Homework 1: Computing Pi by a Monte Carlo Method

Due: Jan. 18 at 11:55pm

This first homework assignment must be completed individually, not with a partner.

If you are ever stranded on a desert island and cannot remember the value of $\pi$, all you need to compute it is a random number generator. This programming assignment will show you how. The method used in this assignment is called a "Monte Carlo" method because it is based on probability.

Here are some useful facts:

- The area of a circle is equal to $\pi$ times the square of the radius.
- The area of a square is equal to the square of the length of the side.
- The distance from a point $(x_1, y_1)$ to a point $(x_2, y_2)$ is equal to the square root of $(x_2 - x_1)^2 + (y_2 - y_1)^2$.

Imagine a circle inscribed inside a square, as shown in the figure below. Since the radius of the circle is $R$, its area is $\pi R^2$. Since the side of the square is $2R$, its area is $4R^2$. Therefore the ratio of the area of the circle to the area of the square is $\pi/4$.

Using a random number generator, you can "throw" random points at the square. The ratio between the number of points that fall inside the circle and the total number of points that fall inside the square is an approximation to the value of $\pi/4$.

In this assignment, you will write a program that generates random "points" inside a square. To do this, use the Java method `Math.random()`, which returns a random number of type `double` between 0.0 and 1.0. You can get random numbers in a different range by multiplying the result of `Math.random()` by a scale factor and adding an offset. For example, if you want random numbers between -1.0 and +1.0, you can use the expression $-1.0 + 2.0 \times Math.random()$. Call your random number generator twice to get the x and y coordinates of a point inside your square. Measure the distance of your random point from the center of the square and decide whether the point is inside a circle inscribed within the square. Generate many points in this way, and compute the ratio of the number of points that are inside the circle to the total number of points. Multiplying this ratio by 4.0 gives an approximation for the value of $\pi$. 
**Hint:** You may choose any size and any placement in the (x,y) plane for your square. But your code will be simpler if you place the center of the square (and therefore the center of the circle) at the point (0, 0).

Your program should print out its approximation to the value of π after generating 10, 100, 1000, 10000, 100000, and 1000000 points. The output of the program should look as shown below (with real numbers in place of the n's.) Don't worry about the number of decimal places in the output—just let `println()` take care of this.

- After 10 tries, pi = n.nnnn
- After 100 tries, pi = n.nnnn
- After 1000 tries, pi = n.nnnn
- After 10000 tries, pi = n.nnnn
- After 100000 tries, pi = n.nnnn
- After 1000000 tries, pi = n.nnnn

Submit one file named `MonteCarlo.java` to Moodle before the assignment deadline. You do not need to submit the output of the program. The file you submit must represent your individual work.