**Name: Najia Jahan**

**Instructor: Hesham A. Auda**

# Assignment 1

**CSC 221 Software Design Laboratory- Summer 2022**

**Date: 26th June 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:

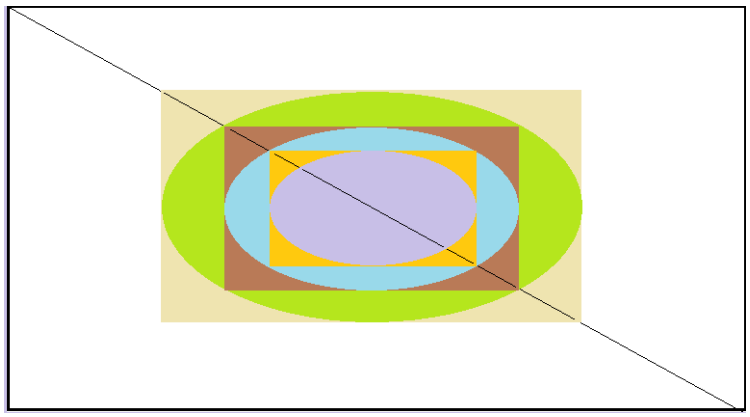
*a.* *getCenter, getMinorAxis, getMajorAxis* — return, respectively, the center point and semi minor and major axes of the **MyOval** object;
*b.* *perimeter, area* — return, respectively, the perimeter and area of the **MyOval** object;
*c.* *toString*— returns a string representation of the **MyOval**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

## [2] Solution Methods

The program has 7 classes including Main. In accordance with the requirements of the assignment, a superclass MyShape has been created. MyColor is an enum class that has been used to color different shapes. Three classes- MyLine, MyRectangle, and MyOval extend the superclass MyShape. Methods used in each class are mentioned below:

1. **Main.java:** All the classes are utilized here to create and output different shapes and colors.
    - ***Imports:***

      javafx.application.Application;

      javafx.scene.Scene;

      javafx.scene.canvas.Canvas;

      javafx.scene.canvas.GraphicsContext;

      javafx.scene.layout.StackPane;

      javafx.stage.Stage;
    - ***Methods :***

      **main()**

      **start() :** This method is utilized to use Javafx applications.
2. **MyPoint.java :** This class is used to create a reference point p(x, y) of the Java display coordinate system which is used in other classes.
    - ***Methods :***

      **setX()**

      **getX()**

      **setY()**

**getY()**

3. **MyColor.java:** This is an enum class. It is used in the superclass and the subclasses to define different colors.

    - ***Imports:***

      javafx.scene.paint.Color;

    - ***Constructor :***

      **MyColor()**

    - ***Methods :***

      **getColor()** : Returns chosen color by using r, g, b, argb values and importing javafx.scene.paint.Color.

4. **MyShape.java:** This is a Superclass in the class hierarchy. The methods used in this class are overridden in the other subclasses. This is a default state since it does not have any specific shape values therefore initially its' methods return 0.

    - ***Imports:***

      javafx.scene.canvas.GraphicsContext;

      javafx.scene.paint.Color;

    - ***Constructor :***

      **MyShape() :** Default constructor.

      **MyShape() :** Parametrized constructor.

    - ***Methods :***

      **getX()** : Returns x coordinate.

      **getY()** : Returns y coordinate.

      **getColor()** : Returns color.

**perimeter()** : Initially returns 0 and gets overridden in other classes.

**area()** : Initially returns 0 and gets overridden in other classes.

**draw():** Draws shapes using GraphicsContext.

5. **<u>MyLine.java</u>:** Extends MyShape. Object of this class is a straight line which is defined by the endpoints of MyPoint class.

   - ***Imports :***

     javafx.scene.canvas.GraphicsContext;

     javafx.scene.paint.Color;

   - ***Constructor :***

     **MyLine()**

   - ***Methods :***

     **getLine()** : Returns an object of MyLine.

     **length():** Calculates and returns the length of a line segment using coordinates and sqrt() function.

     **xAngle():** Calculates and returns the angle with the x-axis of MyLine object.

     **perimeter():** Returns the length of the line.

     **area():** Returns 0.

     **toString():** Returns a string representation of MyLine object in chosen format which includes- two endpoints, length, angle, and color.

