Computer Science, Claremont McKenna College
CS51 - Introduction to Computer Science, Spring 2015

Problem Set 6 Due: 11:55 PM, Mar 12, 2015

General Instructions

Please carefully read and follow the directions exactly for each problem. Files and classes should be named exactly as directed in the problem (including capitalization!) as this will help with grading.

You should create your programs using your preferred text-editor, the Eclipse text editor, or jGrasp. Do not use a word processor such as Microsoft Word, WordPad, Google Docs, Apple's Pages, etc...

Your programs should be formatted in a way that's readable. In other words, indent appropriately, use informative names for variables, etc… Points will be deducted if your indenting style makes it difficult to follow your code. If you are uncertain about what is a readable style, look at the examples from class as a starting point for a reasonable coding style. .

Your programs should compile and run without errors. Please do not submit programs that do not compile! It's better to submit partial implementation that compiles as oppose to implementations that do not compile.

At the top of each file you submit, you should put your name and your email as comments.

This homework set should be submitted via Sakai using the Assignment Menu option (on the left pane). You should only submit the requested ".java" files. Do not submit ".class" files or Eclipse-specific project files. Finally, please do not submit files using Sakai's Dropbox Menu.

Turn in the following file(s):

```
PrintFactors.java
RepeatingStrings.java
RandomWalk.java
```

Problem 1

In a class called `PrintFactors`, write a static void method called `printFactorsAnd` that accepts a single integer parameter `N` and prints out the factors of `N` with the string "and" between the factors. For example, the call `printFactorsAnd(24)` should print the following exactly:

```
1 and 2 and 3 and 4 and 6 and 8 and 12 and 24
```

You may assume that the integer parameter n >= 1.

Supply a main method that calls your method with several test cases.

Problem 2

In a class called `RepeatingStrings`, write a static method called repeatString that takes two parameters: a `String str` and `an int n`. The method then returns a `String` that has the `str` parameter repeated `n` times inside square braces separated by commas. For example the following calls should produce the sample output:

```
System.out.println(repeatString("woohoo", 0));  ---> []
System.out.println(repeatString("woohoo", 1));  ---> [woohoo]
System.out.println(repeatString("woohoo", 2));  ---> [woohoo, woohoo]
System.out.println(repeatString("woohoo", 3));  ---> [woohoo, woohoo, woohoo]
```

Supply a main method that calls your method with the above test cases.

Problem 3

In a class called `RandomWalk`, write a graphical program that utilizes a loop to perform a random 2D walk simulation with the following requirements:

- The random walk should start at coordinates 150, 150
- Each "step" of the random walk is composed of an x and y component. Think of each step as a random delta x and delta y.
- Use a Random object to generate delta x and delta y in the range -10 to 10 (inclusive).
- The simulation should continue until the current position of the walk is outside or "escapes" a circle centered at 150, 150 with a radius of 100.
- When the simulation ends, the program should print the number of steps that it took for the random walk to "escape" the enclosing circle.

To visualize and test your program you should utilize a DrawingPanel to display the current location of the walk using a small (4x4) filled-in square of the color of your choice. You should also draw the outline of the circle that the random walk is trying to escape from. You only have to draw the current location and do not have to show the cumulative path of the walk.

Here are some suggestions:

- You may want to use the DrawingPanel's clear() or clearWithoutRepaint() methods to clear the window between calls to draw the walk's current location
- You should also sleep some number of msec (perhaps ~100msec) between calls to draw the current location of the walk so that the simulation can be followed visually.
- To simplify things, you may want to perform the simulation centered at the origin but draw it centered at 150, 150.