Ashraq Khan, CSc 22100: Software Design Laboratory
Professor Hesham A. Auda

Assignment 1

[1] The Problem

Create a hierarchy of Java classes as demonstrated below:

MyLine is_a MyShape:
MyPolygon is_a MyShape:
MyCircle is_a MyShape:

MyShape is the hierarchy's superclass and inherits the Java class Object. By using JavaFX graphics and the class hierarchy, the user should be able to draw a geometric configuration. For example, an image of alternating concentric circles and inscribed hexagons can be generated, as shown in the assignment sheet. The following additional requirements must be met:

- The code is applicable to canvases of variable height and width.
- The dimensions of the shapes are proportional to the smallest dimension of the canvas.
- The hexagons and circles are filled with different colors of your choice, specified through a MyColor enum reference type.

[2] Solution Methods

Class MyShape

MyShape was built to be the superclass of MyLine, MyPolygon, and MyCircle. It initializes two private integer variables that hold an x and y coordinate, as well as a private color variable. The constructor gives a value to these private variables when called. The class contains 3 getter methods and 3 setter methods. The getter methods are getX(), getY(), and getColor(), which returns the x coordinate, y coordinate, and shape color respectively. The setter methods are setX(), setY(), and setColor(), which sets the x coordinate, y coordinate, and shape color respectively. The imports are for the graphics context, which helps draw the shape, and the color.

There are two more methods: toString(), which prints out the MyShape object and its properties in a certain format, and draw(GraphicsContext gc). The draw method takes a graphics context variable as its parameter and only fills the color of the shape.

Class MyLine

MyLine is an extension of MyShape, and it starts off with 4 private integers specifically for this class. These integers represent the x and y coordinates of two points. The constructor

calls super in order to obtain a color for the line, and then continues to assign values for the coordinates of the two points of the line. The class contains 2 getter methods. The first is getLength(), which uses the distance formula and returns the distance between the two points, and the second method is get_xAngle(), which returns the angle in degrees via arc tangent formula. The imports are for the graphics context, which helps draw the shape, and the color.

There are two more methods: toString(), which prints out the MyLine object and its properties in a certain format, and draw(GraphicsContext gc). The draw method takes a graphics context variable as its parameter and draws a line between two points.

Class MyCircle

MyCircle is an extension of MyShape, and it has a private double variable for radius. The constructor sets up the center point and the color of the circle via the super call. It also sets the radius on the next line. There are 3 getter methods, each one involving the radius variable. The getArea() method returns the area of the circle using the standard pi multiplied by the radius squared. The getPerimeter() method returns the perimeter of the circle by multiplying the radius by 2 times pi. The getRadius() method simply returns the radius. The imports are for the graphics context, which helps draw the shape, and the color.

There are two more methods: toString(), which prints out the MyCircle object and its properties in a certain format, and draw(GraphicsContext gc). The draw method takes a graphics context variable as its parameter and fills the circle a certain color.

Class MyPolygon

MyPolygon is an extension of MyShape, and it goes more in depth than the other classes. It has 4 private variables: one double variable for the radius, one integer variable for the number of sides, and two arrays to store x and y coordinates. The constructor starts off with calling super in order to have a center point as well as a color for the polygon. Furthermore, the radius and sides are set accordingly, and two arrays are created to store the x and y coordinates respectively for each corner of the polygon. The angle is determined from dividing 2 times pi by the number of sides. By using this angle, the points around the polygon can be determined by using the appropriate trigonometric formulas. The imports are for the graphics context, which helps draw the shape, and the color.

There are 6 getter methods, each returning a double value that describes a property of the polygon. The first is getRadius(), which simply returns the distance from the center point to one corner of the polygon. Next is getArea(), which returns an area that was calculated using a formula that works for every normal polygon. The method getPerimeter() multiplies the side length by the number of sides, and returns the perimeter. The following method, getApothem(), returns the apothem, which is the distance from the center point to the middle of one side of the polygon. This is calculated by multiplying the radius by the cosine of 180 degrees divided by the

number of sides. A simple getAngle() method returns the angle between two corners of the polygon. The final method, getSide(), returns the length of one side of the polygon. This is found by multiplying 2 times the radius by the sin of the angle between two corners of the polygon.

