

UNIVERSITY OF CALIFORNIA
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

CS61B
Fall 2001

P. N. Hilfinger

CS 61B: Introduction to Programming, Part II
General Course Information*

Introduction

Welcome to CS 61B. The CS 61 series is an introduction to computer science, with particular emphasis on software and on machines from a programmer's point of view. CS 61A covered high-level approaches to problem-solving, providing you with a variety of ways to organize solutions to programming problems: as compositions of functions, collections of objects, or sets of rules. In CS 61B, we move to a somewhat more detailed (and to some extent, more basic) level of programming. As in 61A, the *correctness* of a program is important. In CS 61B, we're concerned also with *engineering*. An engineer, it is said, is someone who can do for a dime what any fool can do for a dollar. Much of 61B will be concerned with the tradeoffs in time and memory for a variety of methods for structuring data. We'll also be concerned with the engineering knowledge and skills needed to build and maintain moderately large programs.

Instructor: Paul N. Hilfinger, 787 Soda Hall, 642-8401, hilfinger@cs.berkeley.edu

Discussion and Lab Sections

The teaching assistants this semester are Steven Sinha, Kaushik Datta, Ali Lakhia, Adam Kirk, Jean-François Boudier, and Michael Hamler. We will maintain contact information about them on the [class web page](#). There will also be readers grading things other than tests, and volunteer lab assistants staffing the lab sections (at least, we sure hope so).

I really don't care what discussion and lab section you choose to be in, but please make sure you do have them and that their TAs know about you. If you want to change sections, just clear it with the TAs involved. Please don't even think about informing TeleBEARS (or me) if you change sections; it will only cause trouble, and it makes no difference for my records.

*With contributions by Brian Harvey, Mike Clancy, Katherine Yelick, and John Canny.

On-line Resources

The course home page will provide one-stop shopping for course information. All the handouts, homeworks, labs, staff contact information, etc., will be posted there. The home page is <http://www-inst.eecs.berkeley.edu/~cs61b>. If you're logging in over a terminal that can't run Netscape or Explorer, you can still access the page text-only using lynx. Type `lynx` followed by the URL above.

Please *do not* print out on-line information that we hand out in class or that is available through the indicated copy stores. It is a waste of paper and ties up the lab printers. I will try to keep extra paper handouts available if you need them.

The course newsgroup is `ucb.class.cs61b`. For most questions about the course, the newsgroup is the right place to ask them. The course staff read it regularly, so you will get a quick answer. That way, other students benefit by seeing the question and the answer. Don't forget to check the newsgroup before asking your question, just in case someone else has posted it.

The e-mail address `cs61b@cs` will send a message to the entire staff. You can use it for correspondences that you don't want to send to the newsgroup. Again, all the staff read it, so you will usually get a quick reply. If you send a question that is of general interest, we may post the response on the newsgroup (we will keep personal information out of it, of course). To talk with us, the best way is to come during regular office hours (posted on our doors as well as in the home page). Many of us are available at other times by appointment. Please don't be shy; web pages, e-mail, and news are useful, but it's still much easier to understand something when you can talk about it face-to-face.

When logged into our instructional systems for CS61B work, please make sure that you are using the standard configuration for the class—that is, the files `.cshrc`, `.login`, `.emacs`, etc., that should have been in your accounts initially. If you must modify these, we suggest that you continue to have them read our scripts from the `~cs61b/adm` directory, so that we can easily propagate corrections to you. In any case, if you modify these files, you are on your own.

One of the advantages of using Java in the course is that Java programs are highly portable (at least the kind we'll be writing). If you have an installation of Symantek's Cafe, Microsoft's J++, IBM's Visual Age Java, or Sun's JDK (Java Development Kit) at home on your PC or Mac, feel free to use it to develop solutions to programming problems. We will be using version 1.3 (aka 2) of the language. The modified version of Sun's debugger that we use—called `gjdb`—is only likely to work on JDK 1.3; in particular, don't expect it to work at all on a non-Sun Java implementation. You will need to download some `.class` files from us. Details will appear on the class home page. The JDK (which we use in class) is free, and may be downloaded from Sun if you have the time.

Background Knowledge

Some of you may have thought that the stuff you learned in CS 61A was mere esoteric fluff inexplicably thrown at you to weed out the faint of heart. Not true. In fact, although the

syntaxes of Java and Scheme are enormously different, the underlying computational models are surprisingly similar. You will find that almost all the “big ideas” you see in Java had their analogues in what you learned in CS 61A (indeed, one self-test of your understanding of the course material in CS 61B is to check that you see all the similarities). This course will assume you are familiar with CS 61A, and there will be some references to the 61A textbook (Abelson, Sussman, and Sussman). If you haven’t taken 61A, you may be confused sometimes, and you should make sure you review a copy of Abelson, Sussman, and Sussman early on in the semester. We will also make available on-line documentation of the object-oriented extension to Scheme that was used in CS 61A.

All the instructional machines for this course will be running various flavors of the Unix operating system, and it’s essential that you become familiar with it. The introduction *User’s Guide to Unix and EECS Instructional Facilities* is available on-line from the class home page. Another good introduction is “A Practical Guide to The Unix System” by Mark Sobell (Benjamin Cummings pub. 1995) available at bookstores. Over the course of the semester, there are help sessions on various useful computer-related topics; we will put appropriate links on the course home page.

Is this the right course?

This is a course about data structures and programming methods. It happens to also teach Java and a bit of C, since it is hard to teach programming without a language. However, it is not intended as an exhaustive course on Java, the World-Wide Web, creating applets, user interfaces, graphics, or any of that fun stuff. Some of you may have already had a data structures course, and simply want to learn Java or C++. For you, a much better choice would be self-study, or (for C++) CS 9F “C++ for programmers,” a one-unit self-paced course that will teach you more of what you want to know in less time. There is no enrollment limit for that course, and you work through it at your own pace after the first and only lecture. There is a similar course for Java (CS 9G).

Wait-listed?

Enrollments are high this semester. We will try to enroll all who meet the prerequisites, but at the moment, I can’t say how successful we will be. If you do not meet the prerequisites, you will be (or at least, are supposed to be) dropped from the course. There is an appeals procedure for such cases; appeal forms will be available in CS office. Michael-David Sasson, msasson@cs.berkeley.edu, in 379 Soda and Mike Clancy, clancy@cs.berkeley.edu handle appeals. Please do *not* try appealing to us; we have nothing to do with the process!

While we determine who will fit, please go to any discussion or lab in which you can fit (that is, during which you are otherwise free and in which the TA can find room for you). Again: if you are wait-listed but meet the prerequisites (CS61A), then go to lab sections, pick up account forms, do homework, and attend lectures as if you were in the course.

Course Materials

The textbooks for this course are *On to Java 2* (third edition), by Winston and Narasimhan, and *The C Programming Language* (second edition) by Kernighan and Ritchie. These are available through the campus bookstore.

There will also be three readers available through Vick Copy, 1879 Euclid (corner of Euclid and Hearst). The first two readers should be available the first day of class, and the third will be available in a few weeks. The readers contain reference material on the Java programming language, documentation to assist in lab exercises and projects, and material on data structures. We will keep electronic copies of the readers and of all other handouts available through links from the course home page.

Enrollment: Laboratory and Discussion Sections

There will be one lab section and one discussion section per week. We have scheduled lab sections between the Monday and Wednesday lectures, and discussion sections between the Wednesday and Friday lectures. You should expect that new material may be presented during the scheduled labs. Actually, you can do the projects and lab exercises any time, but we can only guarantee organized assistance during the scheduled lab sections in Soda. If we have enough volunteers, we will cover some other times with lab assistants. Check the home page for updates on this. You should plan on attending the discussion sections. Tests will be returned in section.

Computer Accounts

The CS 61B scheduled labs will all be held in 275 Soda, which contains Intel workstations running Solaris. At other times, you can use any computer that is not being used for another scheduled lab. Lab times are normally posted outside each computer room. You will receive a computer account form in lab the first week. Extra forms will be available in 385 Soda after that. Information on computer facilities and computing from home should be available via the class Web page.

You must electronically register the account you intend to use for handing in assignments (only one account, please) during the first week (by Friday's lecture). If you use one of the class accounts we hand out (logins starting with `cs61b`), you will be asked the standard registration questions when you log in. From other (named) accounts, use the command

```
~cs61b/bin/register
```

and follow any instructions it gives you.

Homework and Programming Assignments

There will be ordinary homework assignments during the term, some of which include small programming problems, and three larger-scale programming projects. Everything you turn

in must show your name, your computer account, your discussion section number, and your lab section number. You will turn in everything electronically. Be sure you have an account (registered for CS61B) for that purpose.

Testing and Grading

In addition to homework, there will be two tests during the term, and a final. All tests are open book, open notes, and may cover any material whatever (however, to prevent out-and-out mutiny, I am generally reasonable in the material selected).

The programming projects will count for a total of 90 points, and written homeworks and labs for 20 points. Each of the two tests will count for 20 points; and the final will be worth 50 points. Your letter grade will be determined by total points out of the possible 200:

185	170	160	150	140	130	120	110	100	90	80	75	< 75
A+	A	A-	B+	B	B-	C+	C	C-	D+	D	D-	F

In other words, *there is no curve*; your grade will depend only on how well you do, and not on how well everyone else does. For your information, University guidelines suggest that the average GPA for a lower-division required course be in the range 2.5–2.9, with 2.7 (B-) being typical. This corresponds to getting 50% on tests (typical for my courses), 75% on projects, and 100% on homeworks (on which you get credit simply for turning something in).

If you believe we have misgraded an exam, return it to your TA with a note explaining your complaint. We will regrade the entire test. You should check the on-line solutions first to make sure that this regrade will make your total score go up.

I will grant grades of Incomplete *only* for dire medical or personal emergencies that cause you to miss the final, and only if your work up to that point has been satisfactory. Do *not* try to get an incomplete simply as a way to have more time to study or do a project; that is contrary to University policy.

Inevitably, some of you will have conflicts with the scheduled exam times. I will arrange for alternative test times for those people who have sufficient cause. Sufficient causes include exams scheduled at an overlapping time in another course, medical or family emergencies, or certain major religious holidays (however, I believe I have avoided all of those). Popular reasons that are *not* sufficient cause include having job interviews, having exams or assignments in other courses at nearby times, being behind in your reading, being tired, or being hungover. With the obvious exception of emergencies, you must arrange alternative exam times with me well in advance.

Policy on Collaboration and Cheating

I strongly encourage you to help each other on homework assignments. Ordinary homework is not seriously graded: you get points for handing in something. Naturally, though, it is in your best interest *not* to take advantage of this fact, and to treat homework seriously.

The three projects are *individual* efforts in this class (no partnerships). Feel free to discuss projects or pieces of them *before* doing the work. But you must complete and write up each project yourself. That is, feel free to discuss projects with each other, but be aware that we expect your work to be *substantially* different from that of all your classmates (in this or any other semester).

Copying and presenting another person's project or test work as your own constitutes cheating. Electronically submitted programs are particularly easy to check for copying or trivial changes, and we will be doing that. Any incident of cheating will be reported to the my departmental chairman and to the Office of Student Conduct. I realize that CS and EE programs at Berkeley are very intense, and that students are often under extraordinary pressure to make deadlines. But deadlines are a fact of life, and will persist after college. The trick is to get ahead of them. You can seek advice from the staff early if you feel yourself getting behind in something. Knowing where and how to get advice on things you don't understand is a skill everyone needs to succeed in the real world.

Lateness

We will give no credit for written homework turned in after the deadline. Please do not beg and plead for exceptions; an individual assignment is worth too few points to justify your groveling at the my feet (a comment that probably applies to individual test questions, as well). You can miss an assignment or two and still get your A+.

On programming projects, we are a tad more lenient. I'll penalize an assignment $5N/12$ percent for each N hours it is late, rounded off in some unspecified fashion. During the course of the semester, however, we'll give you a *total* of three late days (72 hours) for free.

Lecture Recording

The lectures will be recorded and available for replay over the Web (at least, that's the story as of the first week of class). Details and links will appear on the class home page when we get them.