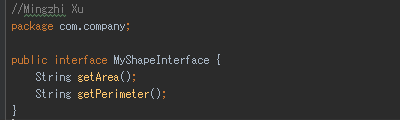
Mingzhi Xu

CSC 22100 Software Design Laboratory Fall 2018

Exercise 2

In this exercise, we are to create a hierarchy of Java classes MyShape, MyPolygon, and MyCircle from exercise 1, and we will be modifying these classes and adding MyRectangle and MyOval class in this exercise. We will also implement interfaces such that the MyShapeInterface interface will contain abstract method getArea which describes the area of an object in the class hierarchy and getPerimeter which describes the perimeter of an object in the class hierarchy.

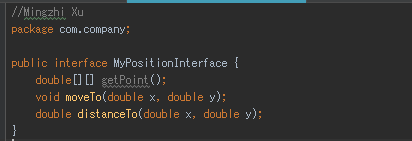
Code for interface MyShapeInterface



Since interface methods are public abstract there is no need to write it out since Java automatically initialize for you and since both methods describes an object, so the type would be String.

Another interface we will implement will be the MyPositionInterface which includes appropriate methods that describe the positional functions and behaviors of the specific object type of the class hierarchy. The interface includes public abstract methods getPoint which returns the point (x, y), moveTo which moves point (x, y) to the point (x + Δx, y + Δy), and distanceTo returns distance from point (x, y) to a point.

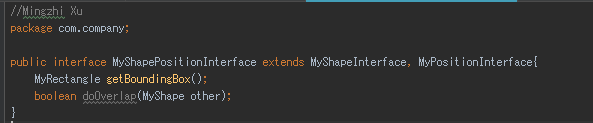
Code for interface MyPositionInterface



Since interface methods are public abstract there is no need to write it out since Java automatically initialize for you and getPoint returns (x, y) so I use a double array to store x points and y points, since moveTo moves the object and does not return anything it is type void and distanceTo returns a distance so I use double.

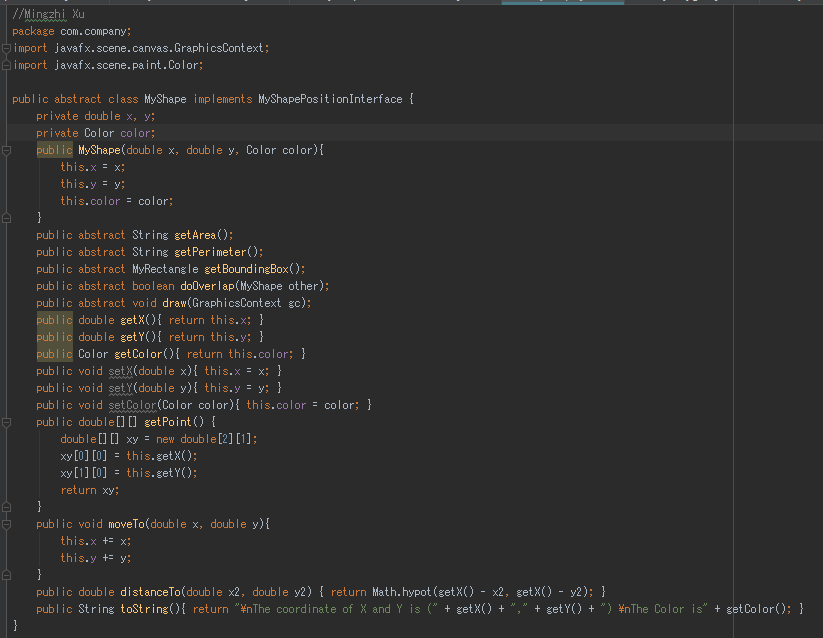
The third interface we will implement is MyShapePositionInterface which extends the interfaces MyShapeInterface and MyPositionInterface, and what this does is that the so the public abstract methods in the previous two interfaces can be declared and used later in the class hierarchy. MyShapePositionInterface will contain methods getBoundingBox which returns a bounding rectangle of an object in the class hierarchy, and doOverlap which returns true or false if two objects in the class hierarchy overlap.

Code for interface MyShapePositionInterface



Since interface methods are public abstract there is no need to write it out since Java automatically initialize for you and getBoundingBox is suppose to return a bounding rectangle so I declare it as a MyRectangle type which will be created later, and doOverlap will compare two MyShape objects to check if their bounding box overlaps and returns true or false so the type is Boolean.

