Report: Assignment:1

MYESHA MAHAZABEEN

CSC 22100, SOFTWARE DESIGN LABORATORY, 1XB [6822], SUMMER 2022

Instructor: Hesham A. Auda
Date: 06/26/2022

Department of Computer Science
The City College of CUNY

CSc 22100 1XB 6822: Software Design Laboratory Summer 2022

Assignment 1

A report uploaded on the Bloackboard's course page for the section showing:

[1] *statement of the problem,*
[2] *solution methods,*
[3] *all codes developed, and*
[4] *outputs produced for the tasks indicated,*

is due by ***11:59 pm on Sunday, 26 June 2022***. ***The deadline is strictly observed***.

1- Create a hierarchy of Java classes as follows:

MyLine *extends* **MyShape**;
MyRectangle *extends* **MyShape**;
MyOval *extends* **MyShape**.

Class MyPoint:

Class **MyPoint** is used by class **MyShape** to define the reference point, **p**(x, y), of the Java display coordinate system, as well as by all subclasses in the class hierarchy to define the points stipulated in the subclass definition. The class utilizes a color of **enum** reference type **MyColor**, and includes appropriate class constructors and methods, including methods that perform point related operations.

Enum MyColor:

Enum **MyColor** is used by class **MyShape** and all subclasses in the class hierarchy to define the colors of the shapes. The **enum** reference type defines a set of constant colors by their red, green, blue, and opacity, components, with values in the range [0 – 255]. The **enum** reference type includes a constructor and appropriate methods, including methods that return the corresponding hexadecimal representation and JavaFx Color objects of the constant colors.

Class MyShape:

Class **MyShape** is the superclass of the hierarchy, extending the Java class Object. An implementation of the class defines a reference point, **p**(x, y), of type **MyPoint**, and the color of the shape of **enum** reference type **MyColor**. The class includes appropriate class constructors and methods, including methods, including methods that perform the following operations:

a. *perimeter, area* – return, respectively, the perimeter and meter of the **MyShape** object. These methods must be overridden in each subclass in the hierarchy. For the **MyShape** object, the methods return zero.
b. *toString* – returns the object's description as a String. This method must be overridden in each subclass in the hierarchy;
c. *draw* – draws the object shape. This method must be overridden in each subclass in the hierarchy. For the **MyShape** object, it paints the drawing canvas in the color specified.

Class MyLine:

Class **MyLine** extends class MyShape. The **MyLine** object is a straight line segment defined by the endpoints $\mathbf{p}_1(x_1, y_1)$ and $\mathbf{p}_2(x_2, y_2)$ of type **MyPoint**. The **MyLine** object may be of any color of **enum** reference type **MyColor**. The class includes appropriate class constructors and methods, including methods that perform the following operations:

a. getLine — returns the **MyLine** object*;*
b. length, xAngle — returns, respectively the length and angle with the *x*-axis of the **MyLine** object;
c. perimeter, area — return, respectively, the perimeter and area of the **MyLine** object; *d. toString* — returns a string representation of the **MyLine** object, including the line's endpoints, length, and angle with the *x*-axis;
e. draw — draws a **MyLine** object.

Class MyRectangle:

Class **MyRectangle** extends class **MyShape**. The **MyRectangle** object is a rectangle of height *h* and width *w*, and a top left corner point **p**(*x*, *y*), and may be filled with a color of **enum** reference type **MyColor**. The class includes appropriate class constructors and methods, including methods that perform the following operations:

f. getTLCP, getWidth, getHeight — return, respectively, the top left corner point, width, and height of the **MyRectangle** object
g. perimeter, area — return, respectively, the perimeter and area of the **MyRectangle** object;
h. toString— returns a string representation of the **MyRectangle** object: top left corner point, width, height, perimeter, and area;
i. draw— draws a **MyRectangle** object.

Class MyOval:

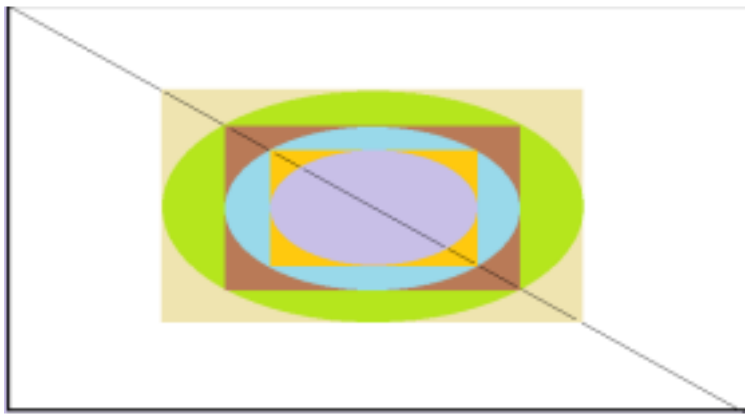
Class **MyOval** extends class **MyShape**. The **MyOval** object is defined by an ellipse within a rectangle of height *h* and width *w*, and a center point **p**(*x*, *y*). The **MyOval** object may be filled with a color of **enum** reference type **MyColor**. The class includes appropriate class constructors and methods, including methods that perform the following operations:

a. getCenter, getMinorAxis, getMajorAxis — return, respectively, the center point and semi minor and major axes of the **MyOval** object;
b. perimeter, area — return, respectively, the perimeter and area of the **MyOval** object; *c. toString*— returns a string representation of the **MyOval** object: semi minor and major axes lengths, perimeter, and area;
d. draw— draws a **MyOval** object.

2- Use JavaFX graphics and the class hierarchy to draw the geometric configuration comprised of a sequence of alternating concentric ovals and inscribed rectangles, as illustrated below, subject to the following additional requirements:

a. The code is applicable to canvases of variable height and width;
b. The dimensions of the shapes are proportional to the smallest dimension of the canvas;
c. The ovals and rectangles are filled with different colors of your choice, specified through an **enum** reference type **MyColor**.

3- Explicitly specify all the classes imported and used in your code.



Best wishes

Hesham A. Auda
06-16-2022

Solution

This project contains five classes, an enum class and a main class. Main class contains everything together to generate the desired result using all the classes. Here, MyShape acts as a super class to MyLine, MyRectangle and MyOval. And these classes extend MyShape. MyColor enum class is used by all classes except the Main class which demonstrates the color of shapes/points. And MyPoint is used by some classes to describe their coordinates.

Classes and Methods:

- **Main.java:** Uses the JavaFx library to draw shapes using graphics by calling on the appropriate stage, scene, canvas and graphics classes. The height and width of canvas are determined by variables so that they can be changed. Additionally, all the dimensions of the shapes depend on canvas' width and height so that they can automatically adjust to variable canvas' height and width.

    - Imports:
        - javafx.application.Application;
        - javafx.scene.Scene;
        - javafx.scene.canvas.Canvas;
        - javafx.scene.canvas.GraphicsContext;
        - javafx.scene.layout.StackPane;
        - javafx.stage.Stage;
        - java.io.IOException;

    - Methods:
        - **start();** method where JavaFx elements are handled
        - **main();** main function

- **MyPoint.java:** This class is used by some classes to describe their coordinates. It also has a method to calculate distance between two points and a method to calculate the angle line between two points created with the x-axis.
    - Attributes:
        - **double x, y :** represents x and y value of a coordinate.
        - **MyColor color:** describes color of a MyPoint object.

    - Methods:
        - **calculateDistance():** returns Euclidean distance between two MyPoint objects.
        - **getXAngle():** returns the angle that the line between two MyPoint objects create with the x-axis.

- **MyColor.java:** This enum class is used to describe colors of shapes and return color of a shape in appropriate format.
    - Imports:
        - javafx.scene.paint.Color;
    - Attributes:
        - **double r, g, b, opacity :** describes r, g, b and opacity of a color.
    - Methods:
        - **getHexRepresentation():** returns hexadecimal code for color specified.
        - **getColor():** returns the corresponding color value from javafx.scene.paint.Color using inserted rgb and opacity values.
- **MyShape.java:** This class is written as a superclass to MyLine, MyRectangle and MyOval class. This class contains a MyPoint object and a MyColor object to describe the shape. There are seven constructors for this class, which include a copy constructor and default constructor that sets all values to 0 and color to black. This class can be used to set the background color of the canvas.
    - Imports:
        - javafx.scene.canvas.GraphicsContext;
    - Attributes:
        - **MyPoint p:** used to keep a reference point of a MyShape object
        - **MyColor color:** is used to describe color of a MyShape object
    - Methods:
        - **perimeter():** returns 0.0 , is made to be overridden by subclasses
        - **area():** returns 0.0 , is made to be overridden by subclasses
        - **toString():** Describes a MyShape object using its MyPoint p attribute.
        - **draw():** fills the background color of the canvas with the object's color.
- **MyLine.java:** This class is a subclass of MyShape class. It is used to draw a straight line connecting two points. This class has five constructors including a copy constructor that sets the end points and color of the line.
    - Imports:
        - javafx.scene.canvas.GraphicsContext;

  - Attributes:
    - **MyPoint p:** is passed on by it's superclass, describes an end point of the line
    - **MyColor color:** is passed on by its superclass, describing the line's color.
    - **MyPoint p2:** is used to describe another endpoint of the line.

  - Methods:
    - **getLine():** returns the current line.
    - **length():** returns length of the line using MyPoint's calculateDistance method.
    - **xAngle():** returns the angle a MyLine object creates with x-axis using MyPoint's calculateAngle method
    - **perimeter():** returns MyLine's length as perimeter.
    - **area():** returns MyLine's area, which is 0.0
    - **toString():** returns a string object containing the endpoints, length, angle with x-axis, area and perimeter
    - **draw():** draws the line itself.

