CS2030 Lab #1 (Question)

Tags & Categories

Tags:

Categories:

Related Tutorials

Task Content

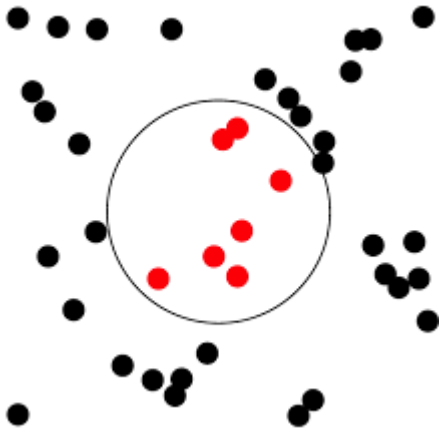
Maximum Disc Coverage

Topic Coverage

- Basic Java syntax and semantics
- Object-oriented principles: abstraction and encapsulation

Problem Description

In this problem, you are given a set of points on a 2D plane. We want to place a unit disc (i.e., a circle of radius 1) so that it covers as many points as possible. *Note that this is different from the lecture exercise where one of the point must be at the centre of the disc.*



What is the maximum number of points that we can cover with the disc at any one time? We will use the following simple (non-optimal) algorithm. First, some observations:

- A disc that covers the maximum number of points must pass through at least two points.
- For every pair of points that is of no more than distance 2 away from each other, there is at most two unit discs that have their perimeter passing through the two points.

This is a follow up to Lab #0 in CodeCrunch. You have to complete the tasks before embarking on this one.

The Task

Given a set of points as input, go through every pair of points, and for each circle that passes through them, count how many points are covered by each circle.

Input is in the following format:

- The first value is an integer, indicating the number of points `n` (`n > 2`).
- The next `n` pairs of values contains the `(x, y)` coordinates of the `n` points.

Take note of the following assumptions:

- The format of the input is always correct;
- There are always at least two points with a positive distance less than 2 between them;
- Output of a `double` value, say `d`, are to be formatted with `String.format("%.3f", d)`;
- Inconsistencies between sample output and actual output involving `-0.000` and `0.000` can be ignored.

This task is divided into five levels. The first four levels are the same as Lab #0 on CodeCrunch.

Only the last level is different. Specifically, you need to

- define a `Main` class with the `main` method to handle input and output;
- check for output format correctness using the `diff` utility (see specific level for usage details). Note that only **one** test case is provided for this;

You need to complete ALL levels to get full credit for this lab.

Level 1

Represent a Point

Design a class `Point` to represent a point object with each pair of `x`- and `y`- coordinates.

```
jshell> /open Point.java

jshell> new Point(0.0, 1.0)
$.. ==> point (0.000, 1.000)

jshell> /exit
```

Click here to submit to CodeCrunch.

Level 2

Find the mid-point and angle of line `pq`

Find the mid-point between two consecutive points `p` and `q` read from the input. At the same time, find the angle (in radians) o
`atan` or `atan2` Math function (refer to the Java API specifications).

```
jshell> /open Point.java

jshell> new Point(0.0, 0.0).midPoint(new Point(1.0, 1.0))
$.. ==> point (0.500, 0.500)

jshell> new Point(0.0, 0.0).angleTo(new Point(1.0, 1.0))
$.. ==> 0.7853981633974483

jshell> new Point(0, 0).angleTo(new Point(-1, -1))
$.. ==> -2.356194490192345

jshell> /exit
```

Click here to submit to CodeCrunch.

Level 3

Moving the point

A point can be moved at an angle θ and distance `d`

Hint: if a point, say`m`, is at `(x, y)`, then moving `m` at an angle θ and distance `d`, would result in the new position having the coo
`cosθ, y + d sinθ)`

```
jshell> /open Point.java

jshell> new Point(0, 0).moveTo(Math.PI / 2, 1.0)
$.. ==> point (0.000, 1.000)

jshell> /exit
```

Click here to submit to CodeCrunch.

Level 4

Creating the Circle

Together with the `Point` class in the preceding level, define the `Circle` class to allow us to create circle objects whose perime
two consecutive points.

By using the mid-point and angle of line `pq`, move the mid-point to the centre of the circle of radius `r` whose perimeter coincide
`q`. You will need to work out the respective angle and distance values.

```
jshell> /open Point.java

jshell> /open Circle.java

jshell> new Circle(new Point(0.0, 0.0), 1.0)
|  Error:
|  Circle(Point,double) has private access in Circle
|  new Circle(new Point(0.0, 0.0), 1.0)
|  ^---------------------------------^

jshell> Circle.getCircle(new Point(0.0, 0.0), 1.0)
$.. ==> circle of radius 1.0 centered at point (0.000, 0.000)

jshell> Circle.getCircle(new Point(0.0, 0.0), -1.0)
$.. ==> null

jshell> Circle.getCircle(new Point(0.0, 0.0), 0.0)
$.. ==> null

jshell> /open Main.java

jshell> Main.createCircle(new Point(0, 0), new Point(1, 0), 1)
$.. ==> circle of radius 1.0 centered at point (0.500, 0.866)

jshell> Main.createCircle(new Point(0, 0), new Point(1, 0), 2)
$.. ==> circle of radius 2.0 centered at point (0.500, 1.936)

jshell> Main.createCircle(new Point(0, 0), new Point(2, 0), 1)
$.. ==> circle of radius 1.0 centered at point (1.000, 0.000)

jshell> Main.createCircle(new Point(0, 0), new Point(0, 0), 2)
$.. ==> null

jshell> Main.createCircle(new Point(0, 0), new Point(3, 0), 1)
$.. ==> null

jshell> /exit
```

Click here to submit to CodeCrunch.

Level 5

Maximum Disc Coverage

You are now ready to find the maximum unit disc coverage. Just keep in mind that if the distance between points p and q is la
then there is no unit circle whose perimeter coincides with the points.

The following is a sample run of the program. User input is underlined.

```
2
0 0
1 0
Maximum Disc Coverage: 2
```

```
4
0 -1
1 0
0 1
-1 0
Maximum Disc Coverage: 4
```

Click here to submit to CodeCrunch.

Check the format correctness of the output by typing the following Unix command

```
$ java Main < test.in | diff - test.out
```