     **draw():** Draws lines using GraphicsContext.

6. **<u>MyRectangle.java</u>:** Extends MyShape. Creates a rectangle object of height h and width w.

   - ***Imports:***

javafx.scene.canvas.GraphicsContext;

javafx.scene.paint.Color;

javafx.util.Pair;

- ***Constructor :***

  **MyRectangle()**

- ***Methods :***

  **getTLCP**() : Returns the point of the Top Left Corner using javafx.util.Pair.

  **getWidth()** : Returns the width of the MyRectangle object.

  **getHeight()** : Returns the height of the MyRectangle object.

  **perimeter():** Calculates and returns the perimeter of the MyRectangle object.

  **area():** Calculates and returns the area of the MyRectangle object.

  **toString():** Returns a string representation of MyRectangle object in chosen format which includes- top left corner point(TLCP), width, height, perimeter, and area.

  **draw():** Draws MyRectangle objects using GraphicsContext.

7. **MyOval.java:** Extends MyShape. Creates an ellipse within the rectangle of height h and width w.

   - ***Imports:***

     javafx.scene.canvas.GraphicsContext;

     javafx.scene.paint.Color;

   - ***Constructor :***

     **MyOval()**

   - ***Methods :***

**getCenter()** : Returns the center of the MyOval object.

**getMinorAxis()** : Returns the length of the semi-minor axis of the MyOval object.

**getMajorAxis()** : Returns the length of the semi-major axis of the MyOval object.

**perimeter():** Calculates and returns the perimeter of the MyOval object.

**area():** Calculates and returns the area of the MyOval object.

**toString():** Returns a string representation of MyOval object in chosen format which includes- lengths of semi-major and semi-minor axes, perimeter, and area.

**draw():** Draws MyOval objects using GraphicsContext.

## [3] All Codes Developed

Below are the screenshots of the codes of all classes.

