Fall 2022 | CSC 22100 | Assignment 1

Alhamza Muswara

Problem/Task:

1. Create a hierarchy of Java classes as follows: "MyRectangle" is a "MyShape"; "MyOval" is a "MyShape".
2. Use JavaFX graphics and the class hierarchy to draw the geometric configuration comprised of a sequence of alternating concentric circles and their 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 circles 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.

Solution methods:

Class MyPoint

The SetX() method assign a new y value for some point (x,y).

```
        void setX(double x) {
                this.x=x;
        }
```

The SetY() method assign a new y value for some point (x,y).

```
        void setY(double y) {
                this.y=y;
        }
```

The SetX() method returns x value on the x-axis for some point.

```
        double getX() {
                return x;
        }
```

The SetY() method returns y value on the y-axis for some point.

```
        double getY() {
                return y;
        }
```

Add or subtract x from this.x(instance variable).

```
        void shiftX(double x) {
            this.x+=x;
        }
```

Add or subtract y from this.y(instance variable).

```
        void shiftY(double y) {
                this.y+=y;
        }
```

Returns the point (x,y) as a string.

```
        public String toString() {
                return "MyPoint [x = "+x+", y = "+y+"]";
        }
```

Returns the distance from P(x,y) to some other point x2,y2.

```
        double getDistance(double x2, double y2){
                return (Math.sqrt(Math.pow(x2-this.x,2)+Math.pow(y2-this.y,2)));
        }
```

Returns the angle of line from the +x counterclockwise

```
        void getAngle(double x2, double y2) {
                  double m=((y2-this.y)/(x2-this.x));
                  System.out.println(Math.atan(m));
        }
```

Class MyShape

By instruction getArea() and getPerimeter() return 0.

```
	double getArea() {
		return 0;
	}

	double getPerimeter() {
		return 0;
	}
```

Returns empty string.

```
	public String toString() {
		return "";
	}
```

By instruction does nothing, and to be overridden in subclasses.

```
	void draw(GraphicsContext gc) {
	}
```

Class MyRectangle

MyRectangle constructor assign width and height variables and uses MyPoint object p to.

```
        MyRectangle (double x, double y, double w, double h){
             this.width=w;
             this.height=h;
             p.setX(x);
             p.setY(y);
        }
```

getP() returns the topleft corner point of the object.

```
        String getP() {
               return p.toString();
        }
```

getWidth() returns the width value.

```
        double getWidth() {
               return width;
        }
```

getHeight() returns the height value.

```
        double getHeight() {
               return height;
        }
```

getArea() return the area of a rectangle which is its height times(*) its width.

```
        @Override
        double getArea() {
             return (width*height);
        }
```

getPerimeter() returns the perimeter of a rectangle which we get by simply adding the length and the width, and then multiplying this sum by two

```
        @Override
        double getPerimeter() {
             return (2*width+2*height);
        }
```

toString() returns a string representation of the MyRectangle object with x, y, width, height, fill, area, and perimeter.

```
        @Override
        public String toString() {
               return "Rectangle[x="+p.getX()+", y="+p.getY()+", width="+width+",
               height="+height+", fill="+fill.toString()+",
               perimeter="+(2*width+2*height)+", area="+(width*height)+"]";
        }
```

draw() draws the geometry, sets the fill or color, and draws a filled geometry.

```
        @Override
        void draw(GraphicsContext gc) {
               gc.strokeRect(p.getX(), p.getY(), this.width, this.height);
               gc.setFill(fill.myColor());
               gc.fillRect(p.getX(), p.getY(), width, height);
        }
```

Class MyOval

Constructor specifies top-left corner point of object as well as width and height of object.

```
        MyOval(double x, double y, double w, double h){
               p.setX(x);
               p.setY(y);
               this.width=w;
               this.height=h;

        }
```

getArea() return a close approximation of the area of the oval.

```
        @Override
        double getArea() {
                    return (Math.PI*(width/2)*(height/2));
             }
```

getPerimeter() returns the perimeter of the oval.

```
        @Override
        double getPerimeter() {
                    return
(2*Math.PI*(Math.sqrt((Math.pow(height/2,2)+Math.pow(width/2,2))/2)));
             }
```

getX() and getY() return the x and y respectively of the center of the oval.

```
        //getX(center)
        double getCenterX() {
               return p.getX()+(width/2);
        }
        //getY,(center)
        double getCenterY() {
               return p.getY()+(height/2);
        }
```