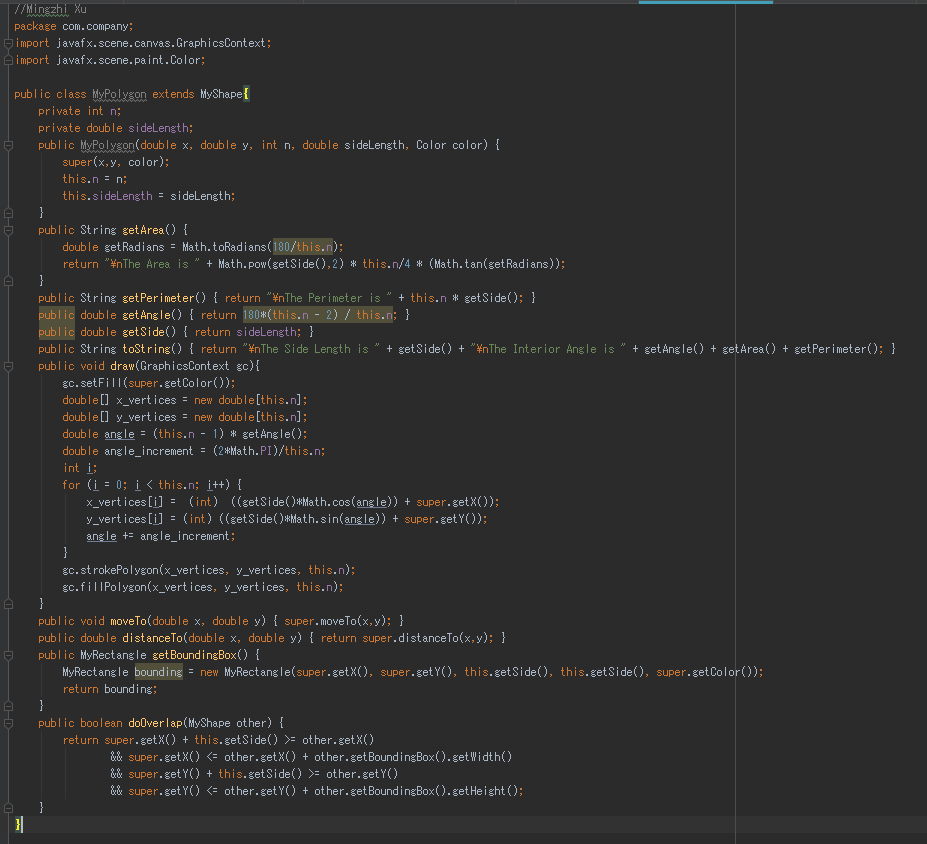
Next, we will modify the MyShape class that was used in exercise 1 but MyShape class will be an abstract super class in this exercise so it contains abstract methods that will needed to be override later in the class hierarchy.

Code for MyShape abstract class

Here we can see that most of the methods are like exercise 1 but since MyShape is an abstract class so it can’t instantiate objects, so we must declare methods such as draw, getBoundingBox, doOverlap, getArea, and getPerimeter as abstract methods which to be override later in the class hierarchy.

Then we will modify the class MyPolygon from exercise 1 and we will be overriding abstract methods from the interfaces.

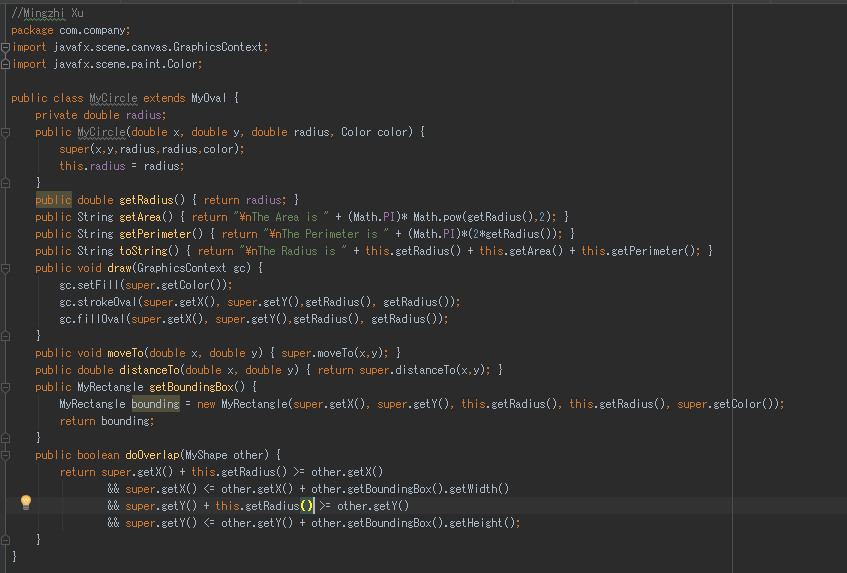
Code for MyPolygon class



Here we can see that most of the methods are unchanged from exercise 1 but the new methods that are needed to override requires the MyRectangle class which is used to create the bounding box and to be used in the overlap method to demonstrate polymorphism.

