Chengliang Tan

Exercise 1

Project report

The goal for this project is to build a java class hierarchy. In this class hierarchy MyShapewill be the superclass, MyLine, MyRectangle and MyOval is its subclasses. Also we have to build an enum type MyColor to fill the shapes and lines we draw with colors.

I built 6 files for this project. MyShape.java stores the superclass MyShape. MyLine.java stores the subclass MyLine. MyRectangle.java stores the subclass MyRectangle. MyOval.java stores the subclass MyOval. MyColor.java stores the enum type MyColor. Main.java is where I will draw the geometric configuration at the end of the exercise.

In MyShape.java, I imported the javafx.scene.canvas.GraphicsContext package for the draw method that is going to be outputting the shapes. I also imported javafx.scene.paint.Color to use colors. I created two integer variables x and y which represents the center coordinates of the shape object. And a Mycolor variable named “color” to store the color of the shape. Also I create a type Color variable named “c” so that I can output the color stored in “color” when I draw the shape. I created two constructors for the MyShape class, one is a default constructor where x and y are initialized to 0, color is initialized to MyColor.BLACK, and c is also initialized to Color.BLACK. The other constructor will create an object based on the user’s input. The constructor has 3 arguments: inX, inY, and inColor, which allow the user to choose the x, y coordinates and the color of the shape, respectively. The getX(), getY(), getColor() method returns the properties of the object. The setX(), setY(), setColor() method allows the user to change the properties of an existing object. The toString() method uses the get functions and returns a string that describes the x and y coordinates and the color of the shape. The draw() method uses GraphicsContext as its argument and once the user calls this method it will fill the whole canvas with the color that is stored in the color variable.

In MyLine.java, I imported the javafx.scene.canvas.GraphicsContext package for the draw method. An MyLine object is defined by 2 points, each point having its own x and y coordinates. Also I created a double variable length to store the length of the line, the length of the line can be obtained by the function Math.sqrt((X2-X1)^2 + (Y2-Y1)^2). The double variable angle stores the angle with the x-axis. The angle can be obtained by Math.*toDegrees*(Math.*atan*((Y2-Y1)/(X2-X1))). I used @Override to create the toString() and draw() methods. The toString() method uses the get functions and returns a string that describes the length and the angle of the MyLine object. The draw() method uses GraphicsContext as its argument and once the user calls this method it will draw a line on the canvas with the color that is stored in the color variable.

In MyRectangle.java, I imported the javafx.scene.canvas.GraphicsContext package for the draw method. A MyRectangle object is defined by its center point (x,y), the height h and width w, it’s color and a boolean variable that determines if the rectangle is filled. The int perimeter variable stores the perimeter of the rectangle which is defined by 2h + 2w. The int area variable stores the area of the rectangle which is defined by h\*w. The boolean variable filled is true if the rectangle is filled with color. In the constructors, I used super() to inherit the constructor I built in the superclass. I used @Override to create the toString() and draw() methods. The toString() method uses the get functions and returns a string that describes the height, the width, the perimeter and the area of the MyRectangle object. The draw() method uses GraphicsContext as its argument and once the user calls this method it will first check if filled is true or false. If true, it will draw a filled rectangle on the canvas with the color that is stored in the color variable. If false, it will draw an outline of a rectangle on the canvas with the color.

In MyOval.java, I imported the javafx.scene.canvas.GraphicsContext package for the draw method. A MyOval object is defined by its center point (x,y), the x-axes, the y-axes, its color and if it is filled. The h and w variables store the x and y axes. The boolean variable filled determines if the MyOval object is filled with color or not. In the constructors, I used super() to inherit the constructor I built in the superclass. I used @Override to create the toString() and draw() methods. The toString() method uses the get functions and returns a string that describes the axes length, the perimeter and the area of the MyOval object. The draw() method uses GraphicsContext as its argument and once the user calls this method it will first check if filled is true or false. If true, it will draw a filled oval on the canvas with the color that is stored in the color variable. If false, it will draw an outline of an oval on the canvas with the color.

The file MyColor.java stores the enum type MyColor. To build the enum type we first have to put a list of colors on the top, such as RED, GREEN, and BLUE. The default constructor initializes the rgb value to 0. The constructor sets the rgb value based on the user’s input. The get functions will return the red, green, and blue value of the color, respectively. The mixColor method allows user to input 2 colors, then the program will add up the red, green, blue values of the two colors respectively and then divide each of them by 2 to get the rgb values for the new mixed color.

To draw the geometric configuration at the end of the exercise, first can build a line extending from coordinate (0,0) to (canvas.getWidth(), canvas.getHeight).

Then we have to find the center of the canvas by diving the height and width of the canvas by 2. Once the center of the canvas is known, we can build 3 rectangles and 3 ovals centered at the center of the canvas and put them in proportional sizes according to the size of the canvas.