Exercises: Basic Java Syntax

If you can get through the first four, you are in good shape. Command line arguments are not all that important, so if you don't get to #5 and following, don't worry very much. Also, since we haven't covered objects yet, the simplest approach for now is, for each exercise, to make a new class and put all your code directly in "main".

1. Make a new project in Eclipse (File --> New --> Project --> Java --> Java Project, or, once you have done it once, just File --> New --> Java Project). But feel free to copy any code from the basic-syntax project, which is the source code from the lectures. Reminder: to copy code from one project to another, go to the first project, expand "default package", R-click on the Java class of interest, choose "Copy", go to the new project, R-click on the project, and choose "Paste".

- Note: I usually close projects that are not actively being edited. That way, I can't accidentally edit a file from the wrong project. To close an open project, right-click on the project name and choose "Close Project". To open a closed project, right-click on the project name and choose "Open Project".

2. Make a program that flips a coin 10 times, saying "heads" or "tails" each time. Recall that Math.random() returns a double between 0 and 1.

- Reminder: to run from Eclipse: R-click, Run As, Java Application

3. Create an array of 4 random numbers (each between 0 and 1). Use one-step array allocation. Loop down the array and print out the values.

4. Create an array of 100 random numbers. Use two-step array allocation. Print out the sum of the square roots of the values.

5. Make a program that simply prints out the number of command line arguments. ("You supplied *x* arguments.").

- If you run it the normal way in Eclipse (R-click, Run As, Java Application), it should say 0 for the number of arguments. You can also find the folder containing your code inside C:\eclipse-workspace and open a DOS window in that location. Then, you can do "java NumArgs foo bar baz" to supply "foo bar baz" as the command line arguments. You can also assign command line args within Eclipse by R-clicking, choosing Run As, selecting Run Configurations, and selecting Arguments tab. Be sure you choose the right program name here and in the next problem. The Eclipse way is more trouble than it is worth, in my opinion.

6. Make a program that prints the command line arguments in reverse order, converted to upper case. (Hint: the lecture didn't say how to turn Strings into upper case.)

7. Change the coin-flipping program of problem #2 to flip a coin the number of times the user specifies.You can supply a command line argument (convert a String to an int with Integer.parseInt(theString)) or supply a value on standard input (use Scanner and the nextInt method).