NAME: **login id:**

Precept:

COS 126 Midterm 2 Programming Exam Fall 2011

This part of the exam is like a mini-programming assignment. You will write code, compile it, download test data for it and run it on your laptop. Debug it as needed. This exam is open book, open browser—but only our course website and booksite! Of course, no internal or external communication is permitted (e.g., talking, email, IM, texting, cell phones) during the exam. You may use code from your assignments or code found on the COS126 website. When you are done, submit your program via the course website using the submit link for Precept Exam 2 on the Assignments page.

Grading. Your program will be graded on correctness, clarity (including comments), design, and efficiency. You will lose a substantial number of points if your program does not compile or if it crashes on typical inputs.

Even though you will electronically submit your code, you must turn in this paper so that we have a record that you took the exam and signed the Honor Code. *Print your name, login ID, and precept number on this page (now),* and write out and sign the Honor Code pledge before turning in this paper. Note: It is a violation of the Honor Code to discuss this midterm exam question with anyone until after everyone in the class has taken the exam. You have 50 minutes to complete the test.

"I pledge my honor that I have not violated the Honor Code during this examination."

_____

Signature

Part 1A	/10
Part 1B	/12
Part 2	/8
TOTAL	/30

Your task in this exam is to implement a (very) simple model for computing cellphone coverage. Given data describing a set of cellphone towers in a state that happens to be a perfect square, you will visualize (in Part 1) and compute (in Part 2) the percentage of the state that is covered.

The description below assumes that you have downloaded the `Picture` and generic `Queue` implementations from the booksite, as instructed. If not, you should do so now.

http://introcs.cs.princeton.edu/java/31datatype/Picture.java
http://introcs.cs.princeton.edu/java/43stack/Queue.java

Part 1A (10 points). First, write a simple class `Tower` that implements the following API:

```
public class  Tower
```

```
              Tower(double x, double y, double r)

      boolean  inRange(double x0, double y0)
       double  getX()
       double  getY()
       double  getR()
```

The constructor should simply save the argument values in instance variables.

The `inRange()` method for a `Tower` built with the values *x*, *y*, and *r* should return `true` if the specified point is within range of the tower (strictly within the circle of radius r centered at (*x*, *y*)), `false` otherwise. To compute the distance between (x_1, y_1) and (x_2, y_2), take the square root of the quantity $(x_2-x_1)^2 + (y_2-y_1)^2$, as usual.

The accessor methods `getX(),getY()` and `getR()` should return the *x*-coordinate of the tower, the *y*-coordinate of the tower, and the range (radius) of the tower, respectively.

Note. Do not forget to use the keyword "this" if your instance variables have the same name as the arguments passed to the constructor or method.

Note. Do not worry if you do not use your accessor methods in the client code for Parts 1B and 2.

Part 1B (12 points). Next, implement a `Tower` and `Queue` client `CoveragePicture` that visualizes the coverage for a set of towers on the unit square. Assume that the tower data is on **standard input**, three `double` values per line (*x*, *y*, and *r* in that order).

Your program must proceed as follows:

- Create an empty queue of towers and a blank 512-by-512 `Picture`. (*Note.* The default color for a blank picture is black.)
- Read the data from standard input and fill the queue with the towers specified by the data.
- Build the 512-by-512 `Picture` with pixel (*i*, *j*) set to `Color.WHITE` if the point (*i*/512, 1-*j*/512) is not within the range of any tower, `Color.BLACK` otherwise.
- Show the picture.

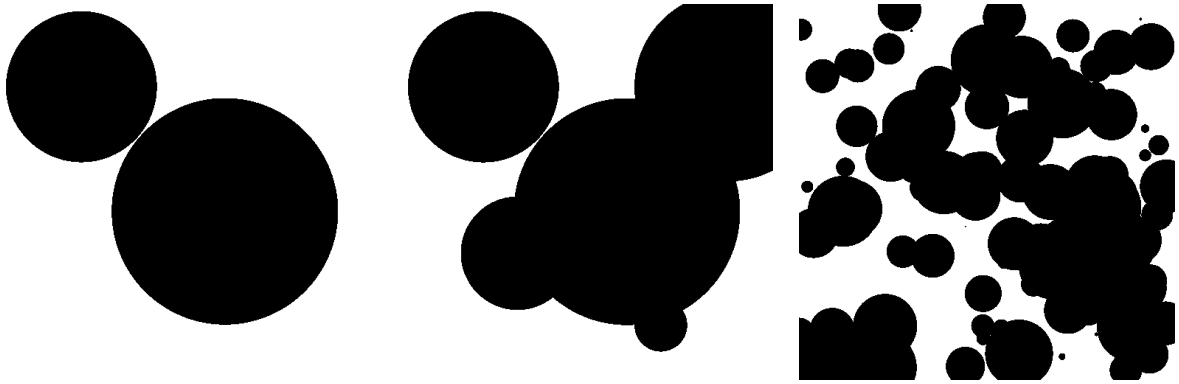
Don't forget to import `java.awt.Color` at the beginning of your program.

To test your program, download our test files

http://introcs.cs.princeton.edu/data/2towers.txt
http://introcs.cs.princeton.edu/data/5towers.txt
http://introcs.cs.princeton.edu/data/100towers.txt

and run them to produce the images below.

`2towers.txt` `5towers.txt` `100towers.txt`



Submission. Submit the two files `Tower.java` and `CoveragePicture.java` via Dropbox at

https://dropbox.cs.princeton.edu/COS126_F2011/Exam2

(This is the submit link for Precept Exam 2 on the Assignments page.) Be sure to click the *Check All Submitted Files* button to verify your submission.

Grading. Your program will be graded on correctness and clarity (including comments) and on your success in structuring the program as specified.

After you have submitted `Tower.java` and `CoveragePicture.java`, go on to Part 2.

Part 2 (8 points). Be sure that you have successfully completed both Part 1A and Part 1B and have submitted `Tower.java` and `CoveragePicture.java` before attempting this part of the exam (which may take longer and counts for less than half as many points).

Write a `Tower` client `Coverage` that takes a `double` value `T` from the command line and estimates the percentage coverage of the unit square by sampling each point in a `T`-by-`T` uniform grid of points, counting the number of points that are within range of some tower in the set of towers on standard input, and dividing by the total number of points.

For example, for `2towers.txt`, the circles are of radius .2 and .3 and do not overlap, so the percentage coverage is

$$\pi(.2)^2 + \pi(.3)^2 \approx 0.4084070449666731$$

and you are estimating that number. Output your estimate printed to one decimal place followed by "per cent coverage" as shown in the example below:

```
% java Coverage 1200 < 2towers.txt
40.8 per cent coverage
```

For `5towers.txt` and `100towers.txt`, the overlap makes the exact coverage difficult to compute, so sampling is a practical alternative:

```
% java Coverage 1200 < 5towers.txt
58.2 per cent coverage

% java Coverage 1200 < 100towers.txt
62.8 per cent coverage
```

Submission. Submit the single file `Coverage.java` via Dropbox at

https://dropbox.cs.princeton.edu/COS126_F2011/Exam2

(This is the submit link for Precept Exam 2 on the Assignments page.) Again, be sure to click the *Check All Submitted Files* button to verify your submission.

Grading. Your program will be graded on correctness and clarity (including comments).