General Course Information | CS61B Fall 2017

Author: P. N. Hilfinger

A. Introduction

Welcome to CS 61B. The CS 61 series is an introduction to computer science, with particular emphasis on software and 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.

B. Background Knowledge

This class assumes you have taken CS61A or E7, or have equivalent background to a student who has taken one of these courses. The course is largely built upon the assumption that you have taken CS61A and E7 students may find the beginning of the course to be a bit scarier, particularly when it comes to object-oriented programming.

We assume you are coming in with zero Java experience. Nonetheless, we will move through basic Java syntax quickly. 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, Python, 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).

If you already have Java experience, great! We hope that you'll help out your fellow students in discussion, lab, and on Piazza, particularly in the opening weeks when everyone is catching up on Java.

It would be nice, though not absolutely necessary, to have some UNIX experience before the semester starts. All the instructional machines for this course will be running various flavors of the UNIX operating system (generally, Ubuntu) and it's a really good idea for you to become familiar with it. There will be online resources for learning Unix linked from the first lab. We will also announce on-campus help sessions on the announcements section of this website. If you'd like a physical reference, you might tryLinux Command Line and Shell Scripting Bible by Richard Blum and Christine Breshnahan, which is fairly recent and positively reviewed. You'll find several other books on the topic on Amazon.

C. Is This the Right Course for Me?

This is a course about data structures and programming methods. It happens to also teach Java, 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, self-study may be a better option. CS 9F (C++ for programmers) and CS 9G (Java for programmers) are both one-unit self-paced courses 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.

Finally, the 1-unit self-paced course CS 47B is for students "with sufficient partial credit in 61B," allowing them (with instructor's permission) to complete the 61B course requirement without taking the full course.

D. Discussion and Lab Sections

Each week there is a 1 hour discussion section and a 2 hour lab section headed by a GSI and supported by volunteer lab assistants. You can find information about the staff running each section on the staff page.

If you decide to permanently switch sections, please clear it with the two TAs involved.

Attendance is optional for discussions and labs. However, your overall level of effort and participation may be taken into account if you're on a grading boundary, and getting to know your discussion TA is one way to demonstrate effort and participation. Also, labs are a time during which you have access to TAs and lab assistants to answer questions and help you over various conceptual hurdles. In general, lab submissions are due at midnight on Friday, although it would be best to complete them during (or before) the lab.

Labs will be online, so you will be able to do much of the work ahead of time. Nonetheless, we encourage you to attend your scheduled lab time. One major purpose of the labs is to give your TA a chance to check up on you and to find out what people are and are not understanding. We've found that with the increasing ability to work anywhere has come an increasing tendency for students to go off by themselves and fall behind. Don't make this mistake. Keep up with homework and lab work and above all *let us know when you don't understand something!*

The course home page will provide one-stop shopping for course information. The course schedule as well as all handouts, homework, labs, FAQs, etc., will be posted there.

Our discussion forum this semester will be Piazza. For most questions about the course, Piazza is the right place to ask them. The course staff read it regularly, so you will get a quick answer. Furthermore, by posting online as opposed to emailing us directly, other students benefit by seeing the question and the answer. Don't forget to check Piazza before asking your question, just in case someone else has already posted it.

The e-mail address cs61b@inst.eecs.berkeley.edu will send a message to the course staff (PNH and the TAs). You can use it for correspondence that you don't want to send to Piazza. We all 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 Piazza (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 the home page). The second best option is to stop by an idle lab. Many of us are available at other times by appointment. Please don't be shy. Web pages, email, and Piazza are useful, but it's still much easier to understand something when you can talk about it face-to-face.

E. Course Materials

The optional textbook for the first seven weeks of this course is *Head First Java, 2nd Edition by Sierra*and Bates (O'Reilly, 2005). This book is recommended to those of you with no Java experience. It's an easy read and super cheap for a textbook. There are also some on-line notes about Java from which we'll assign some readings.

For the second part of the course, we'll use another set of on-line notes about data structures. There are numerous other sources you may find useful, notably Algorithms, 4th Edition.

Both textbooks for this course are optional. Homework will not be assigned from them, and alternate readings will be provided when possible.

Head First Java is not a reference book (it says so on page xxii). The official description of the Java core language is available online in [The Java Language Specification] by James Gosling, Bill Joy, Guy Steele, Gilad Bracha, and Alex Buckley. It's extremely thorough and easy to read (once you understand how to read it).

F. Software

This course does not have an official text editor. You may use Vim, Emacs, Sublime, or IDEs like Eclipse, Netbeans, IntelliJ, Sublime, etc. Whatever you use, however, your submitted solutions must conform to our expected layouts, as indicated in the assignments. We recommend that you use IntelliJ starting as soon as you finish project 0. We will not officially support any IDE other than IntelliJ.

This semester, we will be using Java 8. It is already installed on the lab computers, and it may be provided with your computer as well.