toString() returns the string representation of the oval object that includes: x, y, width, height, fill, area, and perimeter.

```
        //toString
        @Override
        public String toString() {
               return "Oval[x="+p.getX()+", y="+p.getY()+", width="+width+",
height="+height+", fill="+fill.toString()+",
parameter="+2*Math.PI*(Math.sqrt((Math.pow(height/2,2)+Math.pow(width/2,2))/2))+",
area="+Math.PI*(width/2)*(height/2)+"]";
        }
```

draw() draws the geometry, sets the fill or color, and draws a filled geometry.

```
        @Override
        void draw(GraphicsContext gc){
               gc.strokeOval(p.getX(),p.getY(),this.width,this.height);
               gc.setFill(fill.myColor());
               gc.fillOval(p.getX(), p.getY(), width, height);
        }
```

Enum MyColor

Sets the appropriate rgba values for the MyColor objects.

```
        MyColor(int red, int green, int blue, double opacity){
                this.red=red;
                this.blue=blue;
                this.green=green;
                this.opacity=opacity;
        }
```

myColor() method returns a Java FX Color object.

```
        public Color myColor() {
                return Color.rgb(red, green, blue, opacity);
        }
```

toString() method returns a string hexadecimal representation of the RGB values.

```
        public String toString(){
                return String.format("#%02x%02x%02x", red, green, blue);
        }
```

Class Main

Here the start() method sets the dimensions of the canvas object. The PrimaryStage() method sets up and draws the shapes, and calculates each top-left point of each shape to result the required output. The canvas is then added into a scene and the scene into the stage. Finally, our code is finishes and outputs our requested graphics.

```
    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Assignment 1");
        Group root = new Group();

        //CHANGE CANVAS VARAIBLES
        double dx=750;
        double dy=400;
        //^

        Canvas canvas = new Canvas(dx, dy);
        GraphicsContext gc = canvas.getGraphicsContext2D();
        drawShapes(gc,dx,dy);
        root.getChildren().add(canvas);
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
    }

    private void drawShapes(GraphicsContext gc, double dx, double dy) {
       gc.setLineWidth(1);
       MyOval o = new MyOval(0,0,dx,dy);
       o.fill=MyColor.DARKKHAKI;
       o.draw(gc);
       System.out.println(o.fill.toString());

       dx=dx/2;
       dy=dy/2;
       MyRectangle r = new MyRectangle(dx-(dx*Math.sqrt(2))/2,dy-(dy*Math.sqrt(2))/2,
dx*Math.sqrt(2),dy*Math.sqrt(2));
       r.fill=MyColor.BLUE;
       r.draw(gc);

        MyOval o1 = new MyOval(dx-(dx*Math.sqrt(2))/2,dy-(dy*Math.sqrt(2))/2,
dx*Math.sqrt(2),dy*Math.sqrt(2));
        o1.fill= MyColor.DARKORANGE;
        o1.draw(gc);

        double lenx=(dx*Math.sqrt(2))/2;
        double leny=(dy*Math.sqrt(2))/2;
        MyRectangle r1 = new MyRectangle(dx-(lenx*Math.sqrt(2))/2,dy-
(leny*Math.sqrt(2))/2,lenx*Math.sqrt(2),leny*Math.sqrt(2));
        r1.fill= MyColor.GREEN;
        r1.draw(gc);

        MyOval o2 = new MyOval(dx-(lenx*Math.sqrt(2))/2,dy-
(leny*Math.sqrt(2))/2,lenx*Math.sqrt(2),leny*Math.sqrt(2));
        o2.fill= MyColor.RED;
        o2.draw(gc);
}
```

Full Java Code Developed:

Classes Imported

```
import javafx.scene.paint.Color;
import java.lang.Math;
import javafx.application.Application;
import javafx.scene.Group;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
```

Enum MyColor:

```
package application;
import javafx.scene.paint.Color;
public enum MyColor{
        RED (255,0,0,1),
        GREEN (0,255,0,1),
        BLUE (0,0,255,1),
        BLACK (0,0,0,1),
        DARKKHAKI (189, 183, 107, 1),
        DARKORANGE (255,140,0,1)
        ;


        private int red;
        private int green;
        private int blue;
        private double opacity;

        MyColor(int red, int green, int blue, double opacity){
                this.red=red;
                this.blue=blue;
                this.green=green;
                this.opacity=opacity;
        }

        public Color myColor() {
                return Color.rgb(red, green, blue, opacity);
        }


        public String toString(){
                return String.format("#%02x%02x%02x", red, green, blue);
        }


}
```

