Jason Mejia 10/15/2019
Professor Auda
CSC221

Project #2 Report

Objective:

This project is a continuation from the first project. In the first project we created a pattern of rectangles and ovals. For this one the pattern of shapes are going to a polygon of 5 shapes(pentagon) and circles. Also, executing interfaces such as MyPositionInterface, and MyShapePositionInterface. Finally, adding the classes MyPolygon and MyCircle.

Solution:

First adding the two new classes that extend MyShape, which are MyPolygon and MyCircle. These two would create the patterns that are required for the end result. The working with the interfaces was relatively difficult in this project. The way I have used BoundingBox in the project was implementing it within each of the classes. The doOverlap was the most challenging part of the project. For writing in doOverlap() methods in the classes that told you if two objects of the same type overlapped such as two lines.

Code:

MyPositionInterface.java

```java
package Shape;

public interface MyPositionInterface {

		String getPoint();
		void moveTo(double x, double y);
		double distanceTo(double x2, double y2);

	}
```

MyShapePositionInterface.java

```java
package Shape;

public interface MyShapePositionInterface extends MyPositionInterface{
		MyPolygon getBoundingBox();
		boolean doOverlap();
	}
```

MyShape.java

```java
package Shape;

import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;

public abstract class MyShape implements MyShapePositionInterface {

   static boolean doOverLap = false;
   protected double x, y;
   protected Color color;

   public double getX() {
       return x;
   }

   public double getY(){
       return y;
   }

   public Color getColor() {
       return color;
   }

   public void setX(float x) {
       this.x = x;
   }

   public void setY(float y) {
       this.y = y;
   }

   public void setColor(Color color) {
       this.color = color;
   }

   //Methods to be overridden:

   public abstract void draw(GraphicsContext gc);
   public abstract String toString();


   // Interface Methods:

   @Override
   public String getPoint() {
       return "(" + getX() + ", " + getY() + ")";
   }

   @Override
   public void moveTo(double x, double y) {
       this.x = x;
       this.y = y;
   }
```

```
    @Override
    public double distanceTo(double x2, double y2) {
        return Math.sqrt(((x2 - x)*(x2 - x))+((y2 - y)*(y2 - y)));
    }

    @Override
    public MyPolygon getBoundingBox() {
        return null;
    }

    public boolean doOverlap() {
        return false;
    }


}
```

MyLine.java

```
package Shape;

import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;
import java.awt.geom.Line2D;

public class MyLine extends MyShape {
    public MyLine(double x1, double y1, double x2, double y2, Color color) {
        this.x = x1;
        this.y = y1;
        this.x2 = x2;
        this.y2 = y2;
        this.color = color;
    }

    @Override
    public void draw(GraphicsContext gc) {
        gc.setStroke(color);
        gc.strokeLine(x, y, x2, y2);
    }

    @Override
    public String toString() {
        double length = Math.sqrt(((x2 - x) * (x2 - x)) * ((y2 - y) * (y2 - y)));
        double theta = (Math.atan2((y2 - y), (x2 - x)))/(Math.PI/180);
        return "Length: " + length + "px, Angle: " + theta;
    }

    public boolean doOverlap(MyLine b) {
        return Line2D.linesIntersect(this.x, this.y, this.x2, this.y2, b.x, b.y, b.x2,
b.y2);
    }

    private double x2, y2;
}
```

MyOval.java

```java
package Shape;

import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;

public class MyOval extends MyShape {

   protected double width, height, radius;

   public MyOval(){}
   public MyOval(double originX, double originY, double width, double height, Color
color) {
       this.x = originX - (width / 2);
       this.y = originY - (height / 2);
       this.height = height;
       this.width = width;
       this.color = color;
       this.radius = Math.min(width, height);
   }

   @Override
   public void draw(GraphicsContext gc) {
       gc.setFill(color);
       gc.fillOval(x, y, width, height);
   }

   @Override
   public String toString() {
       return "Center: (" + x + ", " + y + ") " +
               "Width: " + width +
               "Height: " + height;
   }

   @Override
   public MyPolygon getBoundingBox() {
       MyPolygon poly = new MyPolygon(x, y, 5, height, color, color);
       return poly;
   }



}
```