You will be able to do any work you'd like on any Windows, Mac OS X, or Linux computer. You may also remotely log into the instructional machines (which you will receive an account for during the first week), though you should be able to most everything in the course by working natively on your own computer. Information for setting up your own computer is linked in Lab 1.

We'll be using the version-control system Git for keeping and tracking your work. Version-control systems allow you maintain a series of "snapshots" of your files at various points in their development. Used properly, this provides you some back-up protection, so that you can recover previous states of your work when something goes wrong. Also for team-oriented projects (as well as in the real world), version-control systems help manage collaborative work.

You will be learning and using Git for this course to track your work. This will allow the staff to track your progress in the course. The first lab will teach you the basics of what you will need to know. Feel free to also read Git documentation.

G. Homeworks and Labs

There will be weekly labs and about half as many homeworks. Labs will take approximately two hours to complete, though some may run slightly longer. HWs will vary from 3 to 10 hours of work. You will turn in everything electronically using Git. All homeworks and labs are individual efforts (without partners).

Homework will be given full credit for passing a substantial proportion of a suite of correctness tests while labs will receive full credit for showing "reasonable effort." Passing all of our autograder tests for homework will ensure full credit.

No extensions or grace hours will be granted for labs or homework except in cases of emergency (with require instructor approval). To allow for situations that may arise in the course of life, the lowest one homework assignment will be dropped and the two lowest labs will be dropped.

H. Programming Projects

There will be three larger programming projects (and a warm-up "miniproject"). The projects will be individual efforts, with the same collaboration rules as HW. You'll have later classes that will provide more opportunities to build your teamwork skills. For 61B, our goal is to help you build the independence and fundamental skills that you'll need to thrive when building large programs.

Projects will have some other tests that we will not report to you. Thus, passing all `make check` tests will not necessarily guarantee receiving full points on the autograded portion of the project. This is intended to encourage you to write your own tests to provide a more complete test suite.

Miniproject 0 and projects 1, 2, and 3 will be worth 16, 28, 28, and 28 points respectively. There may be extra credit opportunities on some projects.

Testing Environment

For all submitted assignments (projects, homeworks, labs), we run the autograder ***on the instructional machines.*** The fact that your assignment runs at home is **not** sufficient; it must run on our setup. This only mirrors the real world: customers run software on the systems they have, not on the specially augmented systems that developers might use to create that software. You can check your submission by logging into an instructional server, and checking out and testing your submission on your local repository there, using a command sequence such as

```
  git fetch origin
  git checkout proj1-2
  cd proj1
  make
```

```
make style
make check
git checkout master    # To get back to the master branch.
```

Also, you can check all results using the Scores tab on our web pages.

I. Exams

There will be two evening tests during the last weeks of September and October, and a final exam on 12/11/2017. All tests are open book, open notes. Officially, exams may cover any material whatsoever. On the other hand, we are known to be reasonable.

Inevitably, some of you will have conflicts with the scheduled exam times. We will arrange for alternative test times for those people who have sufficient cause. Sufficient cause includes conflicting exams for other courses, medical or family emergencies, or important religious holidays.

We will provide a form several weeks before each exam for students to request alternative exams, which will almost certainly occur after the regular exam has occurred. With the obvious exception of emergencies, you must fill out this form by the provided deadline to be considered for an alternate exam.

Popular reasons that are *not* sufficient cause include having job interviews, having a plane ticket that you (or your parents) bought without consulting the schedule, having exams or assignments in other courses at nearby times, being behind in your reading, being tired, or being hung over.

J. Grades

Your letter grade will be determined by the total points out of the possible 200. In other words, *there is no curve.* Your grade will depend solely on how well you do, and not on how well everyone else does.

Category	Percentage	Points
Homework/Labs	~10%	20
Projects	~50%	100
Tests	~17%	34
Final Exam	~23%	46
Total	~100%	200

Letter grade cutoffs:

A+	A	A-	B+	B	B-	C+	C	C-	D+	D	D-	F
182	162	152	142	132	122	112	100	92	83	73	66	0

We 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.

K. Policy on Collaboration and Cheating

Deadlines can be stressful, and we know that under extreme pressure, it becomes tempting to start rationalizing actions that you would otherwise yourself consider inappropriate. Perhaps you'll find yourself facing a 61B project deadline, and under all this stress you'll convince yourself that you're just going to cheat for the moment so you can get the points, and that you'll come back later and really learn the thing you were supposed to have learned in order to restore your karmic balance (we've heard something along these lines a few times).

This is a terrible idea. Obviously it's important to learn how to deal with deadlines, but far more important than that, giving into this sort of pressure under not-so-dire circumstances is going to do some damage to your moral compass. Someday, when the consequences are higher than potentially losing a 1/3rd of a letter grade, you may find yourself committing dishonest acts at the cost of someone else's livelihood or life.

L. Homework and Lab Collaboration Policy

In CS61B, we have three types of assignments: homeworks, labs, and projects. The entire point of homeworks and labs is to learn. For homeworks or labs, you should feel free to collaborate with others however you choose, though keep in mind that greater independence is likely to give you a better learning experience (as long as you aren't totally stuck). **Even though we will allow close collaborations on HWs and labs, your code should still be your own work!** Identical or near identical submissions will be treated as plagiarism.