Class MyPoint:

```java
package application;
import java.lang.Math;

public class MyPoint {
        private double x;
        private double y;

        MyPoint(double x, double y) {
            this.x=x;
            this.y=y;
        }

        void setX(double x) {
                this.x=x;
        }

        void setY(double y) {
                this.y=y;
        }

        double getX() {
                return x;
        }
        double getY() {
                return y;
        }

        void shiftX(double x) {
            this.x+=x;
        }
        void shiftY(double y) {
                this.y+=y;
        }
        public String toString() {
                return "MyPoint [x = "+x+", y = "+y+"]";
        }

        //distance from P(x,y) to some other point x2,y2
        void getDistance(double x2, double y2){
                System.out.println(Math.sqrt(Math.pow(x2-this.x,2)+Math.pow(y2-
                this.y,2)));
        }
          //angle of line from the +x counterclockwise
        void getAngle(double x2, double y2) {
                  double m=((y2-this.y)/(x2-this.x));
                  System.out.println(Math.atan(m));

        }

}
```

Class MyShape:

```
package application;
import javafx.scene.canvas.GraphicsContext;

public class MyShape {
	//MyPoint,MyColor
	MyPoint p= new MyPoint(0,0);
	MyColor fill=MyColor.BLACK;

	//area, perimeter

	double getArea() {
		return 0;
	}

	double getPerimeter() {
		return 0;
	}

	//toString

	public String toString() {
		return "";
	}
	//draw

	void draw(GraphicsContext gc) {
	}
}
```

Class MyRectangle:

```
package application;
import javafx.scene.canvas.GraphicsContext;

public class MyRectangle extends MyShape {
        private double width=0;
        private double height=0;

        //constructor
        MyRectangle(double x, double y, double w, double h){
             this.width=w;
          this.height=h;
          p.setX(x);
          p.setY(y);
        }

        //getP, getWidth, getHeight
        String getP() {
               return p.toString();
        }
        double getWidth() {
               return width;
        }
        double getHeight() {
               return height;
        }

        //area, perameter
        @Override
        double getArea() {
                    return (width*height);
             }
        @Override
        double getPerimeter() {
                    return (2*width+2*height);
             }
        //toString
        @Override
        public String toString() {
               return "Rectangle[x="+p.getX()+", y="+p.getY()+", width="+width+",
height="+height+", fill="+fill.toString()+", parameter="+(2*width+2*height)+",
area="+(width*height)+"]";
        }

        @Override
        void draw(GraphicsContext gc) {
               gc.strokeRect(p.getX(), p.getY(), this.width, this.height);
               gc.setFill(fill.myColor());
               gc.fillRect(p.getX(), p.getY(), width, height);
        }
}
```