MyRectangle.java

```java
package Shape;

import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;

public class MyRectangle extends MyShape {
   private double width, height;
   public MyRectangle(double originX, double originY, double width, double height,
Color color) {
       this.x = originX - (width / 2);
       this.y = originY - (height / 2);
       this.width = width;
       this.height = height;
       this.color = color;
   }

   @Override
   public void draw(GraphicsContext gc) {
       gc.setFill(color);
       gc.fillRect(x, y, width, height);
   }

   @Override
   public String toString() {
       return "Center: (" + x + ", " + y + ") " +
               "Width: " + width +
               "Height: " + height +
               "Area: " + width * height;
   }

   @Override
   public MyPolygon getBoundingBox() {
       MyPolygon rec = new MyPolygon(x, y, 5, height, color, color );
       return rec;
   }


}
```

MyCircle.java

```
package Shape;

import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;
import javafx.scene.shape.ArcType;

public class MyCircle extends MyOval {
   public MyCircle(double x, double y, double radius, Color color) {
       this.x = x - radius;
       this.y = y - radius;
       this.color = color;
       this.radius = radius;
       this.width = radius * 2;
       this.height = radius * 2;

   }

   public void setRadius(double radius) {
       this.radius = radius;
   }

   public double getRadius() {
       return radius;
   }

   @Override
   public void draw(GraphicsContext gc) {
       //super.draw(gc);
          gc.setFill(color);
       gc.fillArc(x, y, radius * 2, radius * 2, 0f, 360, ArcType.OPEN);
   }

   @Override
   public String toString() {
       double perimeter = 2 * Math.PI * radius;
       double area = Math.PI * (radius * radius);
       return "Radius: " + radius + " Perimeter: " + perimeter + " Area: " + area;
   }

   //private double radius;

   public boolean doOverlap(MyOval b) {
       if (distanceTo(b.getX(), b.getY()) < (this.radius + b.radius))
           return true;
       return false;
   }

}
```