M. Project Collaboration Policy

By contrast, the projects were designed not just for learning (particularly how to be self-reliant in the context of large unfamiliar systems), but also for the dual purpose of evaluating of your mastery of the course material. As such, they are intended to be completed primarily on your own, particularly when it comes to writing the actual code.

For projects and exams, we will be absolutely unforgiving. Any incident will result in a failing grade for the course, though Berkeley will let you retake

61B next semester. All incidents of cheating will be referred to the Office of Student Conduct.

What constitutes cheating? **The golden rule of academic dishonesty is that you should not claim to be responsible for work that is not yours.**

This is obviously open to some interpretation, and you'll be getting some help from instructors, the internet, other students, and more throughout the course. This is OK, and we hope that the class is an open, welcoming, collaborative environment where we can help each other build the highest possible understanding of the course material.

To help (but not entirely define) the bounds of acceptable behavior, we have three important rules for projects:

- **By You Alone**: All project code that you submit (other than skeleton code) should be written by you alone, except for small snippets that solve tiny subproblems (examples in the Permitted section below).
- **Do Not Possess or Share Code**: Before a project deadline, you should never be in possession of solution code that you (or your partner) did not write. You will be equally culpable if you distribute such code to other students or future students of 61B (within reason). DO NOT GIVE ANYONE YOUR CODE -- EVEN IF THEY ARE DESPERATELY ASKING. DO NOT POST SOLUTIONS TO PROJECTS ONLINE (on GitHub or anywhere else)! By the way, any public posting of code derived from our skeletons is a violation of copyright as well. If you're not sure what you're doing is OK, please ask.
- **Cite Your Sources**: When you receive significant assistance on a project from someone else, you should cite that assistance somewhere in your source code. We leave it to you to decide what constitutes 'significant'.

For clarity, examples of specific activities are listed below:

Permitted:

- Discussion of approaches for solving a problem.
- Giving away or receiving significant conceptual ideas towards a problem solution. Such help should be cited as comments in your code. For the sake of other's learning experience, we ask that you try not to give away anything juicy, and instead try to lead people to such solutions.
- Discussion of specific syntax issues and bugs in your code.
- Using small snippets of code that you find online for solving tiny problems (e.g. googling "uppercase string java" may lead you to some sample code that you copy and paste into your solution). Such usages should be cited as comments in your hw, lab, and especially project code!

Permitted with **Extreme Caution**:

- Looking at someone else's project code to assist with debugging. Typing or dictating code into someone else's computer is a violation of the "By You Alone" rule.
- Looking at someone else's project code to understand a particular idea or part of a project. This is strongly discouraged due to the danger of plagiarism, but not absolutely forbidden. We are very serious about the "By You Alone" rule!

Absolutely Forbidden:

- Possessing another student's project code in any form before a final deadline, be it electronic or on paper. This includes the situation where you're trying to help someone debug. Distributing such code is equally forbidden.
- Possessing project solution code that you did not write yourself (from online (e.g. GitHub), staff solution code found somewhere on a server it should not have been, etc.) before a final deadline. Distributing such code is equally forbidden.
- Posting solution code to any assignment in a public place (e.g. a public git repository, mediafire, etched into stones above the Mediterranean, etc). This applies even after the semester is over.

We have cheating-detection software, and we will routinely run this code to detect cheating. Every semester, we catch and penalize a significant number of people. Do not be one of them. If you find yourself at such a point of total desperation that cheating begins to look attractive, contact one of the instructors and we can maybe help somehow. Likewise, if 61B is causing massive disruption to your personal life, please contact us directly.

If you admit guilt to an act of plagiarism before we catch you, you will be given zero points, and we will not refer your case to the university administration.

Obviously, the expressive power of Java is a subset of the English language. And yes, you can obviously obey the letter of this entire policy while completely violating its spirit. However, this policy is not a game to be defeated, and such circumventions will be seen as plagiarism.

N. Lateness

We will give no credit for homeworks or labs after the deadline. Your lowest homework score and two lowest lab scores will be dropped. This is intended to handle any contingencies. With hundreds of students, we cannot afford to handle individual cases, though if you have an extreme situation that warrants our attention, please contact Professor Hilfinger via email.

On programming projects, we are a tad more lenient. I'll penalize an assignment 5/12 percent for each hour it is late, rounded off in some unspecified fashion. This means that you will lose roughly 10% of the points for a project each day (though tracking is by hours late, not days).

I'll give you some grace hours for free. They work like this: on the first project, you get 24 hours of additional time past the deadline before we start counting your project as late. On the second project, you also get up to 24 hours plus any leftover late time you had from the first project. Likewise, on the third project you get 24 hours, plus any remaining late time you didn't use for the second project. No extra slip hours for the fourth project.

O. Acknowledgments

Some course handout material derived from Josh Hug's CS61B handout.