There are two more methods: toString(), which prints out the MyPolygon object and its properties in a certain format, and draw(GraphicsContext gc). The draw method takes a graphics context variable as its parameter and fills the polygon a certain color. It also scribes each side of the polygon and creates a perimeter that is the same color as the inside of the polygon.

Class Main

Main is where objects from the other classes are created and drawn. There are multiple imports used here. The first import is javafx.application.Application, which is used to create a separate extended application that holds the graphics all together. Next is javafx.scene.Scene, which holds the content inside the window. The import javafx.stage.Stage is the window itself with functions such as minimize, maximize, resize, and close. Next is the import javafx.scene.canvas.Canvas, which is basically the drawing board. The graphics context comes from javafx.scene.canvas.GraphicsContext, and it behaves like a pen that draws and fills shapes, lines, and colors. The final import is javafx.scene.layout.StackPane, which is a layout for the elements that allows them to be stacked on top of each other instead of horizontally or vertically.

There are two final integer values in the beginning of the class, which are for determining the window width and height. These can be changed to whatever values the user likes, as long as it is an integer. A stage is set up with the title “Assignment 1”, as well as a canvas for the graphics context to draw on. The line width of the graphics context was set for 3 just because of preference.

A series of objects were made to mimic the figure shown on the assignment sheet. Alternating polygons and circles of different colors were made, each one using the radius or apothem of the shape before it. This was to insure that each shape was inscribed with the one outside of it. Lastly, two diagonal black lines were drawn on top of the sequence of shapes to complete the picture. The properties of each object were printed via their respective toString() methods.

Enum MyColor

MyColor imports javafx.scene.paint.Color in order to pass custom RGBA (red, green, blue, and opacity) values and obtain an actual value for the color in hexadecimal. The enum has multiple predefined colors ranging across the spectrum, each having its own unique RGBA values. The private integers r, g, b, a hold values for red, green, blue, and the opacity respectively. The constructor takes in custom values that will be assigned to the private variables. The function getCol() returns a color by using the Color.rgb method from

javafx.scene.paint.Color. An example of this being used is executing MyColor.CYAN.getCol(), which will return the color for cyan.

[3] Codes Developed

----- MyShape.java -----

```
/* -------------------------------------------------------------
Class MyShape is the hierarchy's superclass and inherits the
Java class Object. An implementation of the class defines a
reference point (x, y) and the color of the shape.
------------------------------------------------------------- */
package sample;

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

public class MyShape {
   private int x, y; // Point of shape in pixels
   private Color color;

   MyShape(int x, int y, Color color){
       this.x = x;
       this.y = y;
       this.color = color;
   }

   public int getX(){ return x; }
   public int getY(){ return y; }
   public Color getColor(){ return color; }
   public void setX(int x){ this.x = x; }
   public void setY(int y){ this.y = y; }
   public void setColor(Color color){ this.color = color; }

   public String toString(){
       return String.format("----- Shape Properties -----\n%15s (%d,%d)\n%15s "
+ getColor(),
               "Start Point:",x,y,"Color:");
   }
   public void draw(GraphicsContext gc){
       gc.setFill(color);
       gc.fill();
   }
}
```

----- MyLine.java -----

```
/* -------------------------------------------------------------
Class MyLine inherits class MyShape. The MyLine object is a
straight line defined by the endpoints (x1, y1) and (x2, y2).
The MyLine object may be of any color.
```

```
------------------------------------------------------------ */
package sample;

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

public class MyLine extends MyShape {
   private int x1, y1, x2, y2; // Coordinates of two points of line

   MyLine(int x1, int x2, int y1, int y2, Color color){
       super(0, 0, color);
       this.x1 = x1;
       this.x2 = x2;
       this.y1 = y1;
       this.y2 = y2;
   }

   public double getLength(){
       return Math.sqrt(Math.pow(x2-x1,2) + Math.pow(y2-y1,2));
   }
   public double get_xAngle(){
       return Math.toDegrees(Math.atan((y2-y1)/(x2-x1)));
   }

   @Override
   public String toString(){
       return String.format("----- Line Properties -----\n%15s (%d,%d)\n%15s
(%d,%d)\n%15s %f\n%15s %f\n%15s "+super.getColor(),
               "Point 1:",x1,y1,"Point 2:",x2,y2,"Line
Length:",getLength(),"Angle:",get_xAngle(),"Color:");
   }
   @Override
   public void draw(GraphicsContext gc){
       gc.setStroke(super.getColor());
       gc.strokeLine(x1,y1,x2,y2);
   }
}
```

