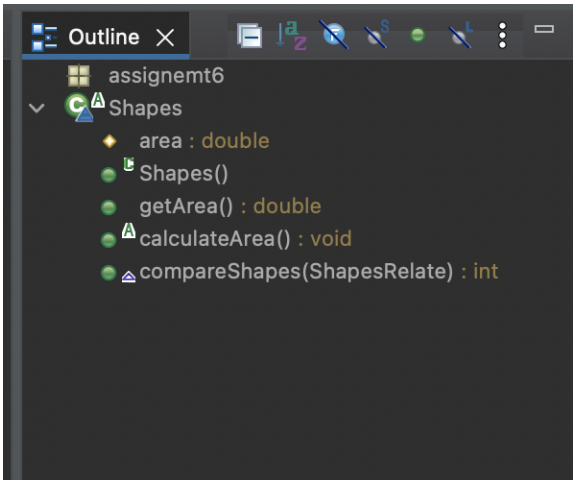
1. Implement in Java, the interface and classes as shown in the UML diagram above. Note: The area of a rectangle is length*width and the area of a circle is PI * radius2
2. The assignment requires you to write a driver class which: creates an ArrayList of type Shapes and populate it with Rectangle and Circle instances.
3. Includes a method with the following signature which uses the compareShapes method to calculate the largest shape based on area in the collection.
   public static Shapes largestShape(List list)
4. Displays the largest shape in the collection.

Code for the superclass Shapes:

```
package assignemt6;

abstract class Shapes implements ShapesRelate {
	//Declaring a protected
	protected double area;

	// Constructor
	public Shapes() {

	}

	//Getter
	public double getArea() {
		return area;

	}

	// abstract class
	public abstract void calculateArea();

	// implementation of the interface method to compare the current shape
area being evaluated with the previous one
	public int compareShapes(ShapesRelate f) {

		Shapes shape = (Shapes) f;

		if (this.getArea() > shape.getArea()) {

			return 1;
```

```
                } else if (this.getArea() < shape.getArea()) {

                        return -1;

                } else {

                        return 0;

                }

        }

} // end of class
```

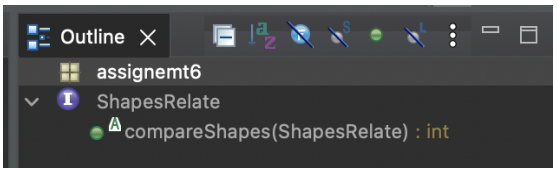
Shapes Outline:



Code for the interface ShapesRelate:

```
package assignemt6;

public interface ShapesRelate {

        // Method is public and abstract and will be implemented and
        //instantiated in the subclasses
        public int compareShapes(ShapesRelate f);
} //end of class
```

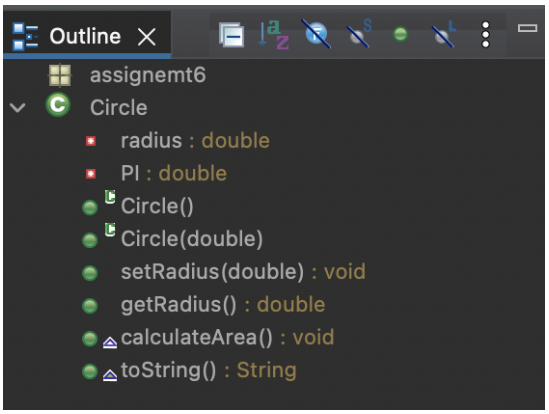
ShapesRelate Outline:



Code for the subclass Circle:

```
package assignemt6;

//This class will call the methods available in Shapes class and ShapesRelate
interface
public class Circle extends Shapes implements ShapesRelate {

	//Declaring private variables
	private double radius;
	private double PI=Math.PI;

	// Constructor
	public Circle() {
		super();
	}

	// Overloaded constructor
	public Circle(double radius) {
		this.radius = radius;

	}

	//Getter and setter
	public void setRadius(double r) {
		radius = r;
	}

	public double getRadius() {
		return radius;
	}

	//Overriding and instantiating the super abstract class
	public void calculateArea() {

		area = PI * (getRadius() * getRadius());
	}


	//Overriding toString() to print the formatted text
	public String toString() {
		return "The largest shape is the Circle of radius= " + radius;
	}

}// end of class
```

Circle Outline:

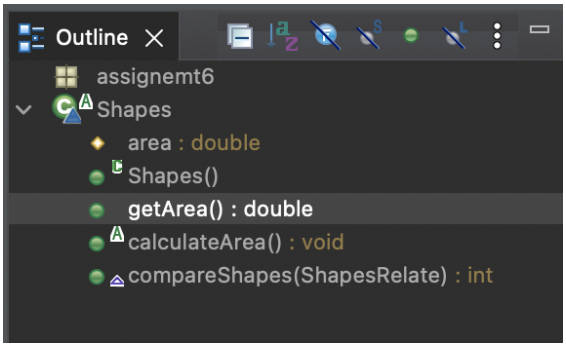


Code for the subclass Rectangle:

```
package assignemt6;

//This class will call the methods available in Shapes class and ShapesRelate interface
public class Rectangle extends Shapes implements ShapesRelate {

	//Declaring private variables
	private double length;
	private double width;


	// Constructor
	public Rectangle() {
		super();
	}

	// Overloaded constructor
	public Rectangle(double lenght, double width) {
		this.length= lenght;
		this.width= width;

	}

	// Getters and setters
	public void setWidth(double width) {
		this.width = width;
	}

	public void setLength(double lenght) {
		this.length = lenght;
	}


	public double getLength() {
		return length;
```

```
        }

        public double getWidth() {
                return width;
        }

        //Overriding and instantiating the abstract class
        public void calculateArea() {

                area = getLength() * getWidth();
        }


        //Overriding toString() to print the formatted text
        public String toString() {
                return "The largest shape is the Rectangle of length= " + length + "
and width= " + width;
        }

} //end of class
```

Rectangle Outline:



Code for the driver ShapesDriver:

```
package assignemt6;

import java.util.ArrayList;

public class ShapesDriver {

        public static void main(String[] args) {

                // Creates an ArrayList of type Shapes
                ArrayList<Shapes> shapesArray = new ArrayList<>();

                // Populates the array with Rectangles and Circle objects
```

```
            shapesArray.add(new Rectangle(4, 3));
            shapesArray.add(new Rectangle(3, 4));
            shapesArray.add(new Rectangle(5, 6));
            shapesArray.add(new Circle(5));
            shapesArray.add(new Circle(6));
            shapesArray.add(new Circle(7));

            // Calls the largestShape method
            largestShape(shapesArray);

        } //end of main()

        // Method largestShape that uses the method calculateArea() to provide
each shape area value
        public static Shapes largestShape(ArrayList<Shapes> shapesArray) {
            Shapes largest = shapesArray.get(0);

            for (int i = 0; i < shapesArray.size(); i++) {
                shapesArray.get(i).calculateArea();
                if ((shapesArray.get(i).compareShapes(largest)) > 0) {
                    largest = shapesArray.get(i);
                }
            }
            // Displays the largest shape in the collection
            System.out.println(largest);
            return largest;

        }

} // end of class
```

Console Output: