Welcome to CS61B!

- In (or preferably before) lab this week, *get* a CS61B Unix account from `https://inst.eecs.berkeley.edu/webacct`.
- If you plan to work from home, try logging in remotely to one of the instructional servers.
- We'll be using Piazza for notices, on-line discussions, questions.
- General information about the course will appear (eventually) on the home page (grading, lateness, cheating policy, etc.).
- Lectures will be be screencast.
- Space is *very* tight this semester. I cannot admit wait-listed or concurrent enrollment students until there is room. So, if you decide the course is not for you, please drop it so others can get in.!
- If you are wait-listed on a lab section and can take an alternative lab instead, you can enroll by removing yourself from the wait list and then re-adding. Welcome to new-system-itis.
- You need not sign up for discussion sections. We suggest attending the one associated with your (e.g., 102B and 102C).

Texts

- There are two readers currently on-line (see the website).
- You could do without printed versions, *except* that we don't allow computers in tests (but do allow printed stuff).
- There will be paper copies at Vick Copy (*not* Copy Central), corner of Hearst and Euclid, before the first test and after I get a good idea of how many are needed.
- Textbook (for first part of the course only) is *Head First Java*. It's kind of silly, but has the necessary material.

Course Organization I

- You read; we illustrate.
- Labs are important: exercise of programming principles as well as practical dirty details go there. Generally we will give you homework points for doing them.
- Homework is important, but really not graded: use it as you see fit and *turn it in!* You get points for just putting some reasonable effort into it.
- Individual projects are *really* important! Expect to learn a lot. Projects are *not* team efforts (that's for later courses).

Course Organization II

- Use of tools *is* part of the course. Programming takes place in a *programming environment:*
  - Handles editing, debugging, compilation, archiving versions.
  - Personally, I keep it simple: Emacs + gjdb + make + git, (documented in one of the readers and on-line). But we'll also look at IntelliJ in lab, and Eclipse is OK, too.
- Tests are challenging: better to stay on top than to cram.
- Tests, 45%; Projects, 45%; HW, 10%
- Stressed? Tell us!

Programming, not Java

- Here, we learn *programming,* not Java (or Unix, or Windows, or...)
- Programming principles span many languages
  - Look for connections.
  - Syntax (`x+y` *vs.* `(+ x y)`) is superficial.
  - E.g., Java, Python, and Scheme have a lot in common.
- Whether you use GUIs, text interfaces, or embedded systems, important ideas are the same.

For next time

- Please read Chapter 1 of *Head First Java*, plus §1.1–1.9 of the on-line book *A Java Reference*, available on the class website.
- This is an overview of most of Java's features.
- We'll start looking at examples on Friday.
- Always remember the questions that come up when you read something we assign:
  - Who knows? We might have made a mistake.
  - Feel free to ask at the start of lectures, by email, or by Piazza.

Acronyms of Wisdom

DBC

RTFM

The First Program

```
public class Hello {

    public static void main(String... args) {
        System.out.println("Hello, world!");
    }
}
```

Advertisement

- The Berkeley Programming Contest is approaching (late September).
- We use it as a qualifying trial for the ACM regional contest in November.
- So, if you know any real hotshots (or are one yourself) tell them about this opportunity to show that they have what it takes.