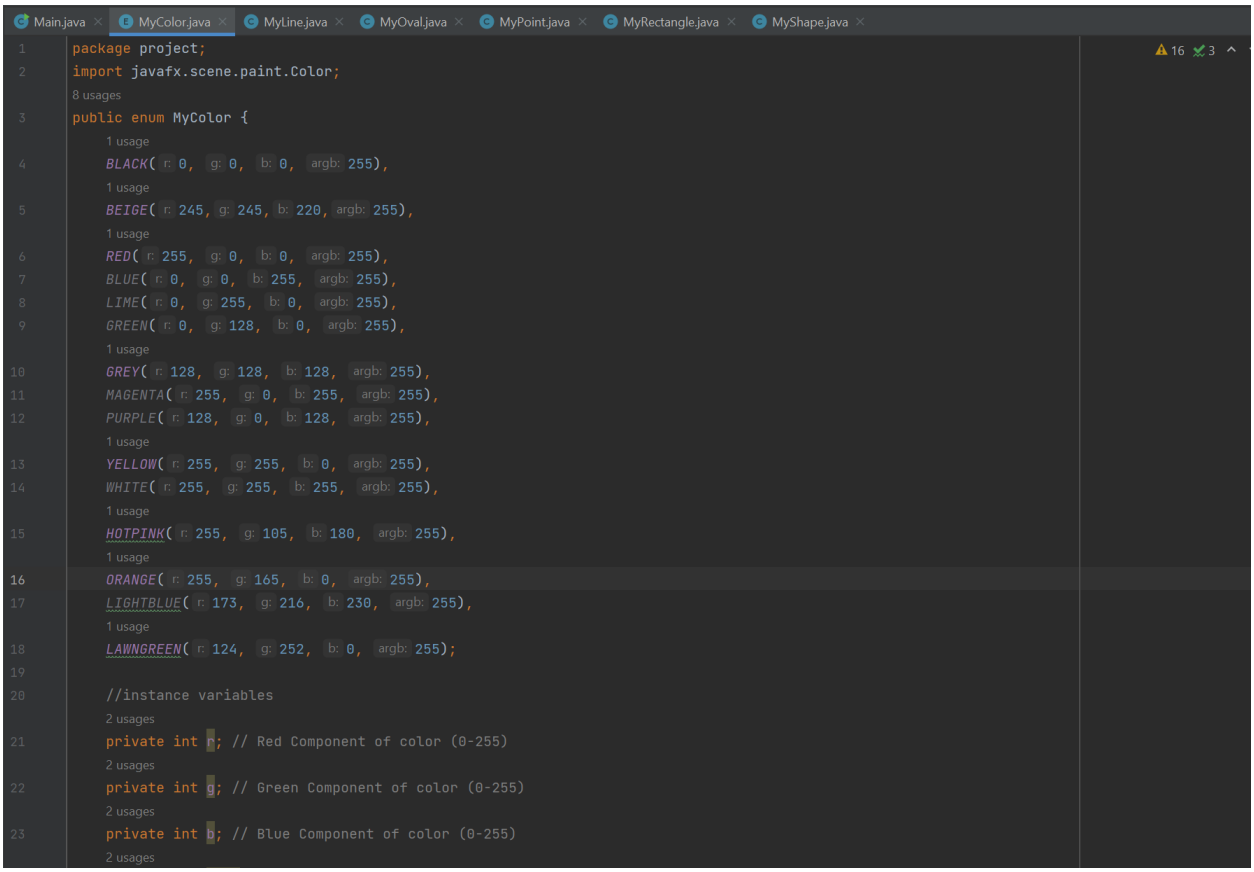
1. Codes for **Main.java:**

Main.java | MyColor.java | MyLine.java | MyOval.java | MyPoint.java | MyRectangle.java | MyShape.java

```java
package project;
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;
public class Main extends Application {
//Boundaries
    6 usages
    final int WIDTH = 1200;
    6 usages
    final int HEIGHT = 800;
    @Override
    public void start(Stage myStage) {
        try {
// Setting up stage, canvas, and graphics context
            myStage.setTitle("Ta daa! A beautiful shape!");
            Canvas canvas = new Canvas(WIDTH, HEIGHT);
            GraphicsContext gc = canvas.getGraphicsContext2D();
            MyShape shape = new MyShape();
            System.out.println(shape + "\n");
// Drawing ovals and rectangles using different colors
            double width = WIDTH / 2, height = HEIGHT / 2, centerX = WIDTH / 2, centerY = HEIGHT / 2;
            double w = width / 2; double h = height / 2;
            MyRectangle rectangle;
            rectangle = new MyRectangle( x: centerX - w,  y: centerY - h, width, height, MyColor.LAWNGREEN.getCol());
            MyOval oval;
            rectangle.draw(gc);
            System.out.println(rectangle + "\n");
            for (int i = 1; i < 6; ++i) {
                if (i % 2 != 0) {
                    if (i == 1){
                        oval = new MyOval( x: centerX - w,  y: centerY - h, width, height, MyColor.BEIGE.getCol());}
                    else if (i == 3){
                        oval = new MyOval( x: centerX - width / 2,  y: centerY - height / 2, width, height, MyColor.YELLOW.getCol());}