MyPolygon.java

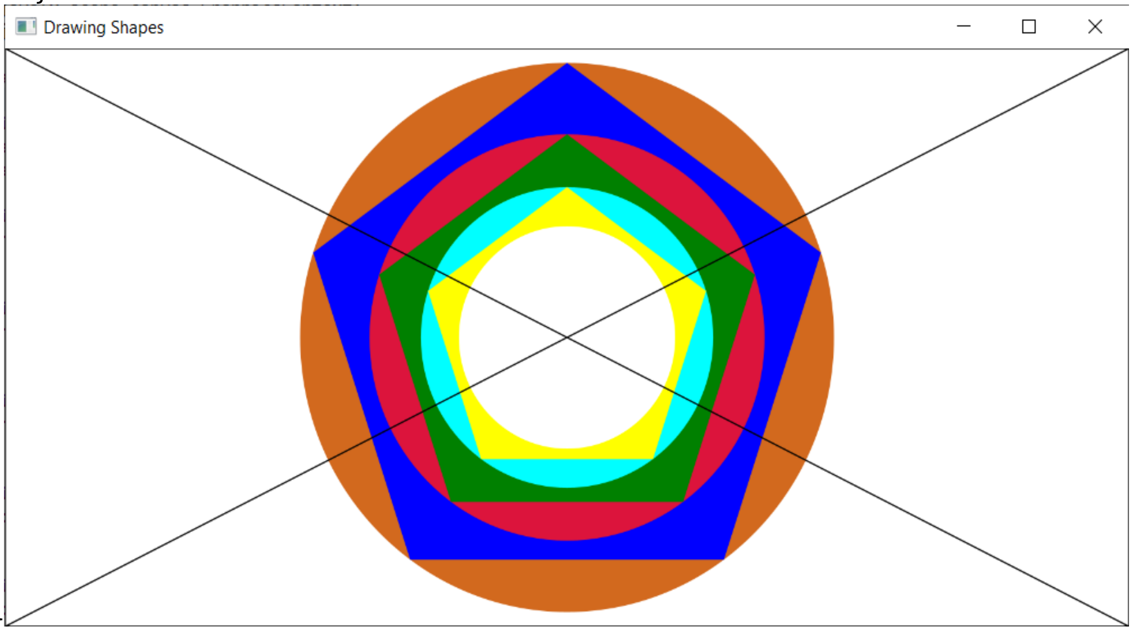
```
package Shape;

import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;

public class MyPolygon extends MyShape {

   private int number_sides;
   private double radius;
   private Color color2;
   private double sideLength;

   public MyPolygon(double x, double y, int number_sides, double radius, Color color,
Color color2) {
       this.x = x;
       this.y = y;
       this.color = color;
       this.color2 = color2;
       this.number_sides = number_sides;
       this.radius = radius;
   }

   @Override
   public void draw(GraphicsContext gc) {
       MyCircle outerCircle = new MyCircle(x, y, radius, color2);
       double[] pointsX = new double[number_sides];
       double[] pointsY = new double[number_sides];

       double angleStep = Math.PI * 2 / number_sides;
       double angle = 0;
       for (int i = 0; i < number_sides; i++, angle += angleStep) {

           pointsX[i] = -Math.sin(angle) * radius + x; // x coordinate of the point,
the negative rotates the shape to the correct orientation
           pointsY[i] = -Math.cos(angle) * radius + y;// y coordinate of the point
       }

       outerCircle.draw(gc);
       gc.setFill(color);
       gc.fillPolygon(pointsX, pointsY, number_sides);
   }

   @Override
   public String toString() {
       sideLength = 2 * radius * Math.tan(Math.PI / number_sides);
       return  "Side length: " + sideLength +
               " Interior angle: " + (180 - (Math.PI * 2 / number_sides) /
(Math.PI/180)) +
               " Perimeter: " + sideLength * number_sides +
               " Area: " + ((radius * radius) * number_sides * Math.sin(360 /
number_sides)) / 2;
   }
```

```
    private boolean valueInRange(double value, double min, double max) {
        return (value >= min) && (value <= max);
    }

    public boolean doOverlap(MyPolygon B) {
        boolean xOverlap = valueInRange(this.x, B.x, B.x + B.radius) ||
                valueInRange(B.x, this.x, this.x + this.radius);

        boolean yOverlap = valueInRange(this.y, B.y, B.y + B.radius) ||
                valueInRange(B.y, this.y, this.y + this.radius);

        return xOverlap && yOverlap;
    }


}
```