- **MyRectangle.java:** This class is a subclass of MyShape class. It is used to draw a rectangle of given height and width. This class has six constructors including a copy constructor that sets the value of all the attributes.

  - Imports:
    - javafx.scene.canvas.GraphicsContext;

  - Attributes:
    - **MyPoint p:** is passed on by superclass MyShape used to describe the geometric center of the rectangle.
    - **MyColor color:** is passed on by superclass MyShape and describes color of the rectangle
    - **double h, w:** describes height and width of the rectangle.

  - Methods:
    - **getHeight():** returns height (h) of the rectangle.
    - **getWidth():** returns width (w) of the rectangle.
    - **getTLCP():** returns top left corner point of a rectangle
    - **perimeter(), area():** returns perimeter and area of the rectangle.
    - **toString():** returns a String object containing top left corner, height, width, perimeter, area of the MyRectangle object.
    - **draw():** draws the MyRectangle object.

- **MyOval.java:** This class is a subclass of MyShape class. It is used to draw an oval of given height and width. This class has seven constructors including a copy constructor that set the value of all the attributes.

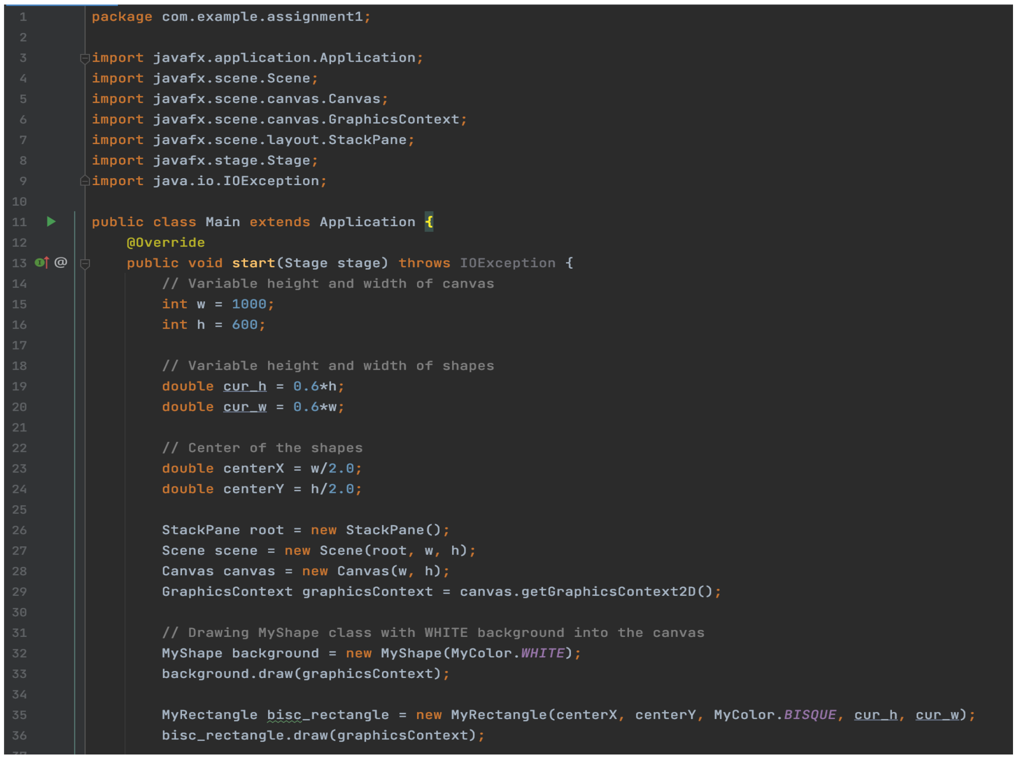
    - Imports:
        - javafx.scene.canvas.GraphicsContext;

    - Attributes:
        - **MyPoint p:** is passed on by superclass MyShape used to describe the center of the oval.
        - **MyColor color:** is passed on by superclass MyShape and describes the color of the oval.
        - **double h, w:** describes height and width of the oval.

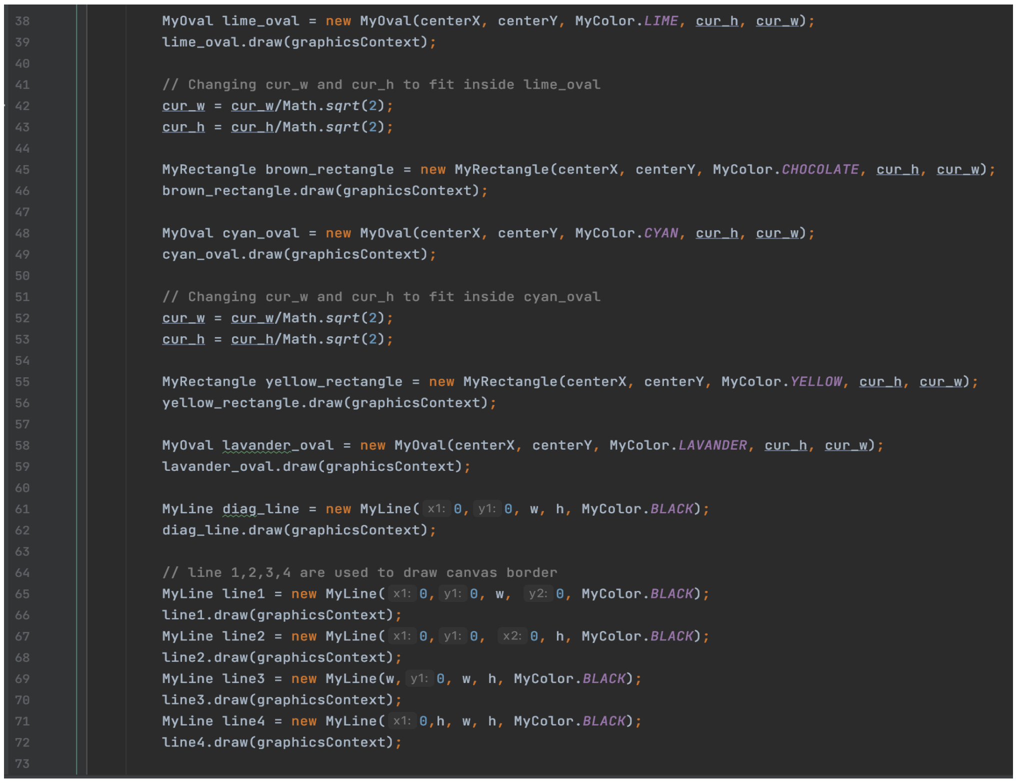
    - Methods:
        - **getMinorAxis():** returns semi minor axis length of the oval.
        - **getMajorAxis():** returns semi major axis length of the oval.
        - **perimeter(), area():** returns perimeter and area of the oval.
        - **toString():** return a String object containing semi major axis, semi minor axis length, perimeter, area of the MyOval object.
        - **draw():** draws the MyOval object.

Screenshots of Source Code:

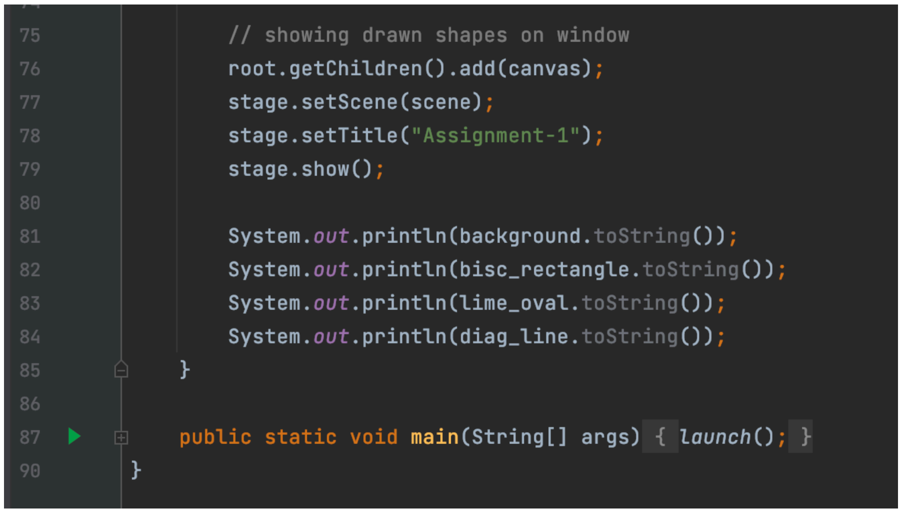
- **Main.java:**



```java
package com.example.assignment1;

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;
import java.io.IOException;

public class Main extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        // Variable height and width of canvas
        int w = 1000;
        int h = 600;

        // Variable height and width of shapes
        double cur_h = 0.6*h;
        double cur_w = 0.6*w;

        // Center of the shapes
        double centerX = w/2.0;
        double centerY = h/2.0;

        StackPane root = new StackPane();
        Scene scene = new Scene(root, w, h);
        Canvas canvas = new Canvas(w, h);
        GraphicsContext graphicsContext = canvas.getGraphicsContext2D();

        // Drawing MyShape class with WHITE background into the canvas
        MyShape background = new MyShape(MyColor.WHITE);
        background.draw(graphicsContext);

        MyRectangle bisc_rectangle = new MyRectangle(centerX, centerY, MyColor.BISQUE, cur_h, cur_w);
        bisc_rectangle.draw(graphicsContext);
```

