Chengliang Tan

Exercise 2 report

For the MyCircle subclass, I inherited the variables x, y, and color from the MyOval class. Additionally, I created the double variables radius, perimeter, and area and the boolean variable filled to define the characteristics of a MyCircle object. In the constructor, I used the super keyword to call the constructor from MyOval class, the input arguments for the MyCircle constructor are double inX, double inY, double inRadius, MyColor inColor, and boolean isFilled . I used the super keyword to call MyOval constructor inside the MyCirlce construtor, where both the width and height will be set as the input inRadius. To build a MyCircle object, users have to use the constructor to input the center point (x,y), the radius, the color, and if the circle is filled. The function getArea() calculates and returns the area of the object using the equation *area = Math.PI*(getRadius()*getRadius())*. The function getPerimeter() calculates and returns the perimeter of the object using the equation *perimeter = 2*Math.PI*getRadius()*. To draw the circle, we need to use the fillOval() function which takes the upper left x and y coordinate of the oval and the height and width of the oval. In the circle case, the height and width would be the same. Also, we have to check the filled variable to determine if the object is being filled. The getPoint() method returns a double array that stores the x and y value of the object in the 0 and 1 index of the array, respectively.

The moveTo() method takes the dx and dy as arguments and moves the point(x,y) of any MyShape object to the point (x+dx, y+dy). The distanceTo() method takes x and y as arguments and calculates the distance between the center point of the MyShape object and the point being inputted using the distance formula.

For the MyPolygon subclass, I inherited the variables x, y, and color from the MyShape superclass. Additionally, I created the integer variable N, double[] array xPoints and yPints, double variable radius, perimeter, area, sideLength, and interAngle and the boolean variable filled to define the characteristics of a MyPolygon object. To build a MyPolygon object, users have use the constructor and input the center point (x,y), the radius, the color, the number of sides N, and if the circle is filled. The method getSideLength() returns the side length of the polygon object using the formula *sideLength = 2*getRadius()*Math.sin(Math.PI/getN())* . The method getPerimeter(0 then uses getSideLength() and multiply it by N to get the perimeter of the polygon object. The getInteriorAngle() method uses the equation *interAngle = ((getN()-2)/getN())*180* to calculate the interior angle of the polygon object. The getArea() method uses the equation *area = (1/2)*getN()*getRadius()*getRadius()* to calculate the area of the MyPolygon object. The method getXPoints() returns the array xPoints with N-1 length and use a for loop from 0 to N and the formula *xPoints[i] = getX() + getRadius()*Math.sin((2*Math.PI*i)/getN())* to fill the array with the x coordinates of the polygon object. The method getYPoints() returns the array yPoints with N-1 length and use a for loop from 0 to N and the formula *yPoints[i] = getY() + -1*(getRadius()*Math.cos((2*Math.PI*i)/getN()))* to fill the array with the x coordinates of the polygon object. To draw the MyPolygon object, I first use the fillOval() method to draw a circle that inscribes the polygon object, then I take the fillPolygon() method and input xPoints[], yPoints[], and N as the argument to draw the polygon object.

I implemented three methods inside the interface MyPositionInterface. The getPoint() method returns the center point(x,y) for any object that calls this method. In each subclass the method getPoint() will be overridden. The moveTo() method takes double variables dx and dy as input and move the center point of the object to the point(x+dx, y+dy). The moveTo() method will also be overridden by every subclass. The distanceTo() method takes double variables x and y as input and returns the distance between the center point of an MyShape object and the point(x,y) using the distance formula. The distanceTo() method will also be overridden by every subclass.

I implemented the interface MyShapePositionInterface to extend the MyPositionInterface and overridden all three methods in the Position Interface. The MyShapePositionInterface contains two new methods: getMyBoundingBox() and doOverlap().

The method MyBoundingBox(0 will be overridden by every subclass. For the MyRectangle class, this method will return null. For the MyLine class, this method will return a rectangle that takes its x1 and y1 values as the upper left x and y points, and using the length of the line as the hypotenuse we can calculate the width and height of the rectangle. For the MyOval class the method will return a rectangle that has the same center point and same height and width as the MyOval object. For the Mycircle class, The getMyBoundingBox() method returns a rectangle that has the same center point and with height and width equal to the radius of the MyCircle object. For the MyPolygon class, The MyBoundingBox() method creates 4 variables, minX, maxX, minY, and maxY. Then I used a for loop to go through the xPoints[] and yPoints[] array and find the minimum x and y and maximum x and y values by comparison.

Then I created a MyRectangle object using minX and minY as its upper left x and y points and maxX - minX as its width and maxY - minY as its height.

There are 10 different cases for the doOverlap() method, which are rectangle vs rectangle, rectangle vs oval, rectangle vs circle, rectangle vs polygon, oval vs oval, oval vs circle, oval vs polygon, circle vs circle, circle vs polygon, and polygon vs polygon. For each of the 10 cases, I compared the difference between the center point of two input shapes with their radius(or height and width) to check if their radius(or height and width) is greater than the difference between their center point. If yes, the method returns true, if the no, then returns false.