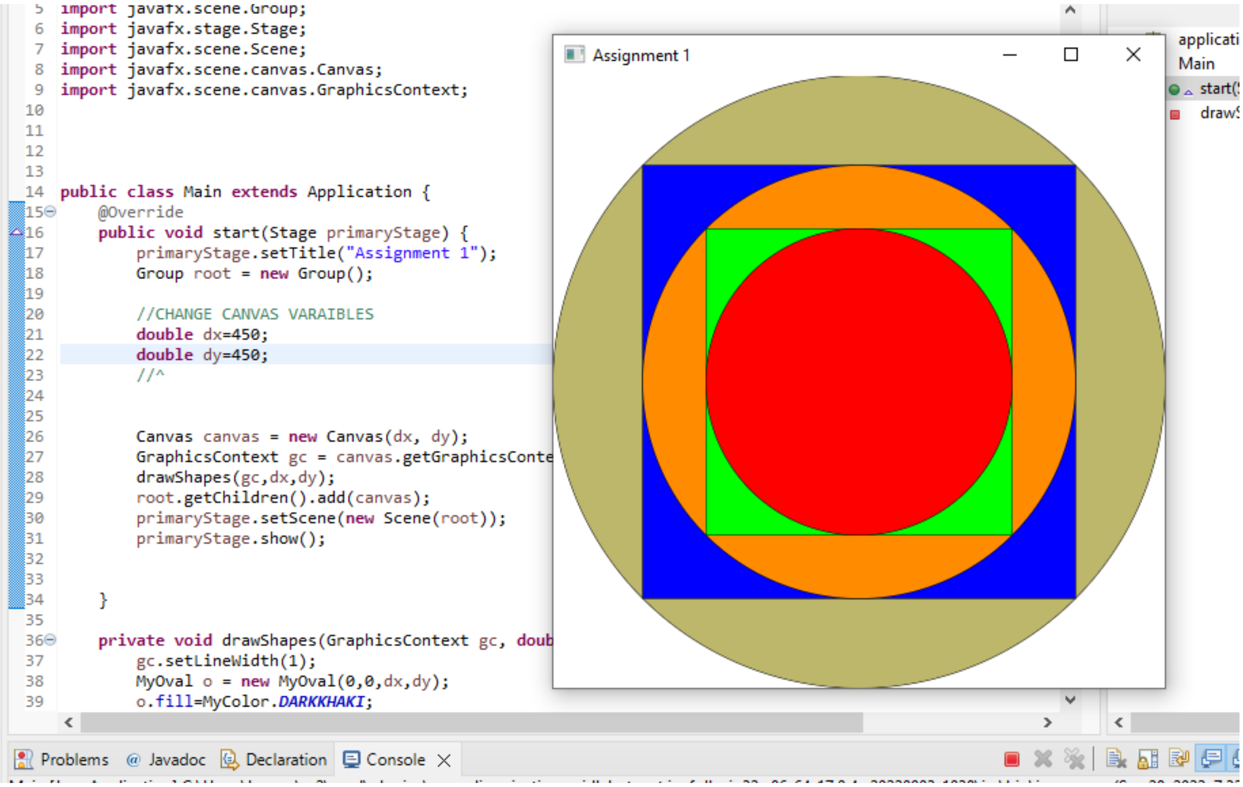
Class MyOval:

```
package application;
import javafx.scene.canvas.GraphicsContext;

public class MyOval extends MyShape {
        private double width;
        private double height;
        //constructor
        MyOval(double x, double y, double w, double h){
               p.setX(x);
               p.setY(y);
               this.width=w;
               this.height=h;
        }

        //area, perimeter
        @Override
        double getArea() {
                    return (Math.PI*(width/2)*(height/2));
        }

        @Override
        double getPerimeter() {
                    return
(2*Math.PI*(Math.sqrt((Math.pow(height/2,2)+Math.pow(width/2,2))/2)));
             }

        //getX, getY, getA, getB

        //getX(center)
        double getCenterX() {
               return p.getX()+(width/2);
        }
        //getY,(center)
        double getCenterY() {
               return p.getY()+(height/2);
        }

        //toString
        @Override
        public String toString() {
               return "Oval[x="+p.getX()+", y="+p.getY()+", width="+width+",
height="+height+", fill="+fill.toString()+", parame-
ter="+2*Math.PI*(Math.sqrt((Math.pow(height/2,2)+Math.pow(width/2,2))/2))+",
area="+Math.PI*(width/2)*(height/2)+"]";
        }
      @Override
        void draw(GraphicsContext gc){
               gc.strokeOval(p.getX(),p.getY(),this.width,this.height);
               gc.setFill(fill.myColor());
               gc.fillOval(p.getX(), p.getY(), width, height);
        }
}
```