----- MyCircle.java -----

```
/* ------------------------------------------------------------
Class MyCircle inherits class MyShape.  The MyCircle object
is defined by its center (x, y) and radius r, and may be
filled with a color.
------------------------------------------------------------ */
package sample;

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

```
public class MyCircle extends MyShape {
   private double radius;

   MyCircle(int x, int y, double radius, Color color){
       super(x,y,color);
       this.radius = radius;
   }

   public double getArea(){
       return Math.PI * Math.pow(radius, 2);
   }
   public double getPerimeter(){
       return 2 * Math.PI * radius;
   }
   public double getRadius(){ return radius; }

   @Override
   public String toString(){
       return String.format("----- Circle Properties -----\n%15s %f\n%15s
%f\n%15s %f\n%15s "+super.getColor(),

"Radius:",getRadius(),"Area:",getArea(),"Perimeter:",getPerimeter(),"Color:");
   }
   @Override
   public void draw(GraphicsContext gc){
       gc.setFill(super.getColor());
       gc.fillOval(super.getX() - radius, super.getY() - radius, 2 * radius, 2
* radius);
       //gc.strokeRect(super.getX() - radius, super.getY() - radius, 2 *
radius, 2 * radius);
   }
}
```

----- MyPolygon.java -----

```
/* ------------------------------------------------------------
Class MyPolygon inherits class MyShape. The MyPolygon object
is a regular polygon defined by the integer parameter, N—
the number of the polygon's equal side lengths and equal
interior angles, and the center (x, y) and radius, r, of the
circle in which it is inscribed. The MyPolygon object may
be filled with a color.
------------------------------------------------------------ */
package sample;

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

public class MyPolygon extends MyShape {
   private double radius;
```

```
   private int sides;
   double xp[];
   double yp[];

   MyPolygon(int x, int y, double radius, int sides, Color color){
       super(x,y,color);
       this.radius = radius;
       this.sides = sides;
       xp = new double[sides];
       yp = new double[sides];
       double ang = (2 * Math.PI)/sides;
       for (int i = 0; i < sides; ++i){
           xp[i] = x + (radius*(-1 * Math.sin(i*ang)));
           yp[i] = y + (radius*(-1 * Math.cos(i*ang)));
       }
   }

   public double getRadius(){ return radius; }
   public double getArea(){
       return Math.pow(radius,2) * sides * Math.sin(getAngle()) * 0.5;
   }
   public double getPerimeter(){ return sides * getSide(); }
   public double getApothem(){ return radius * Math.cos(Math.toRadians(180 /
sides)); }
   public double getAngle(){ return Math.PI/sides; }
   public double getSide(){ return 2 * radius * Math.sin(Math.PI/sides); }

   @Override
   public String toString(){
       return String.format("----- Polygon Properties -----\n%15s %f\n%15s
%f\n%15s %f\n%15s %f\n%15s %f\n%15s "+super.getColor(),

"Radius:",getRadius(),"Area:",getArea(),"Perimeter:",getPerimeter(),
               "Angle:",getAngle(),"Apothem:",getApothem(),"Color:");
   }

   @Override
   public void draw(GraphicsContext gc){
       gc.setFill(super.getColor());
       gc.setStroke(super.getColor());
       gc.setLineWidth(3);
       gc.strokePolygon(xp, yp, sides);
       gc.fillPolygon(xp, yp, sides);
   }

   public void border(GraphicsContext gc){
       gc.setStroke(Color.BLACK);
       for (int i = 1; i < sides; ++i){
           gc.strokeLine(xp[i-1], yp[i-1], xp[i], yp[i]);
```