Next, we will look at the class MyCircle which was used in exercise 1 but this time we will change up the class hierarchy a bit and instead of MyCircle extending MyShape, MyCircle will be extending MyOval which will be a new class that extends MyShape.

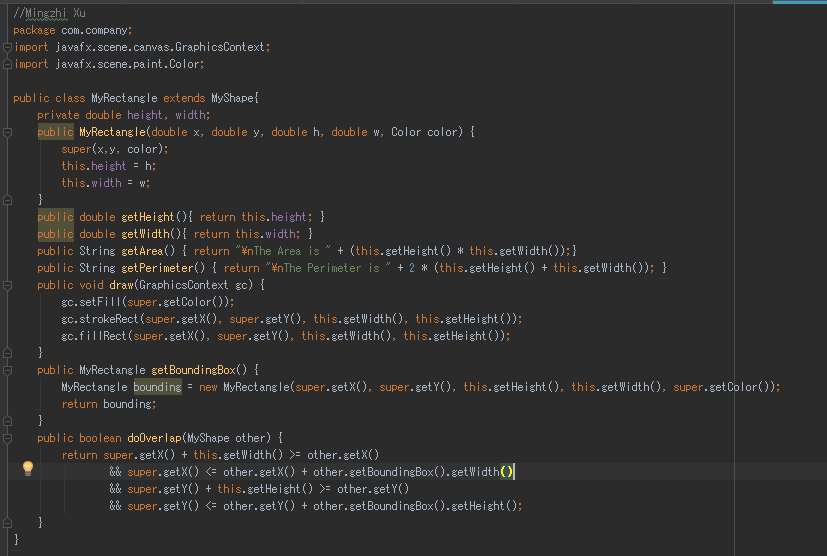
Code for MyCircle class



Here we can see that most of the methods are unchanged from exercise one and similarly to MyPolygon, we will be overriding abstract methods for MyCircle.

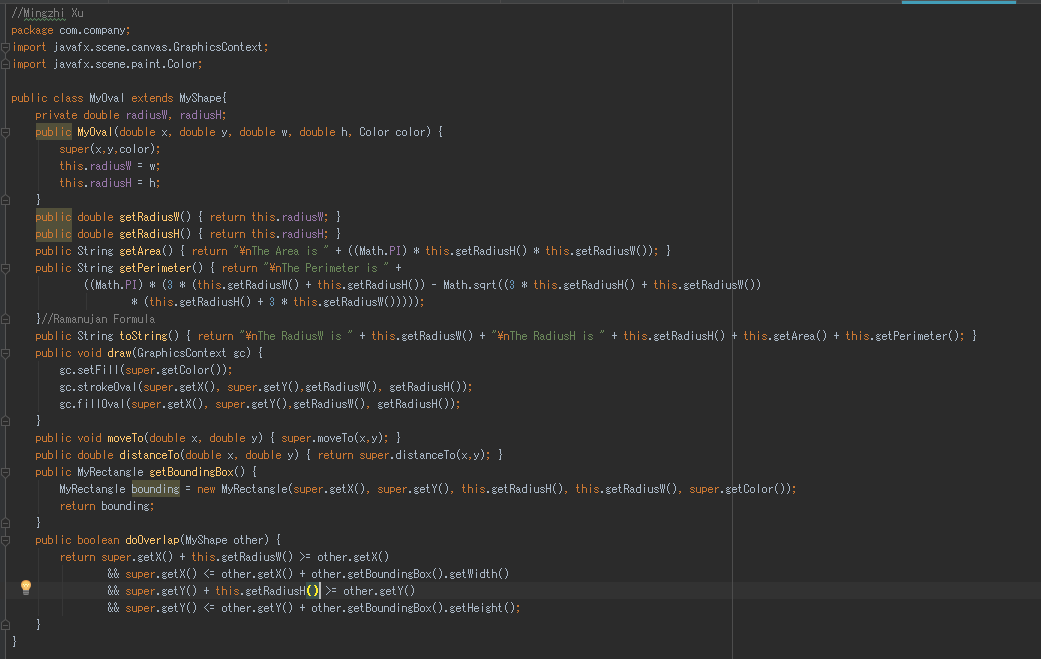
Now we will be creating the new class MyRectangle that will be added to class hierarchy which extends MyShape. The class MyRectangle is like MyPolygon but instead of having equal length on each side, a MyRectangle object will have its own width and height and only have four sides.