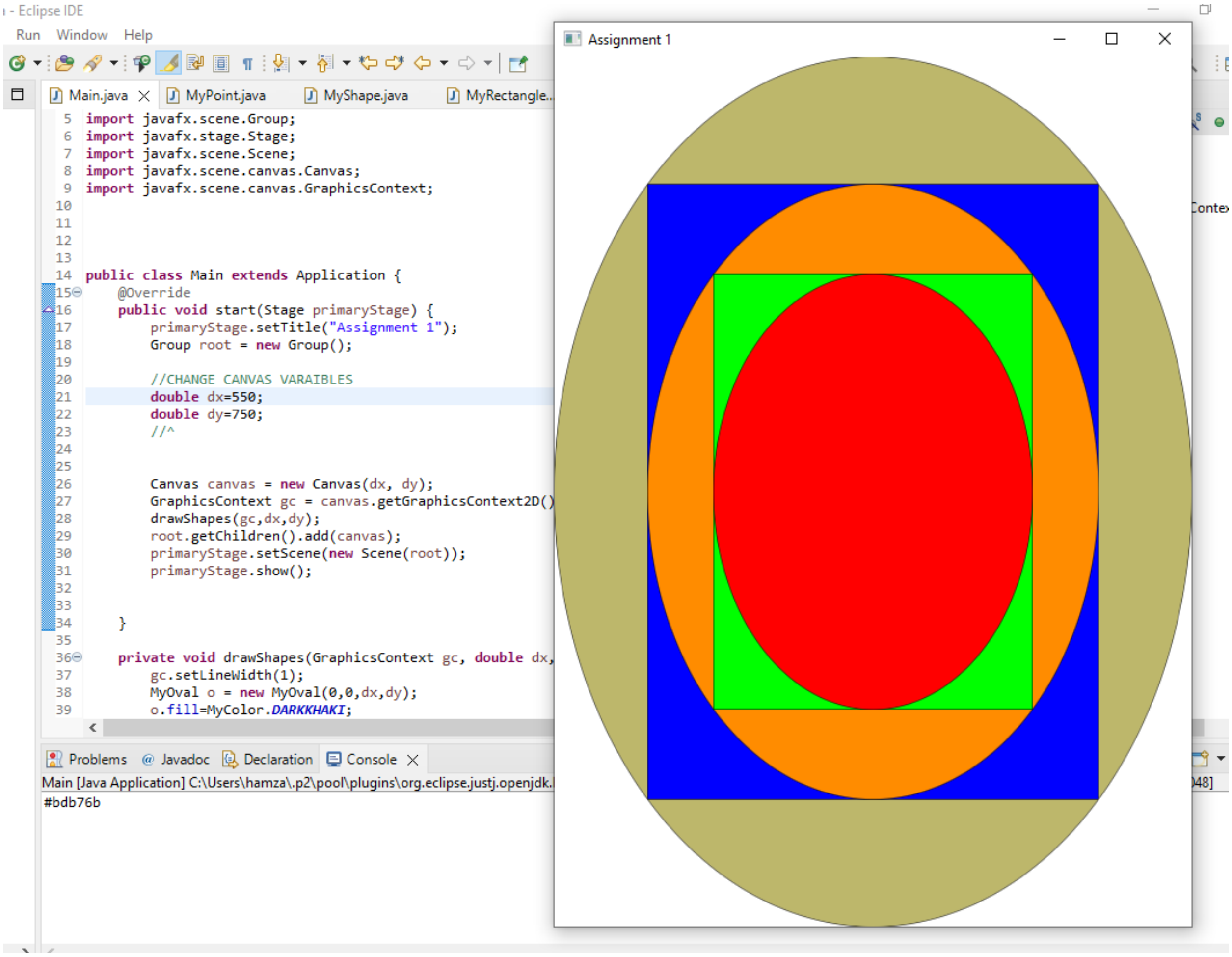
Class Main

```
package application;

import java.lang.Math;
import javafx.application.Application;
import javafx.scene.Group;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;

public class Main extends Application {
       @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Assignment 1");
        Group root = new Group();

        //CHANGE CANVAS VARAIBLES
        double dx=750;
        double dy=400;
        //^
        Canvas canvas = new Canvas(dx, dy);
        GraphicsContext gc = canvas.getGraphicsContext2D();
        drawShapes(gc,dx,dy);
        root.getChildren().add(canvas);
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
        }

    private void drawShapes(GraphicsContext gc, double dx, double dy) {
       gc.setLineWidth(1);
       MyOval o = new MyOval(0,0,dx,dy);
       o.fill=MyColor.DARKKHAKI;
       o.draw(gc);
       System.out.println(o.fill.toString());

       dx=dx/2;
       dy=dy/2;
       MyRectangle r = new MyRectangle(dx-(dx*Math.sqrt(2))/2,dy-(dy*Math.sqrt(2))/2,
dx*Math.sqrt(2),dy*Math.sqrt(2));
       r.fill=MyColor.BLUE;
       r.draw(gc);

        MyOval o1 = new MyOval(dx-(dx*Math.sqrt(2))/2,dy-(dy*Math.sqrt(2))/2,
dx*Math.sqrt(2),dy*Math.sqrt(2));
        o1.fill= MyColor.DARKORANGE;
        o1.draw(gc);

        double lenx=(dx*Math.sqrt(2))/2;
        double leny=(dy*Math.sqrt(2))/2;
        MyRectangle r1 = new MyRectangle(dx-(lenx*Math.sqrt(2))/2,dy-
(leny*Math.sqrt(2))/2,lenx*Math.sqrt(2),leny*Math.sqrt(2));
        r1.fill= MyColor.GREEN;
        r1.draw(gc);
        MyOval o2 = new MyOval(dx-(lenx*Math.sqrt(2))/2,dy-
(leny*Math.sqrt(2))/2,lenx*Math.sqrt(2),leny*Math.sqrt(2));
        o2.fill= MyColor.RED;
        o2.draw(gc);
    }
}
```