                    else {
                        oval = new MyOval( x: centerX - width / 2,  y: centerY - height / 2, width, height, MyColor.ORANGE.getCol());
                    }
                    oval.draw(gc);
                    System.out.println(oval + "\n");
                } else {
                    width /= Math.sqrt(2);
                    height /= Math.sqrt(2);
                    if (i <= 3){
                        rectangle = new MyRectangle( x: centerX - width / 2,  y: centerY - height / 2, width, height, MyColor.HOTPINK.getCol());
                    }
                    else {
                        rectangle = new MyRectangle( x: centerX - width / 2,  y: centerY - height / 2, width, height, MyColor.GREY.getCol());}
                    rectangle.draw(gc);
                    System.out.println(rectangle + "\n\n");}
            }
// horizontal line border
            MyLine line1 = new MyLine( x1: 0,  y1: 0, WIDTH, HEIGHT, MyColor.BLACK.getCol());
            line1.draw(gc);
            System.out.println(line1 + "\n");
            gc.setLineWidth(10);
            gc.strokeRect( x: 0,  y: 0, WIDTH, HEIGHT);
// Setting up stack pane for the shape
            StackPane root = new StackPane(canvas);
            Scene scene = new Scene(root, WIDTH, HEIGHT);
            myStage.setScene(scene);
            myStage.show();
        } catch (Exception e) {
            e.printStackTrace();}
    }
//main method
    public static void main(String[] args) { launch(args); }
}
```