DrawDesign.java

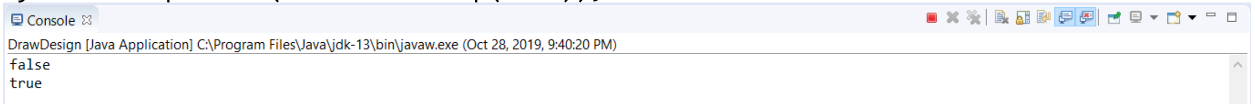
```
import Shape.*;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;

public class DrawDesign extends Application {

      private Color[] colorsArray = {Color.BLUE, Color.GREEN, Color.YELLOW,
Color.BLUEVIOLET,
                 Color.RED, Color.AQUA, Color.CRIMSON, Color.CHOCOLATE};

      public void xWithLines(GraphicsContext gc) {
             double height = gc.getCanvas().getHeight();
             double length = gc.getCanvas().getWidth();

             MyLine line1 = new MyLine(0, 0, 0, height, Color.BLACK);
             MyLine line2 = new MyLine(0, 0, length, 0, Color.BLACK);
             MyLine line3 = new MyLine(0, 0, length, height, Color.BLACK);
             MyLine line4 = new MyLine(length, 0, length, height, Color.BLACK);
             MyLine line5 = new MyLine(length, 0, 0, height, Color.BLACK);
             MyLine line6 = new MyLine(0, height, length, height, Color.BLACK);

             line1.draw(gc);
             line2.draw(gc);
             line3.draw(gc);
             line4.draw(gc);
             line5.draw(gc);
             line6.draw(gc);
         }

      public void polygonsCircles(GraphicsContext gc) {
             double length = gc.getCanvas().getWidth();
```

```
            double height = gc.getCanvas().getHeight();
            double centerX = length / 2;
            double centerY = height / 2;
            double radius;
            int colorPick = 0, colorPick2 = 7;

            if(length <= height) {
                radius = (length / 2) * .95;
            }
            else radius = (height / 2) * .95;

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

                MyPolygon poly = new MyPolygon(centerX, centerY, 5, radius,
colorsArray[colorPick], colorsArray[colorPick2] );
                poly.draw(gc);
                radius = radius - .26 * radius;
                colorPick ++;
                colorPick2 --;
            }
            MyCircle innerCircle = new MyCircle(centerX, centerY, radius,
Color.WHITE);
            innerCircle.draw(gc);


       }


      public void rectangleOvals(GraphicsContext gc) {
            double canvasLength = gc.getCanvas().getWidth();
            double canvasHeight = gc.getCanvas().getHeight();
            double centerX = canvasLength / 2;
            double centerY = canvasHeight / 2;
            double height = canvasHeight * .66;
            double width = canvasLength * .66;
            int colorPick = 0, colorPick2 = 7;

            for(int i = 0; i < 3; i ++) {
                MyRectangle rect = new MyRectangle(centerX, centerY, width, height,
colorsArray[colorPick]);
                rect.draw(gc);
                MyOval oval = new MyOval(centerX, centerY, width, height,
colorsArray[colorPick2]);
                oval.draw(gc);
                height = height * .702;
                width = width * .702;
                colorPick ++;
                colorPick2 --;
            }
        }


        @Override
        public void start(Stage stage)
        {
             Canvas canvas = new Canvas(800, 400);
             GraphicsContext gc = canvas.getGraphicsContext2D();

             DrawDesign x = new DrawDesign();
             x.polygonsCircles(gc);
             x.xWithLines(gc);


             //Testing doOverlap() function

             MyCircle cir1 = new MyCircle(100, 100, 100, Color.GREEN);
             MyCircle cir2 = new MyCircle(210, 210, 100, Color.GREEN);
             MyPolygon r1 = new MyPolygon(300, 300, 5, 200, Color.AQUA, Color.RED);
             MyPolygon r2 = new MyPolygon(650, 300, 5, 200, Color.BLUE,
Color.YELLOW);
             System.out.println(r1.doOverlap(r2));
             System.out.println(cir1.doOverlap(cir2));


             Pane root = new Pane();
             root.getChildren().add(canvas);
             Scene scene = new Scene(root);
             stage.setScene(scene);
             stage.setTitle("Drawing Shapes");
             stage.show();
       }
```



```
      }
```

```
MyCircle cir1 = new MyCircle(100, 100, 100, Color.GREEN);
MyCircle cir2 = new MyCircle(210, 210, 100, Color.GREEN);
MyPolygon r1 = new MyPolygon(300, 300, 5, 200, Color.AQUA, Color.RED);
MyPolygon r2 = new MyPolygon(650, 300, 5, 200, Color.BLUE, Color.YELLOW);
System.out.println(r1.doOverlap(r2));
System.out.println(cir1.doOverlap(cir2));
```



```
DrawDesign [Java Application] C:\Program Files\Java\jdk-13\bin\javaw.exe (Oct 28, 2019, 9:40:20 PM)
false
true
```