CS 61B Data Structures, Spring 2016

Instructor: Josh Hug
Lecture: MWF 3-4 PM, Wheeler Auditorium

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.

In 61A, the *correctness* of a program was our primary goal. In CS61B, 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.

Background Knowledge

This class assumes you have taken CS61A, CS61AS, 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 or CS61AS, 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 very quickly. Though the syntaxes of Java, Python, MATLAB, Scheme, etc. are enormously different, the underlying computational models

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 1st lab. We will also announce on-campus help sessions on the announcements section of this website. If you'd like a physical reference, Paul Hilfinger suggests *A Practical Guide to The Unix System (3rd edition)* by Mark Sobell (Addison-Wesley, 1994).

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.

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. Information about the staff running each section can be found on the staff page.

If you decide to permanently switch sections, it'd be best to switch in Telebears.

Attendance is optional for discussion. 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. Lab attendance is mandatory and missing a lab will result in a 10% penalty to that week's lab, though if you submit the lab early (by 10:00 PM on Wednesday), you may skip lab that week. You may skip attending any number of labs via early submission. During weeks in which labs are not graded, attendance is not required. Lab deadlines on gradescope will always reflect the "true" deadline (in the event that we need to push anything back for reasons like technical glitches).

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!*

Online Resources

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 (Josh 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

To talk with us, the best way is to come during regular office hours (posted on the home page). 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. Office hours are concentrated Monday to Wednesday because we hold labs all day Thursday and Friday.

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.

The optional textbook for the weeks 8-14 of the course is *Algorithms, 4th Edition* by Wayne and Sedgewick.

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 (Java SE 8 Edition) 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).

Software

This course does not have an official text editor. You may use Vim, Emacs, Sublime, or IDEs like Eclipse, Netbeats, IntelliJ, Sublime, etc. Whatever you use, however, your submitted solutions must conform to our expected layouts, as indicated in the assignments. We strongly 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 setting up your own computer is linked in lab 1b.

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.

HWs and Labs

There will be 14 labs and 7 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 Gradescope. All homeworks and labs are individual efforts (without partners).

Homework will be graded on a rigorous suite of correctness tests while labs will receive full credit for "reasonable effort," as evaluated by a small number of relatively simple correctness tests.Passing all tests on Gradescope for homework or labs will ensure full credit as there are no hidden tests.

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 homework assignment will be dropped and the two lowest labs will be dropped.

Programming Projects

There will be 4 larger programming projects. The first two projects will be optionally pair partner projects. The other two projects will be individual efforts, with the same collaboration rules as HW. You'll have later classes which will provide more opportunities to build your teamwork skills. For 61B, our goal is to help you build the

Project 0 and 1 will be relatively easier than projects 2 and 3, taking less time and with greater levels of scaffolding. Project 2 will be a very difficult project (on par with what you might expect from Hilfinger's harder projects). Project 3 will be challenging, but not as time consuming as project 2.

For Project 0, we will release all tests that determine your grade. In other words, passing all tests on Gradescope will allow you to earn full points for the autograded portion of the project.

The other three projects will have some restricted tests which will not be report to students on Gradescope. Thus, passing all tests on Gradescope 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.

Projects 0, 1, 2, and 3 will be worth 10, 25, 40, and 25 points respectively. There will be optional extra credit opportunities for projects 1, 2, and 3. Information TBA in the project 1 spec.

Exams

There will be two evening midterms on February 12th and March 31st and a final exam on May 11th. You will be allowed to bring one letter size page of handwritten notes (front and back) to the first midterm, two to the second midterm, and three to the final. You will not be required to turn in these sheets, and you may reuse them from exam to exam.

Inspired by the great Paul Hilfinger tradition, exams may cover any material whatsoever. For fear of our lives, exams will almost exclusively test material covered in the course.

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 (however, we believe we have avoided all of those).

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.

We release grades for exams on Gradescope. If you believe we have misgraded an exam, request a regrade on the same site with a note explaining your complaint. You should check the online solutions first to make sure that this regrade will make your total score go up as it is possible to lose points from a regrade request.

If you do especially poorly on Midterm 1 (fewer than 10 points), then you will have an opportunity to take a makeup midterm 1 that can boost your score to a maximum of 10 points. Your midterm 1 score will be the better of your actual midterm 1 and the makeup.