2. Codes for **MyPoint.java** class:


3. Codes for **MyColor.java** class:



```
    private int argb; // Opacity Component of color (0-255)

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

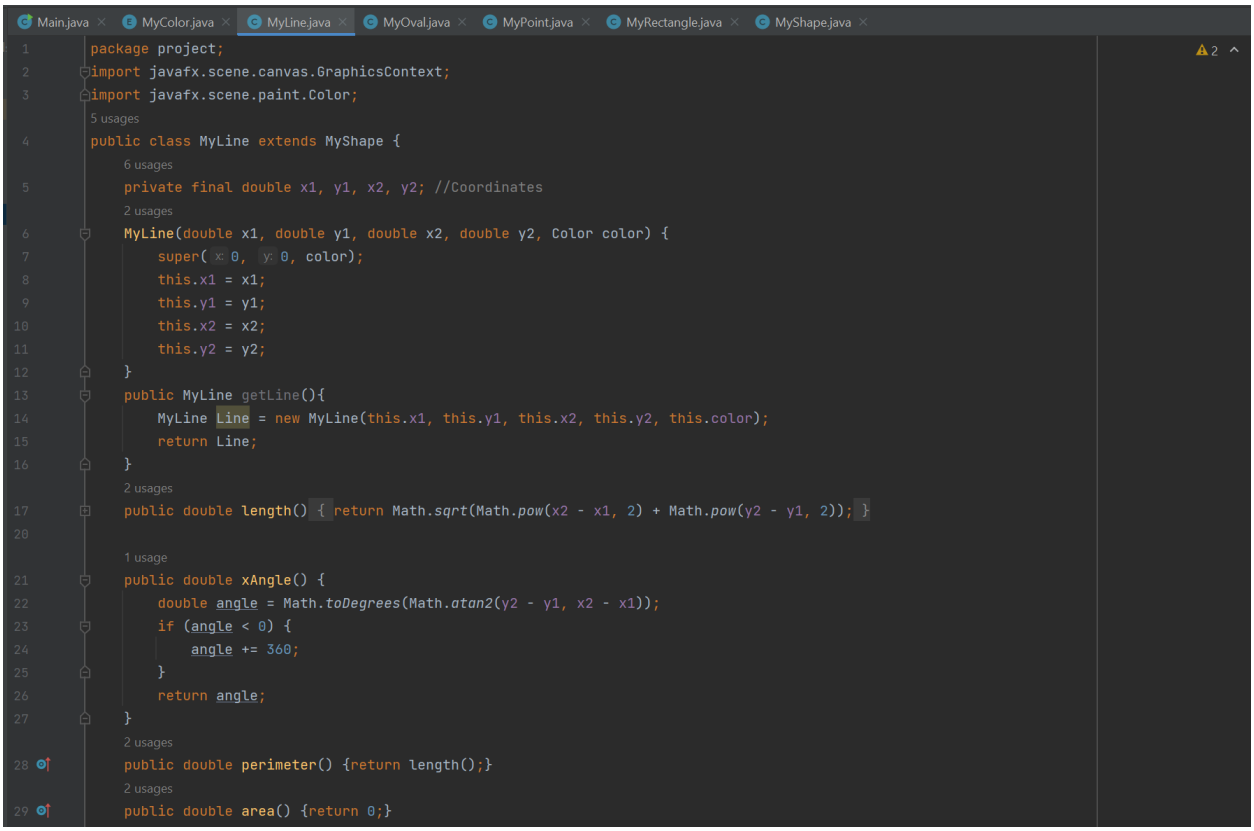
4. Codes for **MyShape.java** class:

Main.java | MyColor.java | MyLine.java | MyOval.java | MyPoint.java | MyRectangle.java | MyShape.java

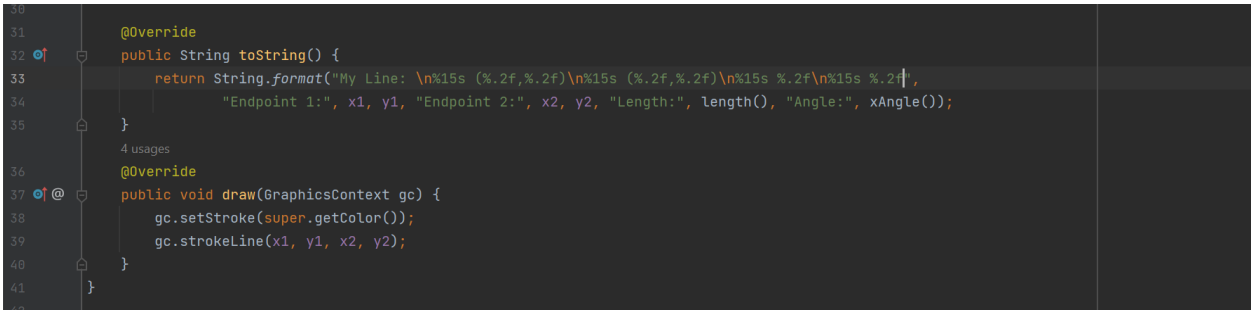
```
package project;