Outputs Produced

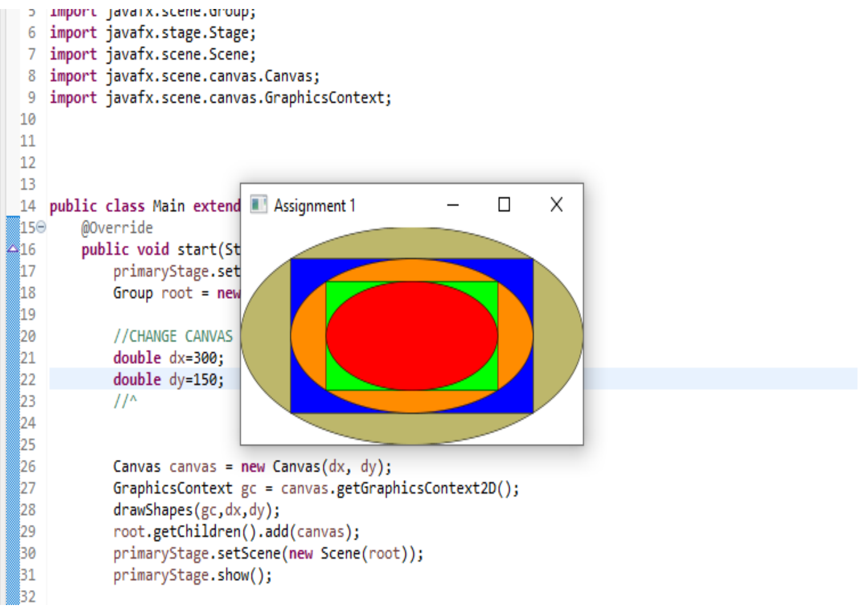


```
import javafx.scene.Group;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;




public class Main extends Application {
    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Assignment 1");
        Group root = new Group();

        //CHANGE CANVAS VARAIBLES
        double dx=450;
        double dy=450;
        //^


        Canvas canvas = new Canvas(dx, dy);
        GraphicsContext gc = canvas.getGraphicsConte
        drawShapes(gc,dx,dy);
        root.getChildren().add(canvas);
        primaryStage.setScene(new Scene(root));
        primaryStage.show();


    }

    private void drawShapes(GraphicsContext gc, doub
        gc.setLineWidth(1);
        MyOval o = new MyOval(0,0,dx,dy);
        o.fill=MyColor.DARKKHAKI;
```



```
import javafx.scene.Group;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;




public class Main extends Application {
    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Assignment 1");
        Group root = new Group();

        //CHANGE CANVAS VARAIBLES
        double dx=550;
        double dy=750;
        //^


        Canvas canvas = new Canvas(dx, dy);
        GraphicsContext gc = canvas.getGraphicsContext2D()
        drawShapes(gc,dx,dy);
        root.getChildren().add(canvas);
        primaryStage.setScene(new Scene(root));
        primaryStage.show();


    }

    private void drawShapes(GraphicsContext gc, double dx,
        gc.setLineWidth(1);
        MyOval o = new MyOval(0,0,dx,dy);
        o.fill=MyColor.DARKKHAKI;
```

```
Main [Java Application] C:\Users\hamza\.p2\pool\plugins\org.eclipse.justj.openjdk.
#bdb76b
```



```
import javafx.scene.Group;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;




public class Main extend
    @Override
    public void start(St
        primaryStage.set
        Group root = new

        //CHANGE CANVAS
        double dx=300;
        double dy=150;
        //^


        Canvas canvas = new Canvas(dx, dy);
        GraphicsContext gc = canvas.getGraphicsContext2D();
        drawShapes(gc,dx,dy);
        root.getChildren().add(canvas);
        primaryStage.setScene(new Scene(root));
        primaryStage.show();

```