If your midterm grades are statistically much worse than your final, we'll replace your midterm grade.

Midterm (both MT1 and MT2) grades can be "shadowed" (aka a weaker version of what Dan Garcia calls "clobbering") by the final. The way it works is that if you are X standard deviations from the mean, your midterm scores will be replaced by a score equivalent to X - 0.5 standard deviations from the mean. This policy can only help, and cannot hurt your score.

Effectively, this only applies if you improve substantially on your final: an improvement of over 0.5 standard deviations.

For example, suppose Bilbo scored 0.4 standard deviations above the mean on MT1, 0.9 standard deviations above the mean on MT2, and 1.1 standard deviations above the mean on the final. Then Bilbo's midterm scores will be replaced by 1.1 - 0.5 = 0.6 standard deviations above the mean, as long as this is an improvement.

Since Bilbo already got 0.4 above the mean on MT1, his score would be replaced by 0.6. However, his score on MT2 would not be changed (since he is already 0.9 above the mean).

In Bilbo's case, his MT1 score would be replaced by -0.2, and MT2 score would be untouched (since it is already better).

Or in pseudocode:

```
your_devs = (your_final_score - final_mean) / final_stddev
your_potential_replacement = (your_devs - 0.5) * midterm_stddev + mi
your_shadowed_midterm_score = max(your_midterm_score, your_potential_
```

Grades

Your letter grade will be determined by the total points out of the possible 384. 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	~22%	84
Projects	~26%	100
Midterms	~26%	100
Final Exam	~26%	100
Total	100%	384

A+	A	A-	B+	B	B-	C+	C	C-	D+	D	D-	F
374	348	327	309	285	270	248	215	184	160	135	100	0

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 55% on tests (typical for past exams), 75% on projects, and 100% on homework and lab assignments.

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.

Extra Credit

1. 1 point each for projects 1, 2, and 3. Details in project specs.
2. 1 point for mid-semester survey.
3. 1 point for taking our end-of-semester survey.
4. 2 points for taking the HKN survey.

Gold Points Boosting

On projects 2 and (probably) 3, you will have the opportunity to earn gold points. These act sort of like extra credit, though the higher your exam scores, the less your gold points will count. You should only pursue gold points if you really enjoy the projects, as the amount of credit you'll earn per hour spent is relatively low compared to other facets of the course (e.g. exam studying).

If you earn g gold points on a project and T points on all exams (after taking into account shadowing), then your total score in the course will be boosted by 2*(g - g * T/200).

For example, if you earn 4 gold points on project 2, and have 120 points on your exams total, then you'll earn a gold boost of 2*(4 - 4 * 120/200) = 3.2 to your total score in the class.

If we decide a project is too difficult, we reserve the right to move some part of the project into the gold parts section. In such a case, we may readjust the distribution of gold points.

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 (I'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.

HW and Lab Collaboration Policy

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.

Cheating on a homework or lab will result in a 25 point penalty on your total score. Cheating a second time on a homework or lab will result in an F in the course. All incidents of cheating will be referred to the Office of Student Conduct.

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 (or with your partner for the first two projects), 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. As above, 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:

1. **By You Alone**: All project code that you submit (other than skeleton code) should be written by you (and if applicable, your project or project 1 partner) alone, except for small snippets that solve tiny subproblems (examples in the Permitted section below).
2. **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

else)! If you're not sure what you're doing is OK, please ask.

3. **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 dictacting 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,

We have advanced 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 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.

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 1200 students, we cannot afford to handle individual cases, though if you have an extreme situation that warrants our attention, please contact one of the head TAs (Sarah, Leo, Jimmy).

For the first day after a programming project is due, we'll penalize an assignment 1/12 percent for each hour it is late. After the first 24 hours, you'll incur a penalty of 5/12 percent (0.417%) for each hour late, rounded off in some unspecified fashion. This means that you will lose roughly 10% of the points for a project each day late (though tracking is by hours late, not days). Late submissions will be submitted through the special "Late" version of each project on gradescope.

Acknowledgements

Some course handout material derived froms Paul Hilfinger's CS61B handout.