COS226 Week 3 Activity

For this handout we will use Point2D.java which is located on the last page and here: http://algs4.cs.princeton.edu/25applications/Point2D.java.html

1. *Static Comparators.*

   (a) Suppose you use: `Arrays.sort(pts);` where `pts` is an array of Point2D objects. Which comparison method in Point2D is used?

   (b) Write a Java code fragment that instead sorts with the same result using one or more of the static comparators defined in `Point2D (X_ORDER, Y_ORDER, R_ORDER)`

2. *Dynamic Comparators.*

   (a) What is the difference between a `Comparable` and a `Comparator`?

   (b) What is the difference between a static and a dynamic `Comparator`?

(c) Suppose we want to create a Comparator that compares two points based on their distance from some third point, call it w. Fill in the code below. You may find the Point2D provided at the end of this handout useful.

```
public static class DistanceComparator implements Comparator<______________> {
    Point2D ________;

    public DistanceComparator(_____________) {
        _________ = __________;
    }

    public int compare(____________ p, ____________ q) {
        double distToP = p.distanceTo(_______);
        double distToQ = q.distanceTo(_______);
        if (distToP < distToQ) return ____;
        if (distToP > distToQ) return ____;
        return ____;
    }
}
```

(d) (Optional; if time) Now suppose we want to use our comparator to sort a list of Points called by their distance from the origin. Fill in the code below to accomplish this task.

```
Point2D[] points = getRandomPoints();

Point2D origin = new Point2D(___, ___);

Comparator<Point2D> originComp = ________________________________________;

Arrays.sort(points, _______________________);
```

3. (Optional; if time) *3-way Merge sort* is a modification of the merge sort algorithm that considers 3 “equal” sub arrays instead of 2 sub arrays.

   (a) Given 3 sorted sub arrays of size N/3, how many comparisons are needed to merge them to a sorted array of size N. Provide your answer in tilde notation.

   (b) Argue that number of compares to sort an array of size N using 3-way merge sort is still linearithmic.

   (c) Given a choice, would you choose 3-way or 2-way merge sort? Justify your answer.

This is a subset of the Point2D class found on the booksite.

```
import java.util.Comparator;
import java.util.Arrays;
public class Point2D implements Comparable<Point2D> {
                                                                                  //compare by
    public static final Comparator<Point2D> X_ORDER = new XOrder(); // x co-ord
    public static final Comparator<Point2D> Y_ORDER = new YOrder(); // y co-ord
    public static final Comparator<Point2D> R_ORDER = new ROrder(); // polar radius

    private final double x;    // x coordinate
    private final double y;    // y coordinate

    public Point2D(double x, double y) {
        if (Double.isInfinite(x) || Double.isInfinite(y))
            throw new IllegalArgumentException("Coordinates must be finite");
        if (Double.isNaN(x) || Double.isNaN(y))
            throw new IllegalArgumentException("Coordinates cannot be NaN");
        if (x == 0.0) x = 0.0;  // convert -0.0 to +0.0
        if (y == 0.0) y = 0.0;  // convert -0.0 to +0.0
        this.x = x;
        this.y = y;
    }

    public int compareTo(Point2D that) {
        if (this.y < that.y) return -1;
        if (this.y > that.y) return +1;
        if (this.x < that.x) return -1;
        if (this.x > that.x) return +1;
        return 0;
    }


    // compare points according to their x-coordinate
    private static class XOrder implements Comparator<Point2D> {
        public int compare(Point2D p, Point2D q) {
            if (p.x < q.x) return -1;
            if (p.x > q.x) return +1;
            return 0;
        }
    }

    public static void main(String[] args) {
        Point2D[] points = SomeOtherClass.getPoints();
        Arrays.sort(points, Point2D.X_ORDER);
    }
}
```