C212/A592 Spring 17 Lab 4

Intro to Software Systems

Instructions:

- Review the requirements given below and Complete your work. Please submit all files through Canvas.
- The grading scheme is provided on Canvas

Point Accumulator

1. Create a class named PointAccumulator – you will be reusing your Point class from Assignment 1.
2. Here is the API for your PointAccumulator class:

```
public class PointAccumulator {
        Point[] points; // An array of type point

        public PointAccumulator() { } // constructor

        public String run() { } // populates points array from user input

        public static void main(String[] args) { }
}
```

   NOTE: *You can create as many private helper methods as you want, but you should not have any more public methods.*

   Programming tips: Remember to test and compile often!! This is an extremely powerful technique. Complete a step, then test it. Iterative testing will make for a much smoother programming experience.

   Break the assignment down into steps. For example, at first just worry about getting user input. Complete that step then test it (and test along the way). Then work on *just 1* of the next steps, and test. Optimizing early and writing to much code before testing does not end well.

3. Here are the main steps for this assignment:
   a. Prompt user to input the number of points they wish to store
      i. The *run()* method is where you want to prompt the user and populate the points array with user input.
      ii. Check for valid input of an integer, if not, return from the run method with value “invalid input” e.g.,

```
final int N; // final means variable can't be changed once initialized
if (scan.hasNextInt()) {
        // process input (assign a value to N from the scanner)
} else {
        return "invalid input";
}
```

iii. Initialize points array to this size: e.g. points = new Points[N]

b. Once you have the array points initialized, then prompt to enter the *x* and *y* coordinate on the same line. You will then be creating a new Point with that *(x,y)* value and storing it in your array.
   i. You will want to create a second scanner for collecting the points
   ii. Assume for each line, the input in the scanner is: 2 3
   So just the *x* then *y* value with a space, and no other characters

Hints: It is always a good programing habit to choose a for loop over a while loop if possible (if you know the number of times your loop should run). In this case we know how many times, that is N number of times from previous user input. For each iteration of the for loop, you can read the entire line with the scanner, then parse out the *x* and *y* values from that line.

If using *scan.nextLine(),* a String will be returned.

You can use a combination of *Integer.parseInt()* and *charAt()* since you know the *x coordinate* is at position 0 and the *y* coordinate is at position 2.

You could also look in into the String class split method. It would look something like:

```
String line = scan.nextLine();
String[] coords = line.split(" ");
```

the *x* and *y* coordinate will end up at position 0 and 1 respectively in the array

c. Now you will have an array filled with points. We will be getting 2 pieces of information from these points. (this is where you will want to have some private helper methods, probably 1 for each step)
   (1) The point furthest from the origin
   (2) The two points with the greatest distance

d. To find the two points with the greatest distance, you will need to use a doubly nested for loop. It will compare the first points to the rest of the points, then the second point to the rest of the points, then third point to the rest, and so on until exhausted.
e.g.

```
int maxDistance = 0;
// index into array of points furthers from each other
int p1 = 0;
int p2 = 0;
for (some loop condition) {
        for (some loop condition) {
                // compare points[i] to points [j]
                // update maxDistance p1, and p2 if necessary
        }
}
```

e. I used the following 5 points for a test case:
   (0, 6)
   (1, 1)
   (5, 0)
   (1, -1)
   (-1, -2)

   My solution printed:

```
Point furthest from origin is: (0.0, 6.0)
Points with furthest distance are: (0.0, 6.0) (-1.0, -2.0)
```