CMSI 186
Programming Laboratory
Spring 2008

Midsemester Tips

Having now seen two sets of programs and lab reports from you, I wanted to compile (no pun intended) some points that will help improve the quality of your work. As overall themes, consider the following quotes from some computer science notables:

Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.

— Donald Knuth

Programming can be fun, so can cryptography; however they should not be combined.

— Charles Kreitzberg and Ben Shneiderman

Clarity is better than cleverness.

— Part of the Unix philosophy, as phrased by Eric S. Raymond

With these pithy statements in mind, let's get to some specifics; they're not as cleverly worded as above, but I hope that they find their way into your future work.

General

- Follow the specification. If you were asked to build a Lego brick of a certain size and shape, you wouldn't want to change that in any way, because that brick is meant to go somewhere, and may no longer fit if you change it. The specifications in this course are not arbitrary rules to make your life more difficult — they provide practice opportunities to prepare you for when they'll significantly, and perhaps critically, affect an overall project or functionality.
- Note that, as you gain experience, you'll find ways to follow the specification while simultaneously extending it. That is, your program will still "fit" the overall design, but may have new elements that go beyond what the original design could have done. But until you *do* know how to do this…follow the specification.

Lab Report

- Maintain section and subsection titles in your lab reports — all formal documents do this.
- Similarly, there is no need to retain the descriptions provided in the outline (i.e., "This section says this and that about them and those.").
- Stick to plain text (i.e., follow the specification).
- Maintain the proper tone in your report text. Remember, this is a formal document that constitutes your "record" of the assignment.

Code

- Indent properly (and use spaces instead of tabs for consistency).
- Space out your code consistently — if you like spaces before and after parentheses, then do that across the board; that said, there *is* a happy medium between using too many and too few spaces. If you're curious about the rules that I use in the code that I write, just ask.
- Clean up after yourself — throw away unused variables, check spelling, etc. This shows that you cared enough to review your work even after you've gotten it to function correctly.
- Watch out for duplicate code — that is, code that does almost the same thing but appears more than once. Unify this code if you know how, and if you don't, ask me.
- Write your tests first — first, they help clarify, *to yourself and before you actually write anything else*, what is expected of the code; second, they then communicate these expectations explicitly to others. As mentioned before, it is very satisfying to see your tests start as failures, then gradually emerge as successes during implementation.
- Of course, you can't write your tests *absolutely* first. You also need *stubs* — empty versions of the methods that you will eventually fill out. Very easy (and quick), and they typically flow out of the specification. So, stubs then tests.

- The key mentality with tests is finding the *boundary cases* — that is, the scenarios that may hit rarely-called sections of your code, or the ones that will make your code perform more complicated or error-prone work. Put another way, think of your tests as challenges to how good or robust your code really is.

Tools

- Don't let your tools force you to diverge from the specification. *Make* them work with it — and if you can't, question your choice of tool (or your qualification to use it).
- It's always useful to be proficient at more than one way of doing something. Thus, if you find that you're "stuck" on one tool and are helpless without it, take that as a sign that you should pick up an alternative way of doing your work.

Java Specifics

- Learn and follow Java conventions — understandably, you don't necessarily have a good handle on them yet, but I've been pointing them out here and there; once mentioned, try not to have them mentioned again.
- Master those packages — if you're still not clear on them, let me know, and we'll try to settle this once and for all