Code for MyRectangle class



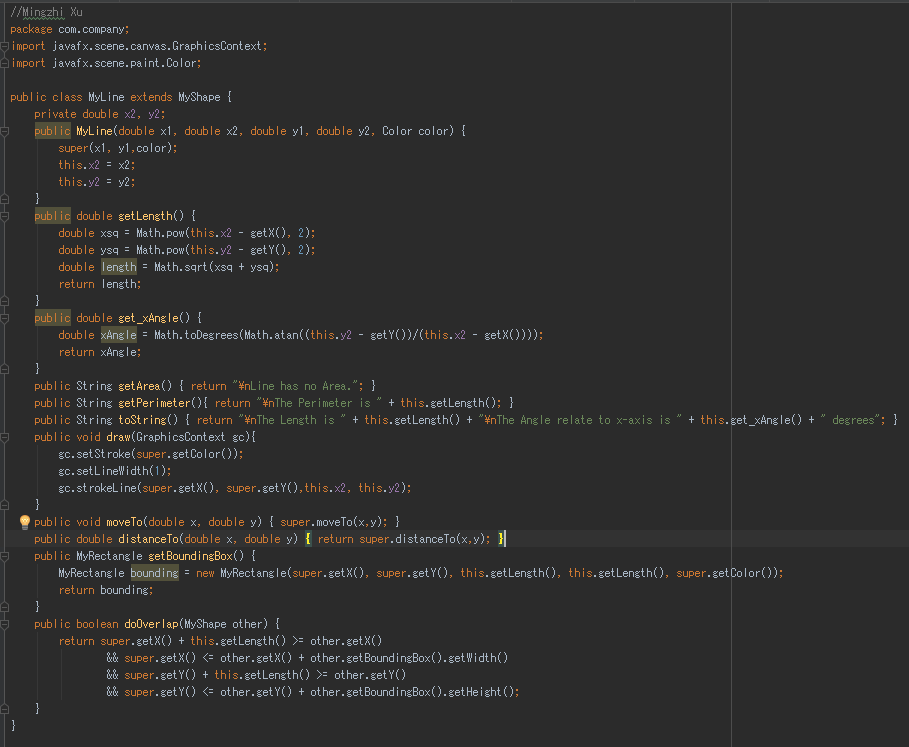
Lastly, we will be creating the new class MyOval that will be added to class hierarchy which extends MyShape. The class MyOval is like MyCircle but instead of having equal height and width radius, MyOval object have its own width radius and height radius.

Code for MyOval class

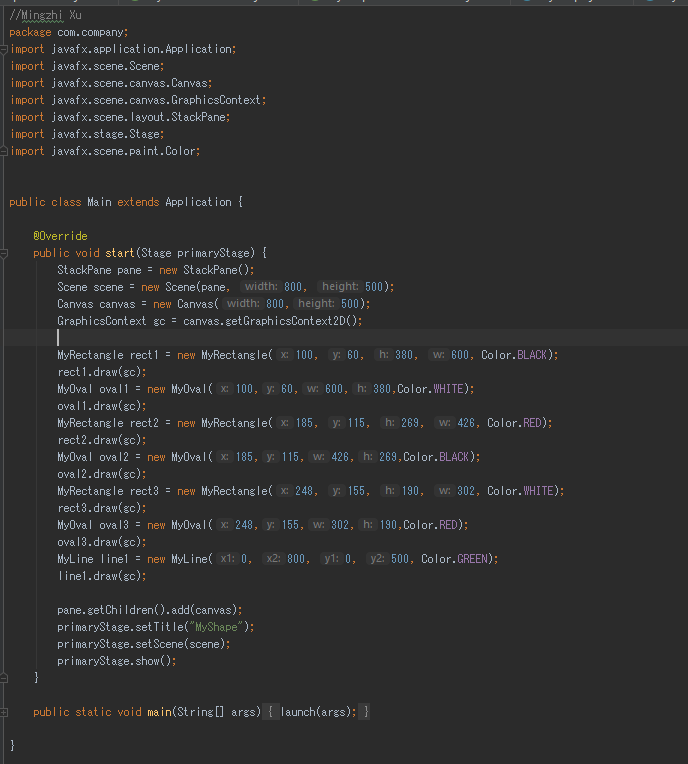


Here we can see that most of the method are like MyCircle class but since MyOval has a different width radius than height radius, we can use the Ramanujan formula to determine the perimeter.

We will also have the class MyLine but nothing much have changed from exercise 1.

Code for MyLine class

Finally, we will draw rectangles and ovals in the main class. We can determine the width and height of the inner rectangle using a ratio of half of the width of the outer rectangle over square root 2 and half of the height of the out outer rectangle over square root 2. There won’t be much change on the main class since we are just drawing different shapes compared to exercise 1.

Code for main class

Here is the result of drawing those rectangles and ovals.