ENSF 409

Principles of Software Development

Term Project
Milestone 2

Student Names: Harsohail Brar, Ryan Holt, Gary Wu
Section: B01
Date: Apr. 5, 2019

In Lab Exercise 1:

In Lab Exercise 2:

Drawing.java

```
import java.util.Iterator;
import java.util.LinkedHashSet;

public class Drawing {
	LinkedHashSet<Line> lines;

	public void drawPolygon(LinkedHashSet <Line> lines){
		this.lines = lines;
		Polygon p = new Polygon(lines);
	System.out.println(p);
	System.out.printf("The perimeter of the polygon %d is %.2f: \n" , Polygon.classID(),
perimeter(p));
	}

	private double perimeter(Polygon p){
		Iterator<Line> it = p.getLine();
		double perim = 0;
		while(it.hasNext()){
			perim += it.next().distance();
		}
		return perim;
	}

	public static void main(String[] args) {
		Drawing drawing = new Drawing();

		Point [] points = {
					new Point(20,30), new Point (50, 100), new Point (105, 30),
					new Point(120,130), new Point (150, 200), new Point (200, 130),
				new Point(320,330), new Point (250, 400), new Point (400, 330)
					};

		Line [] lines = {
					new Line(points[0], points[1]),
					new Line(points[1], points[2]),
					new Line(points[2], points[0]),
					new Line(points[3], points[4]),
					new Line(points[4], points[5]),
					new Line(points[5], points[3]),
					new Line(points[6], points[7]),
					new Line(points[7], points[8]),
```

```
                                        new Line(points[8], points[6])
                                        };

        LinkedHashSet<Line> poly1 = new LinkedHashSet<Line>();
        poly1.add(lines[0]);
        poly1.add(lines[1]);
        poly1.add(lines[2]);

        drawing.drawPolygon(poly1);

        LinkedHashSet<Line> poly2 = new LinkedHashSet<Line>();
        poly2.add(lines[3]);
        poly2.add(lines[4]);
        poly2.add(lines[5]);

        drawing.drawPolygon(poly2);

        LinkedHashSet<Line> poly3 = new LinkedHashSet<Line>();
        poly3.add(lines[6]);
        poly3.add(lines[7]);
        poly3.add(lines[8]);

        drawing.drawPolygon(poly3);
        }

}
```

Line.java

```
class Line {

        Point start, end;
        private static int classID = 0;
        private int objID;

        public Line(Point a, Point b) {
        start = a;
        end = b;
         objID = ++ classID;
        }


        public double distance(){
           return Point.distance(start, end);
   }

   public String toString()
   {
      // THIS METHOD DOESN'T WORK. AS PART OF EXERCISE-2 STUDENTS MUST FIX IT
      // TO RETURN A STRING WITH TWO COORDINATES OF THE START AND END POINTS
      // OF A LINE, IN THE FORMAT SHOWN IN THE EXAMPLE BELOW:
```

```
        // Line 1: starts at (20, 30), and ends at (50, 100)
        //
            return "starts at " + start.toString() + ", and ends at " + end.toString();
    }
}
```

Point.java

```
class Point {
            private int x, y;

            public Point(int x, int y) {
            this.x = x;
            this.y = y;
            }

            static public double distance(Point a, Point b){
                        double diffx = a.x - b.x;
                        double diffy = a.y - b.y;
                        return Math.sqrt(diffx * diffx + diffy * diffy);
            }

            public String toString(){
        // THIS METHOD DOESN'T WORK. AS PART OF EXERCISE-2 STUDENTS MUST FIX IT
        // TO RETURN A STRING WITH THE COORDINATES OF A POINT IN THE FORMAT
SHOWN
        // IN THE EXAMPLE BELOW:
        // (20, 30)
                        return "(" + Integer.toString(x) + ", " + Integer.toString(y) + ")";
            }
}
```