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

5 usages  3 inheritors
public class MyShape  {
    private MyPoint point;
    3 usages
    private double x,y;
    4 usages
    public Color color;
    1 usage
    MyShape() {
        this.x = 0;
        this.y = 0;
        this.color = MyColor.RED.getCol();
    }
    3 usages
    MyShape(double x, double y, Color color) {
        this.x = x;
        this.y = y;
        this.color = color;
    }
    6 usages
    public double getX() { return this.x; }
    6 usages
    public double getY() { return this.y; }
    5 usages
    public Color getColor() {
        return color;
    }
    2 usages  3 overrides
    public double perimeter() { return 0; }
    2 usages  3 overrides
    public double area() { return 0; }
```

```
    @Override
    public String toString() {
        return String.format("Default My Shape: \n%15s (%.2f,%.2f)\n\t\tPerimeter: 0.0\n\t\tArea: 0.0", "Center:", getX(), getY());
    }
    4 usages  3 overrides
    public void draw(GraphicsContext gc){

    };

}
```

5. Codes for **MyLine.java** class:

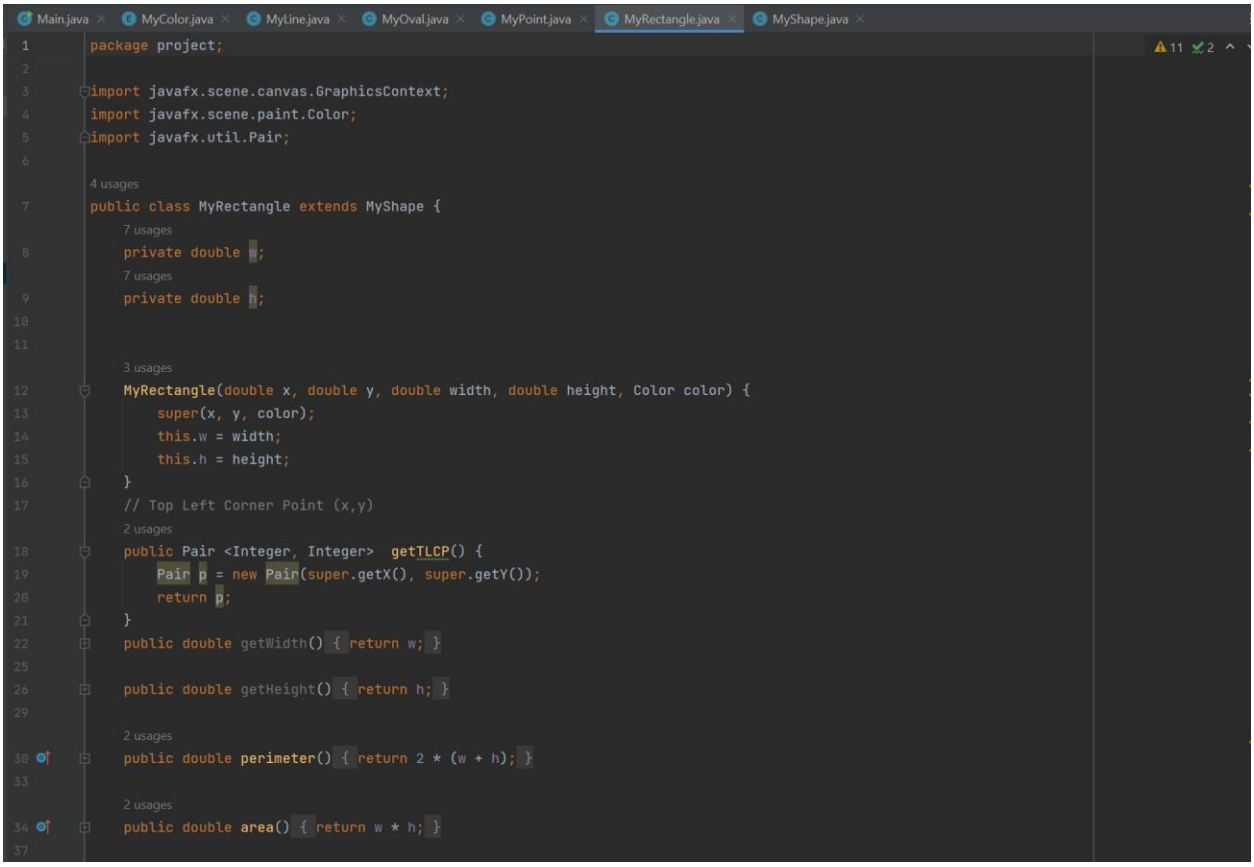


```java
package project;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;
5 usages
public class MyLine extends MyShape {
    6 usages
    private final double x1, y1, x2, y2; //Coordinates
    2 usages
    MyLine(double x1, double y1, double x2, double y2, Color color) {
        super( x: 0,  y: 0, color);
        this.x1 = x1;
        this.y1 = y1;
        this.x2 = x2;
        this.y2 = y2;
    }
    public MyLine getLine(){
        MyLine Line = new MyLine(this.x1, this.y1, this.x2, this.y2, this.color);
        return Line;
    }
    2 usages
    public double length() { return Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2)); }

    1 usage
    public double xAngle() {
        double angle = Math.toDegrees(Math.atan2(y2 - y1, x2 - x1));
        if (angle < 0) {
            angle += 360;
        }
        return angle;
    }
    2 usages
    public double perimeter() {return length();}
    2 usages
    public double area() {return 0;}
```



```java
    @Override
    public String toString() {
        return String.format("My Line: \n%15s (%.2f,%.2f)\n%15s (%.2f,%.2f)\n%15s %.2f\n%15s %.2f",
                "Endpoint 1:", x1, y1, "Endpoint 2:", x2, y2, "Length:", length(), "Angle:", xAngle());
    }
    4 usages
    @Override
    public void draw(GraphicsContext gc) {
        gc.setStroke(super.getColor());
        gc.strokeLine(x1, y1, x2, y2);
    }
}
```

6. Codes for **MyRectangle.java** class:



```
package project;

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

4 usages
public class MyRectangle extends MyShape {
    7 usages
    private double w;
    7 usages
    private double h;