```
        }
        gc.strokeLine(xp[sides-1], yp[sides-1], xp[0], yp[0]);
    }
}
```

----- Main.java -----

```
/* Ashraq Khan // CS 22100 Assignment 1 // 09/21/2020 */

package sample;

/* Imports */
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

/* Driver class */
public class Main extends Application {
    // Sets boundaries of stage
    final int WIDTH = 1000;
    final int HEIGHT = 600;

    @Override
    public void start(Stage primaryStage) {
        try {
            // Sets up primary stage, canvas, and graphics context
            primaryStage.setTitle("Assignment 1");
            Canvas canvas = new Canvas(WIDTH, HEIGHT);
            GraphicsContext gc = canvas.getGraphicsContext2D();
            gc.setLineWidth(3);

            // Prints alternate hexagons and circles of different colors
            MyPolygon hexGray = new MyPolygon(WIDTH/2,HEIGHT/2,
200,6,MyColor.GREY.getCol());
            hexGray.draw(gc);
            MyCircle circleY = new MyCircle(WIDTH/2,HEIGHT/2,
hexGray.getApothem(), MyColor.YELLOW.getCol());
            circleY.draw(gc);
            MyPolygon hexGreen = new MyPolygon(WIDTH/2,HEIGHT/2,
circleY.getRadius(), 6,MyColor.LIME.getCol());
            hexGreen.draw(gc);
            MyCircle circleP = new MyCircle(WIDTH/2,HEIGHT/2,
hexGreen.getApothem(), MyColor.HOTPINK.getCol());
            circleP.draw(gc);
            MyPolygon hexBlue = new MyPolygon(WIDTH/2,HEIGHT/2,
circleP.getRadius(),6,MyColor.CYAN.getCol());
            hexBlue.draw(gc);
```

```
            hexGray.border(gc);

            // Lists properties of all shapes
            System.out.println(hexGray + "\n\n" + circleY + "\n\n" + hexGreen +
"\n\n" + circleP + "\n\n" + hexBlue + "\n\n");

            MyLine line1 = new MyLine(0,WIDTH,0,HEIGHT,MyColor.BLACK.getCol());
            MyLine line2 = new MyLine(0,WIDTH,HEIGHT,0,MyColor.BLACK.getCol());
            line1.draw(gc);
            line2.draw(gc);
            gc.strokeRect(0, 0, WIDTH, HEIGHT);

            // Lists properties of all lines
            System.out.println(line1 + "\n\n" + line2 + "\n\n");

            // Sets stack pane and scene for image to appear on
            StackPane root = new StackPane(canvas);
            Scene scene = new Scene(root,WIDTH,HEIGHT);
            primaryStage.setScene(scene);
            primaryStage.show();


        } catch(Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

----- MyColor.java -----

```
/* ---------------------------------------------------------
Enum MyColor sets up pre-made colors of varying RGB values
and opacities. These will be used to color shapes and lines.
--------------------------------------------------------- */

package sample;

import javafx.scene.paint.Color;

public enum MyColor{
    RED(255,0,0,255), BLUE(0,0,255,255),
    LIME(0,255,0,255), CYAN(0,255,255,255),
    GREEN(0,128,0,255), GREY(128,128,128,255),
    MAGENTA(255,0,255,255), PURPLE(128,0,128,255),
    VIOLET(148,0,211,255), YELLOW(255,255,0,255),
    WHITE(255,255,255,255), BLACK(0,0,0,255),
```

```
   HOTPINK(255,105,180,255);

   private int r,g,b,a; // Value for red, green, blue, and opacity

   MyColor(int r, int g, int b, int a){ this.r = r; this.g = g; this.b = b;
this.a = a; }
   public Color getCol() { return Color.rgb(r, g, b, (double)(a/255)); }
}
```

[4] Outputs Produced