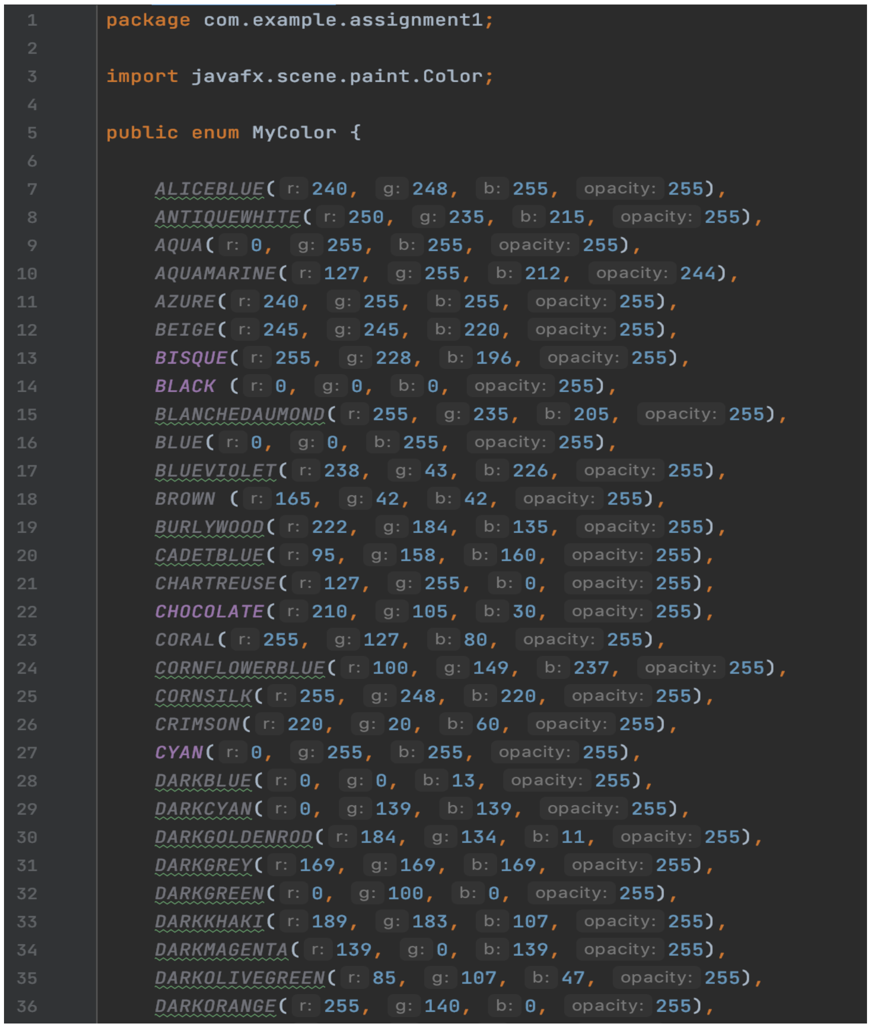


```java
        MyOval lime_oval = new MyOval(centerX, centerY, MyColor.LIME, cur_h, cur_w);
        lime_oval.draw(graphicsContext);

        // Changing cur_w and cur_h to fit inside lime_oval
        cur_w = cur_w/Math.sqrt(2);
        cur_h = cur_h/Math.sqrt(2);

        MyRectangle brown_rectangle = new MyRectangle(centerX, centerY, MyColor.CHOCOLATE, cur_h, cur_w);
        brown_rectangle.draw(graphicsContext);

        MyOval cyan_oval = new MyOval(centerX, centerY, MyColor.CYAN, cur_h, cur_w);
        cyan_oval.draw(graphicsContext);

        // Changing cur_w and cur_h to fit inside cyan_oval
        cur_w = cur_w/Math.sqrt(2);
        cur_h = cur_h/Math.sqrt(2);

        MyRectangle yellow_rectangle = new MyRectangle(centerX, centerY, MyColor.YELLOW, cur_h, cur_w);
        yellow_rectangle.draw(graphicsContext);

        MyOval lavander_oval = new MyOval(centerX, centerY, MyColor.LAVANDER, cur_h, cur_w);
        lavander_oval.draw(graphicsContext);

        MyLine diag_line = new MyLine( 0, 0, w, h, MyColor.BLACK);
        diag_line.draw(graphicsContext);

        // line 1,2,3,4 are used to draw canvas border
        MyLine line1 = new MyLine( 0, 0, w,  0, MyColor.BLACK);
        line1.draw(graphicsContext);
        MyLine line2 = new MyLine( 0, 0,  0, h, MyColor.BLACK);
        line2.draw(graphicsContext);
        MyLine line3 = new MyLine(w, 0, w, h, MyColor.BLACK);
        line3.draw(graphicsContext);
        MyLine line4 = new MyLine( 0,h, w, h, MyColor.BLACK);
        line4.draw(graphicsContext);

```

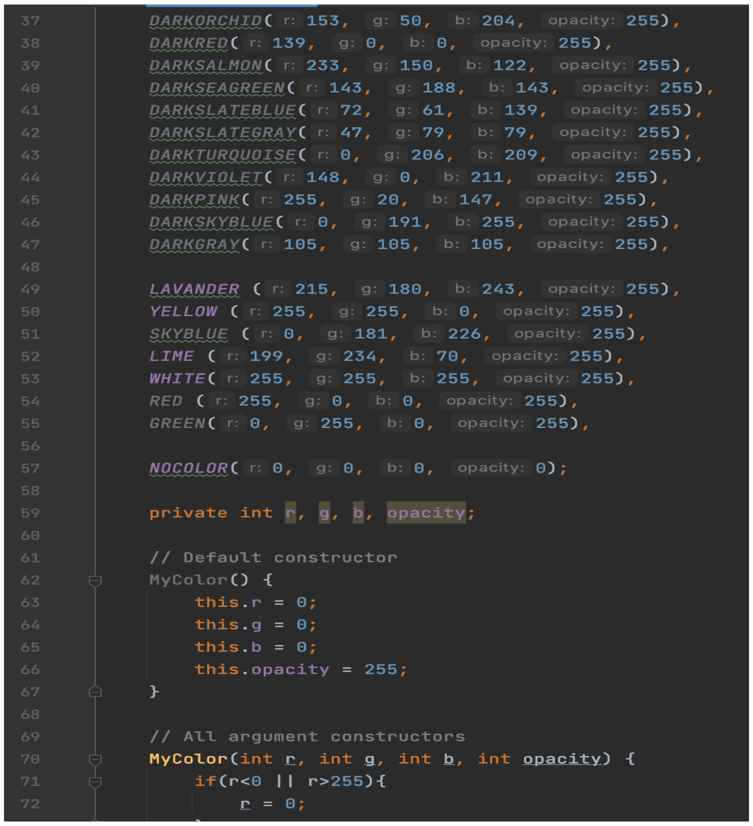


```java
        // showing drawn shapes on window
        root.getChildren().add(canvas);
        stage.setScene(scene);
        stage.setTitle("Assignment-1");
        stage.show();

        System.out.println(background.toString());
        System.out.println(bisc_rectangle.toString());
        System.out.println(lime_oval.toString());
        System.out.println(diag_line.toString());
    }

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

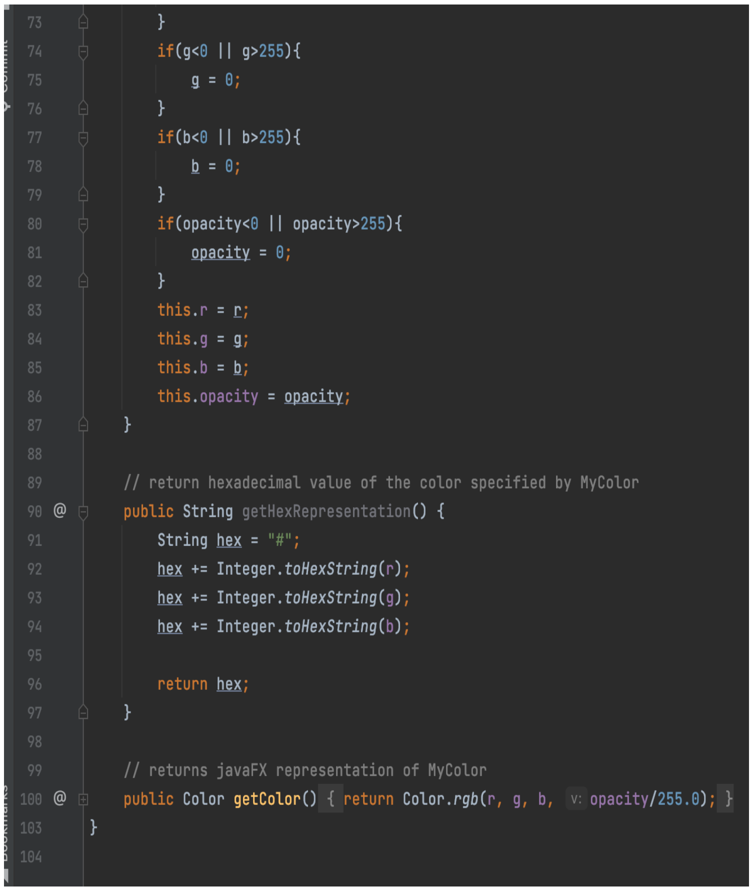
- **MyColor.java:**



```java
package com.example.assignment1;

import javafx.scene.paint.Color;

public enum MyColor {

    ALICEBLUE( r: 240, g: 248, b: 255, opacity: 255),
    ANTIQUEWHITE( r: 250, g: 235, b: 215, opacity: 255),
    AQUA( r: 0, g: 255, b: 255, opacity: 255),
    AQUAMARINE( r: 127, g: 255, b: 212, opacity: 244),
    AZURE( r: 240, g: 255, b: 255, opacity: 255),
    BEIGE( r: 245, g: 245, b: 220, opacity: 255),
    BISQUE( r: 255, g: 228, b: 196, opacity: 255),
    BLACK ( r: 0, g: 0, b: 0, opacity: 255),
    BLANCHEDAUMOND( r: 255, g: 235, b: 205, opacity: 255),
    BLUE( r: 0, g: 0, b: 255, opacity: 255),
    BLUEVIOLET( r: 238, g: 43, b: 226, opacity: 255),
    BROWN ( r: 165, g: 42, b: 42, opacity: 255),
    BURLYWOOD( r: 222, g: 184, b: 135, opacity: 255),
    CADETBLUE( r: 95, g: 158, b: 160, opacity: 255),
    CHARTREUSE( r: 127, g: 255, b: 0, opacity: 255),
    CHOCOLATE( r: 210, g: 105, b: 30, opacity: 255),
    CORAL( r: 255, g: 127, b: 80, opacity: 255),
    CORNFLOWERBLUE( r: 100, g: 149, b: 237, opacity: 255),
    CORNSILK( r: 255, g: 248, b: 220, opacity: 255),
    CRIMSON( r: 220, g: 20, b: 60, opacity: 255),
    CYAN( r: 0, g: 255, b: 255, opacity: 255),
    DARKBLUE( r: 0, g: 0, b: 13, opacity: 255),
    DARKCYAN( r: 0, g: 139, b: 139, opacity: 255),
    DARKGOLDENROD( r: 184, g: 134, b: 11, opacity: 255),
    DARKGREY( r: 169, g: 169, b: 169, opacity: 255),
    DARKGREEN( r: 0, g: 100, b: 0, opacity: 255),
    DARKKHAKI( r: 189, g: 183, b: 107, opacity: 255),
    DARKMAGENTA( r: 139, g: 0, b: 139, opacity: 255),
    DARKOLIVEGREEN( r: 85, g: 107, b: 47, opacity: 255),
    DARKORANGE( r: 255, g: 140, b: 0, opacity: 255),
```

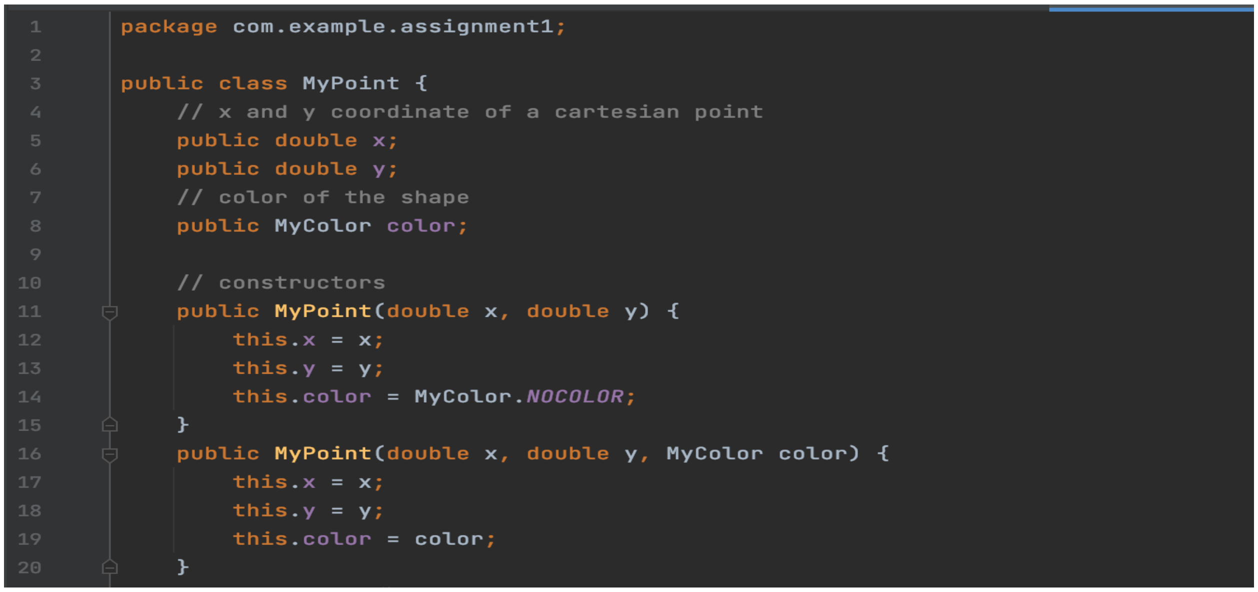


```
    DARKORCHID( r: 153, g: 50, b: 204, opacity: 255),
    DARKRED( r: 139, g: 0, b: 0, opacity: 255),
    DARKSALMON( r: 233, g: 150, b: 122, opacity: 255),
    DARKSEAGREEN( r: 143, g: 188, b: 143, opacity: 255),
    DARKSLATEBLUE( r: 72, g: 61, b: 139, opacity: 255),
    DARKSLATEGRAY( r: 47, g: 79, b: 79, opacity: 255),
    DARKTURQUOISE( r: 0, g: 206, b: 209, opacity: 255),
    DARKVIOLET( r: 148, g: 0, b: 211, opacity: 255),
    DARKPINK( r: 255, g: 20, b: 147, opacity: 255),
    DARKSKYBLUE( r: 0, g: 191, b: 255, opacity: 255),
    DARKGRAY( r: 105, g: 105, b: 105, opacity: 255),

    LAVANDER ( r: 215, g: 180, b: 243, opacity: 255),
    YELLOW ( r: 255, g: 255, b: 0, opacity: 255),
    SKYBLUE ( r: 0, g: 181, b: 226, opacity: 255),
    LIME ( r: 199, g: 234, b: 70, opacity: 255),
    WHITE( r: 255, g: 255, b: 255, opacity: 255),
    RED ( r: 255, g: 0, b: 0, opacity: 255),
    GREEN( r: 0, g: 255, b: 0, opacity: 255),

    NOCOLOR( r: 0, g: 0, b: 0, opacity: 0);

    private int r, g, b, opacity;

    // Default constructor
    MyColor() {
        this.r = 0;
        this.g = 0;
        this.b = 0;
        this.opacity = 255;
    }

    // All argument constructors
    MyColor(int r, int g, int b, int opacity) {
        if(r<0 || r>255){
            r = 0;
```

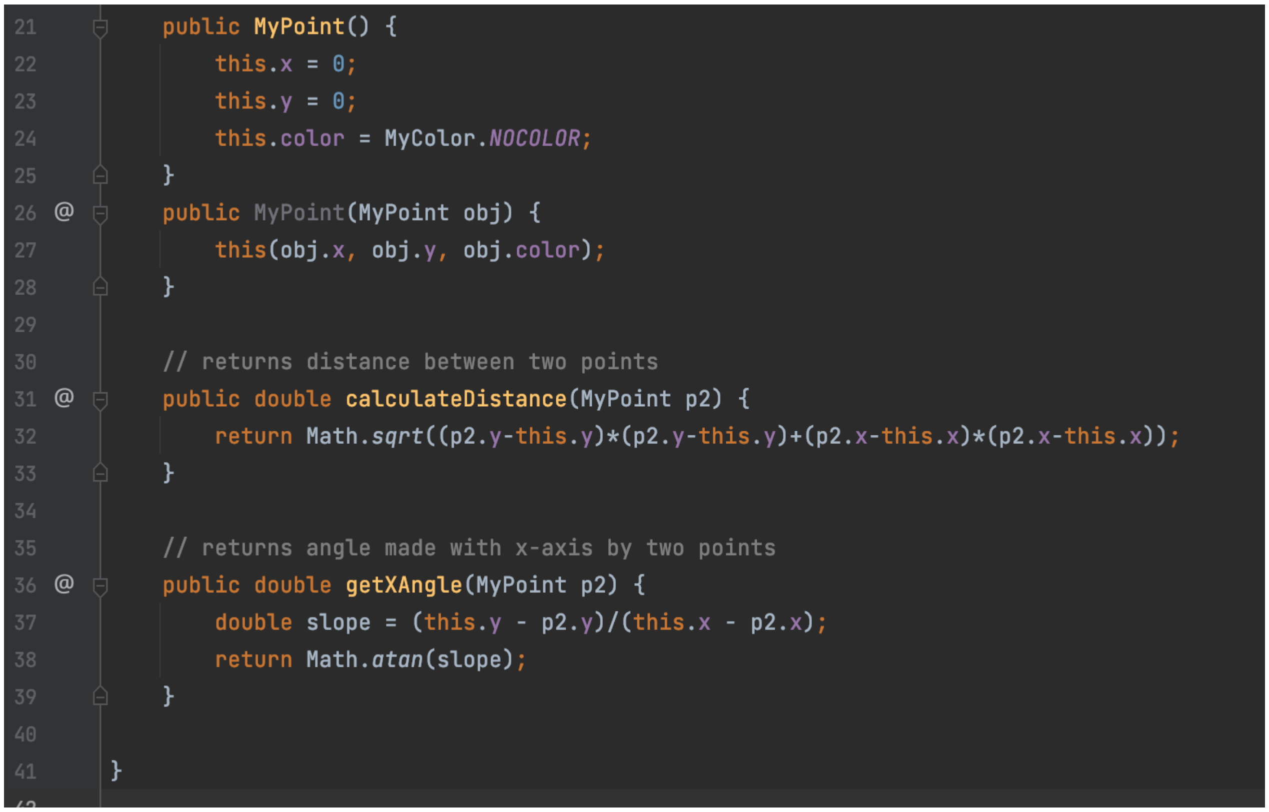


```
        }
        if(g<0 || g>255){
            g = 0;
        }
        if(b<0 || b>255){
            b = 0;
        }
        if(opacity<0 || opacity>255){
            opacity = 0;
        }
        this.r = r;
        this.g = g;
        this.b = b;
        this.opacity = opacity;
    }

    // return hexadecimal value of the color specified by MyColor
    public String getHexRepresentation() {
        String hex = "#";
        hex += Integer.toHexString(r);
        hex += Integer.toHexString(g);
        hex += Integer.toHexString(b);

        return hex;
    }

    // returns javaFX representation of MyColor
    public Color getColor() { return Color.rgb(r, g, b, v: opacity/255.0); }
}
```

- **MyPoint.java:**

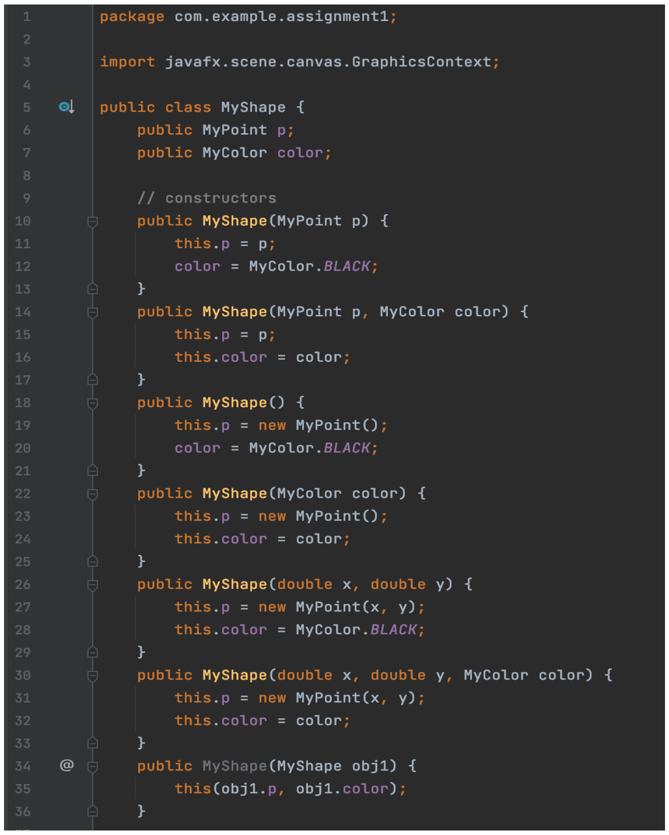


```java
package com.example.assignment1;

public class MyPoint {
    // x and y coordinate of a cartesian point
    public double x;
    public double y;
    // color of the shape
    public MyColor color;

    // constructors
    public MyPoint(double x, double y) {
        this.x = x;
        this.y = y;
        this.color = MyColor.NOCOLOR;
    }
    public MyPoint(double x, double y, MyColor color) {
        this.x = x;
        this.y = y;
        this.color = color;
    }
```

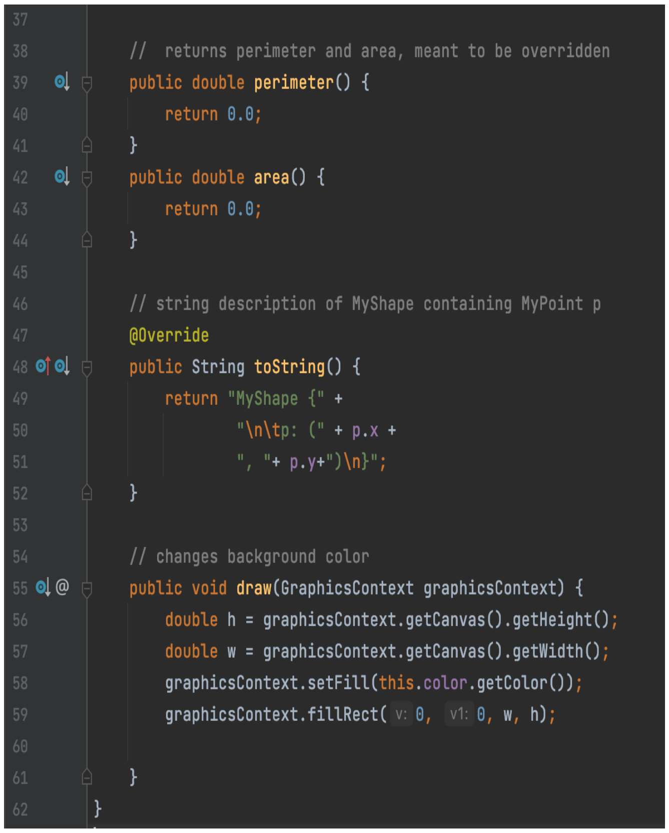


```java
    public MyPoint() {
        this.x = 0;
        this.y = 0;
        this.color = MyColor.NOCOLOR;
    }
    public MyPoint(MyPoint obj) {
        this(obj.x, obj.y, obj.color);
    }

    // returns distance between two points
    public double calculateDistance(MyPoint p2) {
        return Math.sqrt((p2.y-this.y)*(p2.y-this.y)+(p2.x-this.x)*(p2.x-this.x));
    }

    // returns angle made with x-axis by two points
    public double getXAngle(MyPoint p2) {
        double slope = (this.y - p2.y)/(this.x - p2.x);
        return Math.atan(slope);
    }

}
```

- **MyShape.java:**

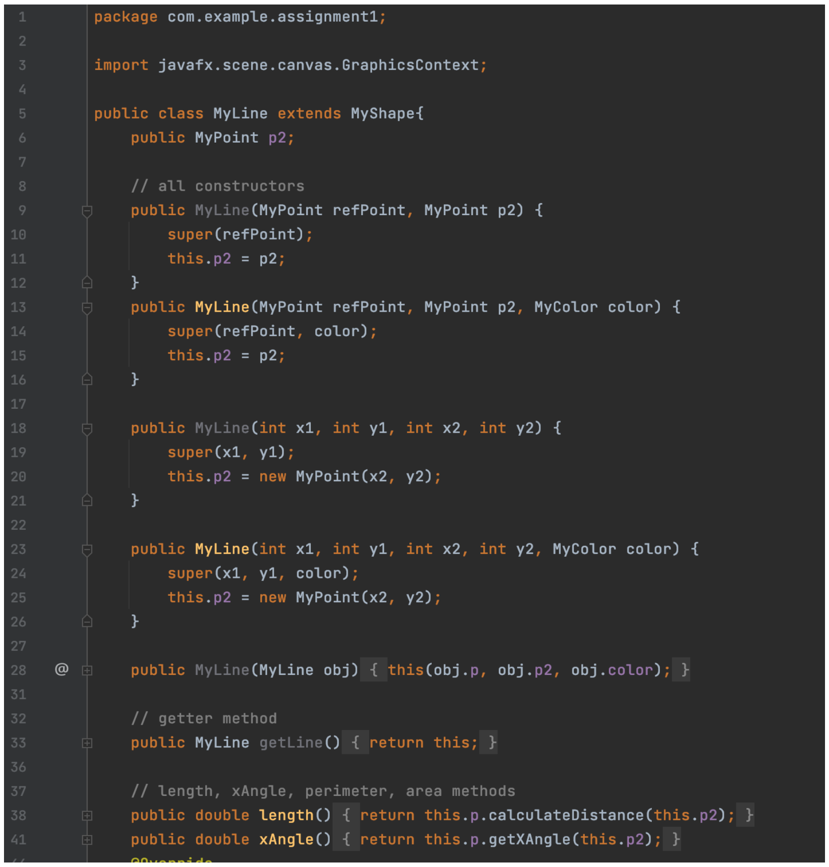


```java
package com.example.assignment1;

import javafx.scene.canvas.GraphicsContext;

public class MyShape {
    public MyPoint p;
    public MyColor color;

    // constructors
    public MyShape(MyPoint p) {
        this.p = p;
        color = MyColor.BLACK;
    }
    public MyShape(MyPoint p, MyColor color) {
        this.p = p;
        this.color = color;
    }
    public MyShape() {
        this.p = new MyPoint();
        color = MyColor.BLACK;
    }
    public MyShape(MyColor color) {
        this.p = new MyPoint();
        this.color = color;
    }
    public MyShape(double x, double y) {
        this.p = new MyPoint(x, y);
        this.color = MyColor.BLACK;
    }
    public MyShape(double x, double y, MyColor color) {
        this.p = new MyPoint(x, y);
        this.color = color;
    }
    public MyShape(MyShape obj1) {
        this(obj1.p, obj1.color);
    }
```

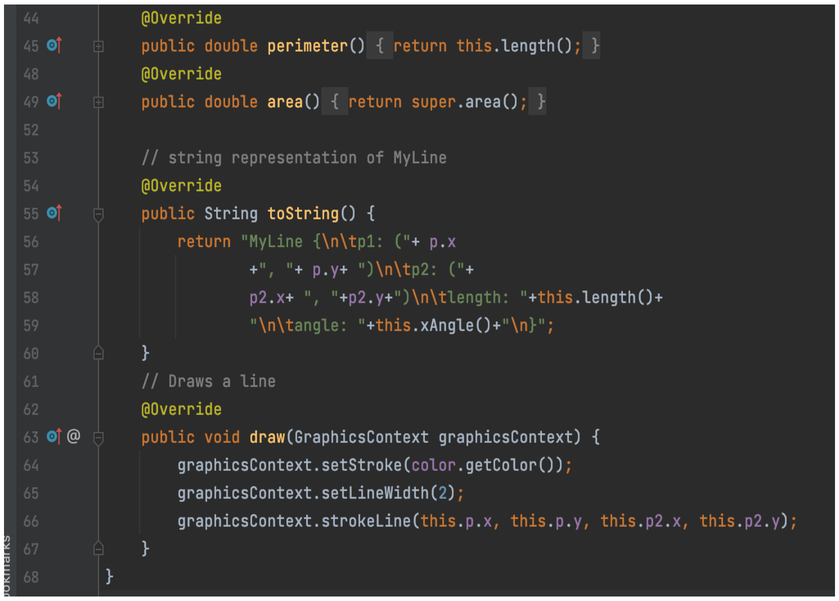


```java

    //  returns perimeter and area, meant to be overridden
    public double perimeter() {
        return 0.0;
    }
    public double area() {
        return 0.0;
    }

    // string description of MyShape containing MyPoint p
    @Override
    public String toString() {
        return "MyShape {" +
                "\n\tp: (" + p.x +
                ", "+ p.y+")\n}";
    }

    // changes background color
    public void draw(GraphicsContext graphicsContext) {
        double h = graphicsContext.getCanvas().getHeight();
        double w = graphicsContext.getCanvas().getWidth();
        graphicsContext.setFill(this.color.getColor());
        graphicsContext.fillRect( v: 0,  v1: 0, w, h);

    }
}
```

- **MyLine.java:**

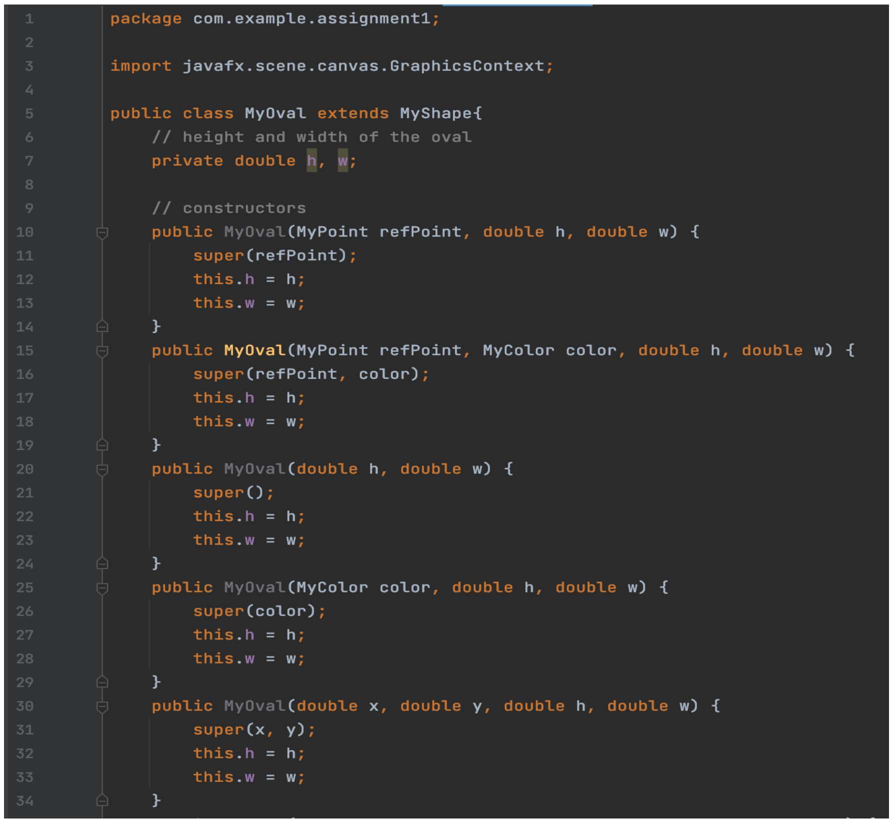


```java
package com.example.assignment1;

import javafx.scene.canvas.GraphicsContext;

public class MyLine extends MyShape{
    public MyPoint p2;

    // all constructors
    public MyLine(MyPoint refPoint, MyPoint p2) {
        super(refPoint);
        this.p2 = p2;
    }
    public MyLine(MyPoint refPoint, MyPoint p2, MyColor color) {
        super(refPoint, color);
        this.p2 = p2;
    }

    public MyLine(int x1, int y1, int x2, int y2) {
        super(x1, y1);
        this.p2 = new MyPoint(x2, y2);
    }

    public MyLine(int x1, int y1, int x2, int y2, MyColor color) {
        super(x1, y1, color);
        this.p2 = new MyPoint(x2, y2);
    }

    public MyLine(MyLine obj) { this(obj.p, obj.p2, obj.color); }

    // getter method
    public MyLine getLine() { return this; }

    // length, xAngle, perimeter, area methods
    public double length() { return this.p.calculateDistance(this.p2); }
    public double xAngle() { return this.p.getXAngle(this.p2); }
```

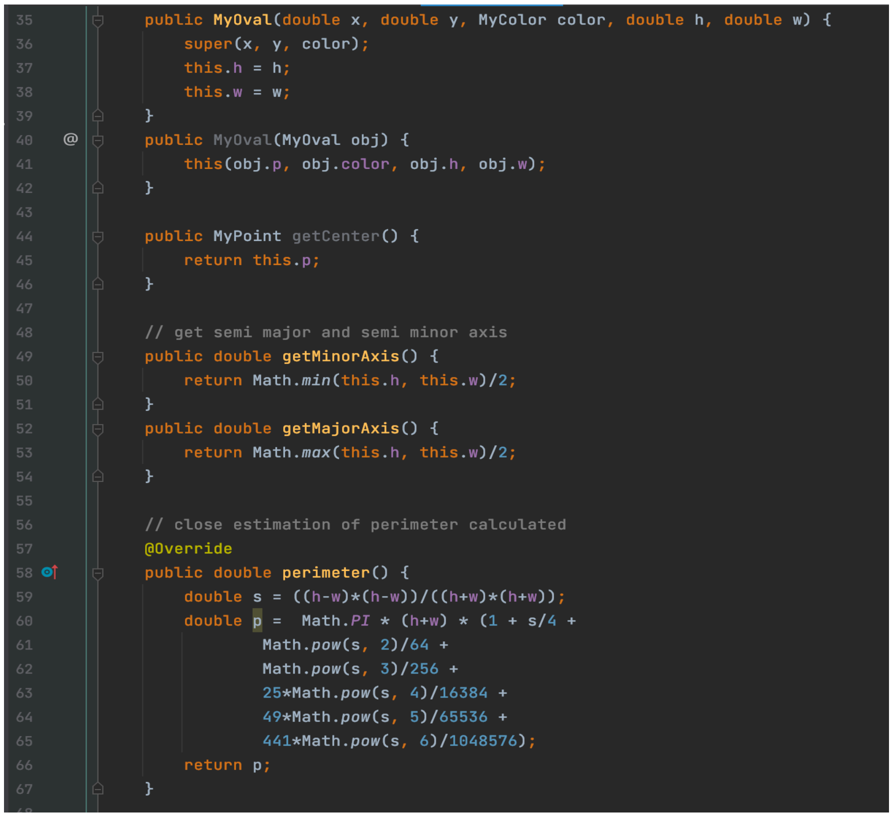


```java
    @Override
    public double perimeter() { return this.length(); }
    @Override
    public double area() { return super.area(); }

    // string representation of MyLine
    @Override
    public String toString() {
        return "MyLine {\n\tp1: ("+ p.x
                +", "+ p.y+ ")\n\tp2: ("+
                p2.x+ ", "+p2.y+")\n\tlength: "+this.length()+
                "\n\tangle: "+this.xAngle()+"\n}";
    }
    // Draws a line
    @Override
    public void draw(GraphicsContext graphicsContext) {
        graphicsContext.setStroke(color.getColor());
        graphicsContext.setLineWidth(2);
        graphicsContext.strokeLine(this.p.x, this.p.y, this.p2.x, this.p2.y);
    }
}
```

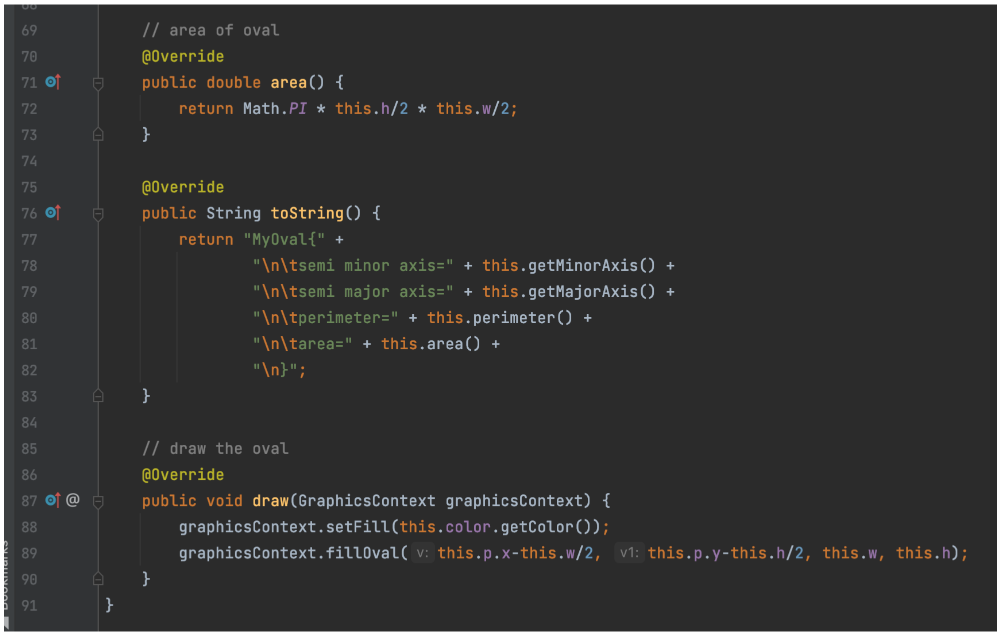
- **MyOval:**



```java
package com.example.assignment1;

import javafx.scene.canvas.GraphicsContext;

public class MyOval extends MyShape{
    // height and width of the oval
    private double h, w;

    // constructors
    public MyOval(MyPoint refPoint, double h, double w) {
        super(refPoint);
        this.h = h;
        this.w = w;
    }
    public MyOval(MyPoint refPoint, MyColor color, double h, double w) {
        super(refPoint, color);
        this.h = h;
        this.w = w;
    }
    public MyOval(double h, double w) {
        super();
        this.h = h;
        this.w = w;
    }
    public MyOval(MyColor color, double h, double w) {
        super(color);
        this.h = h;
        this.w = w;
    }
    public MyOval(double x, double y, double h, double w) {
        super(x, y);
        this.h = h;
        this.w = w;
    }
```

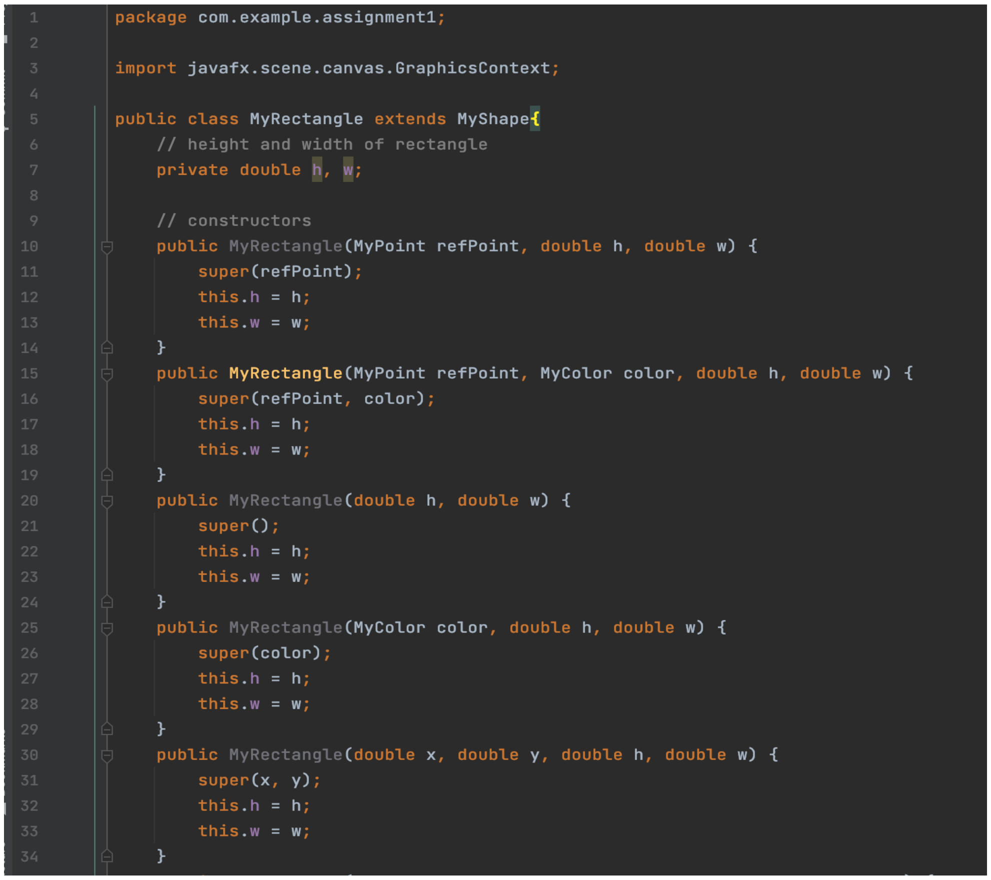


```java
    public MyOval(double x, double y, MyColor color, double h, double w) {
        super(x, y, color);
        this.h = h;
        this.w = w;
    }
    public MyOval(MyOval obj) {
        this(obj.p, obj.color, obj.h, obj.w);
    }

    public MyPoint getCenter() {
        return this.p;
    }

    // get semi major and semi minor axis
    public double getMinorAxis() {
        return Math.min(this.h, this.w)/2;
    }
    public double getMajorAxis() {
        return Math.max(this.h, this.w)/2;
    }

    // close estimation of perimeter calculated
    @Override
    public double perimeter() {
        double s = ((h-w)*(h-w))/((h+w)*(h+w));
        double p =  Math.PI * (h+w) * (1 + s/4 +
                Math.pow(s, 2)/64 +
                Math.pow(s, 3)/256 +
                25*Math.pow(s, 4)/16384 +
                49*Math.pow(s, 5)/65536 +
                441*Math.pow(s, 6)/1048576);
        return p;
    }
```

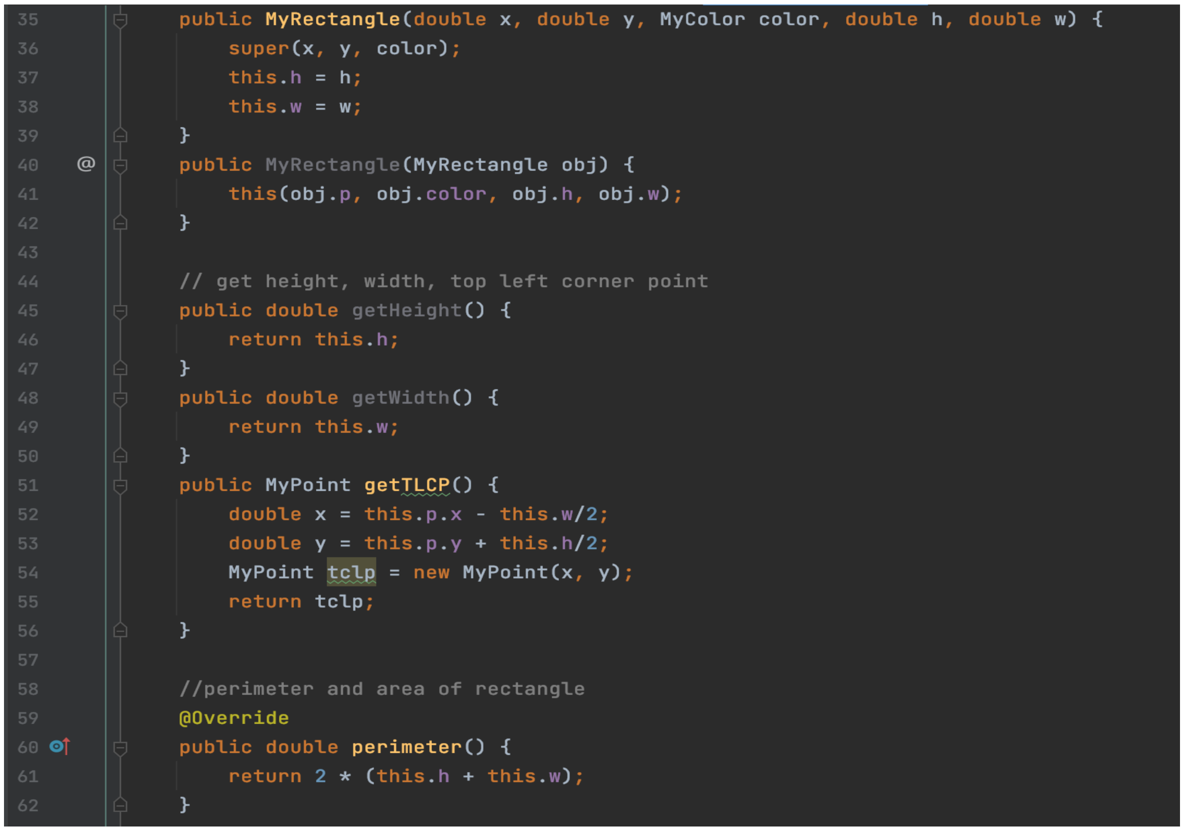


```
    // area of oval
    @Override
    public double area() {
        return Math.PI * this.h/2 * this.w/2;
    }

    @Override
    public String toString() {
        return "MyOval{" +
                "\n\tsemi minor axis=" + this.getMinorAxis() +
                "\n\tsemi major axis=" + this.getMajorAxis() +
                "\n\tperimeter=" + this.perimeter() +
                "\n\tarea=" + this.area() +
                "\n}";
    }

    // draw the oval
    @Override
    public void draw(GraphicsContext graphicsContext) {
        graphicsContext.setFill(this.color.getColor());
        graphicsContext.fillOval( v: this.p.x-this.w/2,  v1: this.p.y-this.h/2, this.w, this.h);
    }
}
```

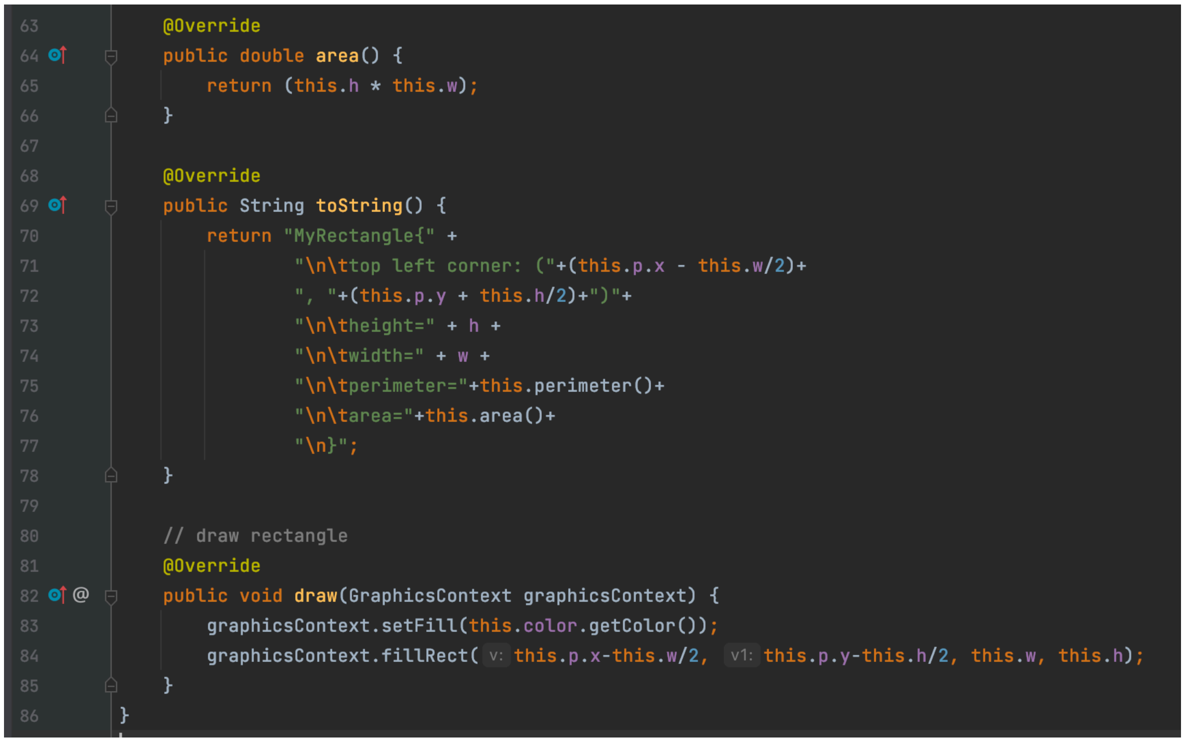
- **MyRecngle.java:**



```
package com.example.assignment1;

import javafx.scene.canvas.GraphicsContext;

public class MyRectangle extends MyShape{
    // height and width of rectangle
    private double h, w;

    // constructors
    public MyRectangle(MyPoint refPoint, double h, double w) {
        super(refPoint);
        this.h = h;
        this.w = w;
    }
    public MyRectangle(MyPoint refPoint, MyColor color, double h, double w) {
        super(refPoint, color);
        this.h = h;
        this.w = w;
    }
    public MyRectangle(double h, double w) {
        super();
        this.h = h;
        this.w = w;
    }
    public MyRectangle(MyColor color, double h, double w) {
        super(color);
        this.h = h;
        this.w = w;
    }
    public MyRectangle(double x, double y, double h, double w) {
        super(x, y);
        this.h = h;
        this.w = w;
    }
```



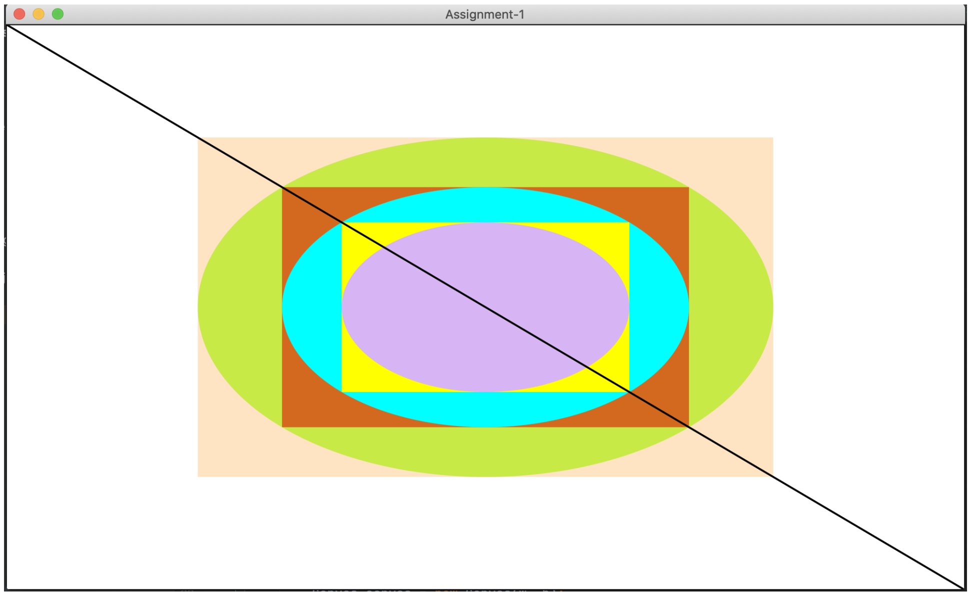
```java
    public MyRectangle(double x, double y, MyColor color, double h, double w) {
        super(x, y, color);
        this.h = h;
        this.w = w;
    }
    public MyRectangle(MyRectangle obj) {
        this(obj.p, obj.color, obj.h, obj.w);
    }

    // get height, width, top left corner point
    public double getHeight() {
        return this.h;
    }
    public double getWidth() {
        return this.w;
    }
    public MyPoint getTLCP() {
        double x = this.p.x - this.w/2;
        double y = this.p.y + this.h/2;
        MyPoint tclp = new MyPoint(x, y);
        return tclp;
    }

    //perimeter and area of rectangle
    @Override
    public double perimeter() {
        return 2 * (this.h + this.w);
    }
```



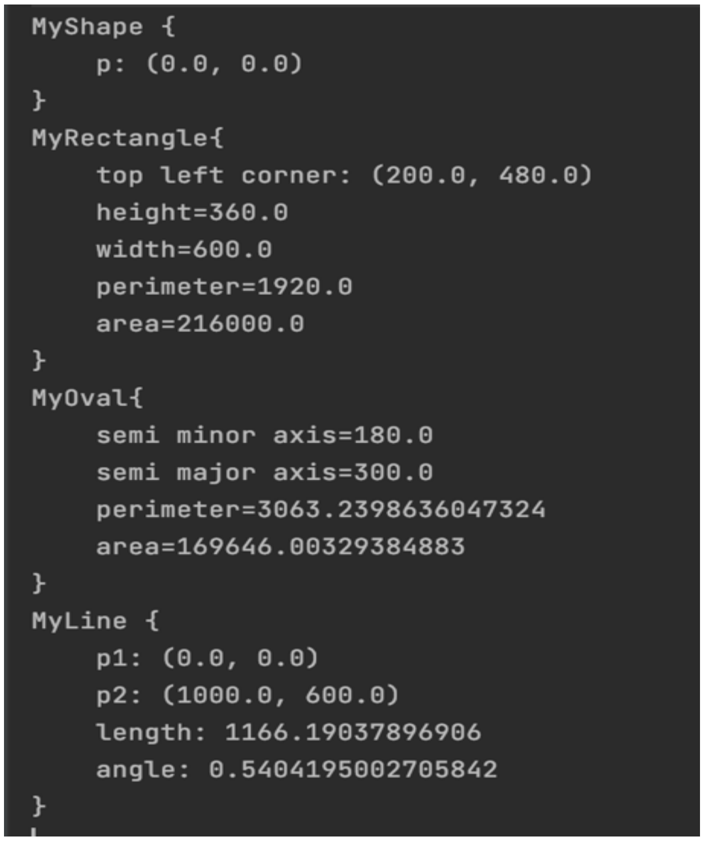
```java
    @Override
    public double area() {
        return (this.h * this.w);
    }

    @Override
    public String toString() {
        return "MyRectangle{" +
                "\n\ttop left corner: ("+(this.p.x - this.w/2)+
                ", "+(this.p.y + this.h/2)+")"+
                "\n\theight=" + h +
                "\n\twidth=" + w +
                "\n\tperimeter="+this.perimeter()+
                "\n\tarea="+this.area()+
                "\n}";
    }

    // draw rectangle
    @Override
    public void draw(GraphicsContext graphicsContext) {
        graphicsContext.setFill(this.color.getColor());
        graphicsContext.fillRect( v: this.p.x-this.w/2,  v1: this.p.y-this.h/2, this.w, this.h);
    }
}
```

Screenshot of Output:

- **JavaFX Window:**



- **Console:**



```
MyShape {
    p: (0.0, 0.0)
}
MyRectangle{
    top left corner: (200.0, 480.0)
    height=360.0
    width=600.0
    perimeter=1920.0
    area=216000.0
}
MyOval{
    semi minor axis=180.0
    semi major axis=300.0
    perimeter=3063.2398636047324
    area=169646.00329384883
}
MyLine {
    p1: (0.0, 0.0)
    p2: (1000.0, 600.0)
    length: 1166.19037896906
    angle: 0.5404195002705842
}
```