    3 usages
    MyRectangle(double x, double y, double width, double height, Color color) {
        super(x, y, color);
        this.w = width;
        this.h = height;
    }
    // Top Left Corner Point (x,y)
    2 usages
    public Pair <Integer, Integer>  getTLCP() {
        Pair p = new Pair(super.getX(), super.getY());
        return p;
    }
    public double getWidth() { return w; }

    public double getHeight() { return h; }

    2 usages
    public double perimeter() { return 2 * (w + h); }

    2 usages
    public double area() { return w * h; }
```

```
    4 usages
    @Override
    public void draw(GraphicsContext gc) {
        gc.setFill(super.getColor());
        gc.setStroke(super.getColor());
        gc.strokeRect(super.getX(), super.getY(), w, h);
        gc.fillRect(super.getX(), super.getY(), w, h);
    }

    @Override
    public String toString() {
        return String.format("My Rectangle: \n%15s (%.2f,%.2f)\n%15s %.2f\n%15s %.2f\n%15s %.2f\n%15s %.2f",
                "TLCP:", getTLCP().getKey(), getTLCP().getValue(), "Width:", w, "Height:", h, "Perimeter:", perimeter(),"Area:", area());
    }
}
```

7. Codes for **MyOval.java** class:

Main.java | MyColor.java | MyLine.java | MyOval.java | MyPoint.java | MyRectangle.java | MyShape.java

```java
package project;

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

4 usages
public class MyOval extends MyShape {
    2 usages
    private final MyPoint center;
    6 usages
    private double w;
    6 usages
    private double h;

    3 usages
    MyOval(double x, double y, double width, double height, Color color) {
// (x,y) is top left corner
        super(x, y, color);
        this.w = width;
        this.h = height;
        this.center = new MyPoint( x: x + width / 2,  y: y + height / 2);
    }
    public MyPoint getCenter() { return center; }
    2 usages
    public double area() { return Math.PI * w * h * 0.5; }
    2 usages
    public double perimeter() { return 2 * Math.PI * Math.sqrt((w + h) / 2); }

    1 usage
    public double getMinorAxis(){return h/2;}
    1 usage
    public double getMajorAxis(){return w/2;}

    @Override
    public String toString() {
        return String.format("My Oval: \n%15s %.2f\n%15s %.2f\n%15s %.2f\n%15s %.2f" , "Length of Semi-major axis:", getMajorAxis(),
                "Length of Semi-minor axis:", getMinorAxis(), "Perimeter:", perimeter(), "Area:", area());
    }
    4 usages
    @Override
    public void draw(GraphicsContext gc) {
        gc.setFill(super.getColor());
        gc.setStroke(super.getColor());
        gc.strokeOval(super.getX(), super.getY(), w, h);
        gc.fillOval(super.getX(), super.getY(), w, h);
    }
}
```

## [4] Output of the Tasks

Below are the screenshots of the outputs of required tasks. There will be two different outputs- one will be the console output where the objects of the classes will print their string representation, another one will be the Javafx window where the shapes will be shown filled with chosen colors.

1. Console Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Default My Shape:
        Center: (0.00,0.00)
        Perimeter: 0.0
        Area: 0.0

My Rectangle:
          TLCP: (300.00,200.00)
         Width: 600.00
        Height: 400.00
     Perimeter: 2000.00
          Area: 240000.00

My Oval:
Length of Semi-major axis: 300.00
Length of Semi-minor axis: 200.00
     Perimeter: 140.50
          Area: 376991.12

My Rectangle:
          TLCP: (387.87,258.58)
         Width: 424.26
        Height: 282.84
     Perimeter: 1414.21
          Area: 120000.00
```

```
My Oval:
Length of Semi-major axis: 212.13
Length of Semi-minor axis: 141.42
     Perimeter: 118.14
          Area: 188495.56

My Rectangle:
          TLCP: (450.00,300.00)
         Width: 300.00
        Height: 200.00
     Perimeter: 1000.00
          Area: 60000.00

My Oval:
Length of Semi-major axis: 150.00
Length of Semi-minor axis: 100.00
     Perimeter: 99.35
          Area: 94247.78

My Line:
    Endpoint 1: (0.00,0.00)
    Endpoint 2: (1200.00,800.00)
        Length: 1442.22
         Angle: 33.69


Process finished with exit code 0
```

2. JavaFX window Output: