Lab 11: Max Disc Cover, Revisited

Submission deadline: 2359, Friday, November 17, 2017.

(If you need a deadline extension, you can submit before Sunday, November 19, with no penalty).

Setup

The skeleton code from Lab 11 is available on `cs2030-i` under the directory `~cs2030/lab11`.

Maximum Disc Coverage

For our final lab of this semester, I think it is apt that we revisit our first lab on maximum disc coverage and resolve the problem. You will perhaps realize that how much that way you code have changed in 12 weeks!

It might be useful at this point to re-read the problem statement for Lab 1 [../lab1/index.html].

This time, the `Circle` and `Point` are given to you, so you do not need to fill in any blanks in these two classes.

Things that we have covered since the first lab include exceptions, collections, and code without side effects. Some minor changes have been made to `Circle` and `Point` to reflect these:

- The `Circle` and `Point` are both immutable now. So `moveTo` returns a new object instead of modifying the existing one. No side effects! Some of you got

your Lab 1 wrong because this side effect.

- If the `Circle` constructor takes in two points that coincide or too far apart, then, instead of returning a circle with `NaN` as radius, which is silly, we simply throw an `IllegalArgumentException` as we should!
- The `Circle` constructor does not take in a flag that indicates if the center falls on the left or right. It always return the circle with the center on the left. To get "the other" circle, we simply swap the order of the two points. This little change will make our code simpler and cleaner!

In the main program, we now use Java Collection (a `List`) instead of raw array to store the points. There is no longer a need to know in advanced how many points are there in the input file (recall we had to put the number of points as the first line of input).

The code that uses scanner to read from the inputs are now written using streams. No more for loops or boiler plate code!

Max Disc Coverage on Bus Stops

In this lab, we will use the class `MaxDiscCover` to find out, in Singapore, how dense is the densest clusters of bus stops? We will treat each bus stop as a point on 2D plane (recall that we have their longitude and latitude information). As such, each `BusStop` is now a subclass of `Point`.

Note that, `MaxDiscCover` does not know anything about bus stops -- it just solves the problem on a set of points. The problem now takes in radius of the circle to cover the points with (instead of always having a circle of unit distance). We set the radius of the circle to the degree-equivalent of 2.5-minute walking distance, so that any bus stops contained within the circle is within 5-minute walking distance away.

In other words, we use `MaxDiscCover` to solve the following problem: Given a set of bus stops with their location on 2D plane [1][#fn:1], and a circle with a given radius, we want to place the circle so that it covers as many bus stops as possible. What is the maximum number of bus stops that we can cover with the disc at any one time? In

other words, what is the maximum size of a cluster of bus stops that are within 5-minute walking distance from each other?

Your task is to fill in the `solve` method in `MaxDiscCover`. *Solve it using `Stream` instead of a double for loop like we did in Lab 1.* You may find the `product` method that we have written for Exercise 5 [../exercise5/index.html] useful.

Your program should output one number that shows how many bus stops are there in the circle that maximally cover the bus stops:

```
java LabEleven bus-stops.csv
13
```

(For your own curiosity, you can find out where the bus stops are located -- but that is not required for this lab).

Marking Scheme

You get 4 marks for correct answer. This contributes another 4 marks (out of 100) towards your final grade.

Marks will be deducted according to severity of the bugs.

Submission

When you are ready to submit your lab, on `cs2030-i`, run the script

```
~cs2030/submit11
```

which will copy all files matching `*.java` (and nothing else) from your `~/lab11` directory on `cs2030-i` to an internal grading directory. We will test compile and test run with a sample input to make sure that your submission is OK.

You can submit multiple times, but only the most recent submission will be graded.

---

1. The Earth is flat.