Homework: Final Program

CSS 161—Fundamentals of Computing

Summary

Build an interactive software application, a simulation, or a game.

Work Items

Submit your program as **a zip file of a directory** that contains all your Java source code files and your features-to-be-graded list:

1. Please submit only a single zip file via Canvas. Do not email your instructor your work.
2. Name your directory *LastFirst_HW_Final_Prog*. Name the zip file of that directory the same as that directory but with a *.zip* extension: *LastFirst_HW_Final_Prog.zip*. In both filenames, put in your own last name and first name for "Last" and "First". You **must name your directory and file this way.**
3. If you're not sure how to create a zip file of a directory, please do a Google search to find out. The process to do so will be different depending on what operating system you're using.
4. The name of your features-to-be-graded list (see Program Requirements below) should be named *LastFirst_HW_Final_Prog_Features.txt*.

Program Description

Write a program that demonstrates the skills we've learned throughout this quarter. This type of project offers only a few guidelines, allowing you to invest as much time and polish as you want, as long as you meet the program requirements described below. You can write a simple game (e.g., tic-tac-toe, Battleship), a simulation (e.g., a zero-dimensional energy balance model of the climate), or an application (e.g., a mortgage calculator). Your main requirement is that you demonstrate **six of the following features** in your software program and that you comment and document your code well:

1. Functional Decomposition: Use methods (a.k.a., functions) to break up a large program into meaningful chunks, using input to and output from those functions where appropriate.

2. Looping with Repetition Control Structures: Use two of the following structures: `for`, `while`, `do/while`, `foreach`.

3. Nested Loops: Use a loop within a loop in your program. Note that this is automatically accomplished when using Multi-Dimensional Arrays.

4. Branching with Selection Control Structures: Use multiple (i.e., more than one) `if/else` and/or `switch` statements in your code. (You don't need to use both kinds of statements. `if/else` means any kind of `if/else` statement.)

5. File I/O: Read from or write to a file in your software. Examples of this include be reading in a preset pattern for the computer opponent's answers in a game of rock/paper/scissors, or writing a file that logs each move the player makes, effectively recording a history of the game.

6. Using Multiple Classes: Build and use more than one class in your project. Note that a class that only has a `main` method and does not have any instance variables or methods does not count as a "class" in this feature. Thus, if you have a driver class that only has a main method and you want credit for this feature, you need at least two other classes in addition to the driver class.

7. One-dimensional arrays: Make use of a one-dimensional array in your software. If it is a partially-filled array, keep track of its current number of live elements with an `int`.

8. Class Design using Access Modifiers: Make all class-wide instance variables private in your class, and provide "getters" and "setters" to get and set the data accordingly.

9. Multi-Dimensional Arrays: Make use of an array with a dimensionality greater than one.

10. Recursion: Include a recursively designed function in your software, complete with (a) the recursive step and (b) the base step. (We did not cover this in the course, but if you know how to use it and want to, I'm willing to count it as a gradable feature for this assignment.)

11. Variable Tracing and Testing: Include a method(s) that can be used to provide output for tracing variables for the purpose of testing whether your code is working. The method(s) should be called `test`-something, e.g., `testStatistics`. Somewhere in your program, there should be a call to that method(s). In the code you submit, that call should be commented out, but I should be able to find it.

A few thoughts: Don't be intimidated by the requirements above. Reuse, reuse, reuse! Think about how you can you accomplish multiple goals simultaneously. For example, doesn't reading input from a file accomplish both File I/O and Looping with Repetition Control Structures? If you don't know where to start, take an example that looks interesting to you and start by adding to it.

NB: The requirement to comment and document your code well is in addition to the six features you've chosen. (Also, there are additional grading rubrics listed on the Canvas page for the assignment, but those form a relatively small part of the assignment grade.)

Finally, submit a file with your source code (put it in the same directory as your source code) where you list which six of the above features you want to be graded on. **Make sure you number**

these features from one to six; the order doesn't matter, but I'll use the numbering as references while grading. Make this file a plain text file and with an extension *.txt*.

About this Document

Course taught at the University of Washington Bothell. By Rob Nash, Summer 2013, with edits by Johnny Lin, Autumn 2014.