CS 61B Data Structures, Spring 2015

Instructor: Josh Hug
Lecture 1: MWF, 2-3 PM, 1 Pimentel
Lecture 2: MWF, 11-12 PM, 150 Wheeler

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 and linked list manipulation.

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 are surprisingly similar.
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. However, if the section is officially full or doing so would somehow cause you significant grief, it's OK to simply clear it with the two TAs involved.

Attendance is optional for discussion, but may be taken into account if you're on a grading boundary. 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 any number of labs via early submission. During weeks in which labs are not graded, attendance is not required.

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 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). 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.

Course Materials

The textbook for the first six weeks of this course is *Head First Java, 2nd Edition* by Sierra and Bates (O'Reilly, 2005).. This book is highly recommended to those of you with no Java experience. It's an easy read, and is super cheap for a textbook.

The textbook for the weeks 7-13 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 7 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.

This semester, we will be using Java 7. It is already installed on the lab computers, and it may be provided with your computer as well. If it is not already installed you can download JDK 7 from the Oracle site. Information for setting up Java on your home computer will be linked in lab 1B.

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 remotely. Information for doing so will be linked in lab 1B. All of the tools we use in the course should be relatively easy to set up, except for the gjdb debugger, which will only work on Mac OS X or Linux. More on this in a later lab. Rather than working natively on your own computer, you may find it more convenient

to simply use the instructional machines remotely by logging in from home. Details will provided 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, in later, team-oriented courses (as well as the real world), version-control systems help manage collaborative work.

You will be learning and using Git for this course to tracking and submit your work for grading. The first lab will teach you the basics of what you will need to know. Feel free to also read Git documentation.

Computer Accounts

Every student will receive a computer account form in the first week. More information on obtaining these forms will be announced on the course site.

You must electronically register the account you intend to use for handing in assignments (only one account, please) by the second week (by Friday's lecture). The class web page contains a link with which you can register your account and perform other adminstrative actions. Please use it to give your registration information and to generate the keys you need to work remotely and hand in homework assignments.

When logged into our instructional systems for CS 61B work, please make sure that you are using the standard configuration for the class -- that is, the files `.bashrc`, `.emacs`, etc., that should have been in your accounts initially. If you do modify these files, you are on your own.

HWs and Labs

There will be 14 labs and 9 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 (as described in lab 1). All homeworks and labs are individual efforts.

Homeworks will be graded on correctness. Most homeworks will have two sets of automated tests. The 'release' test will be available immediately upon release of the assignment. When you submit your homework, you will receive an email with feedback from the 'release' test. There will also be a set of 'secret' tests that will be run two days after the homework is officially due.

You will have until the end of the semester to correctly pass the secret tests. Half of each homework will be based on passing the 'release' test, and half on the 'secret' tests. *Note: This policy is under revision and is going to be made somewhat friendlier. Details soon (noted on: Feb 25th, 2015).*

We grade projects and homework *on the instructional machines.* Very often, you will find that for various reasons (different software versions, different operating systems), a Java program

will work on your home machine, but not on ours. Be sure to test your software on the instructional machines before submission.

Lab grading schemes vary. Most labs will be on correctness, but there will be no secret tests. Some of the labs are freebies (during project weeks).

Homeworks will be due at 10 PM on their respective due dates. Labs are due by 10:00 PM each Sunday after they are assigned. There is a short (unspecified) grace period for homeworks. There are no slip hours or slip days for homeworks. Your two lowest homeworks/labs will be dropped (for a total of two drops, not four).

Programming Projects

There will be 4 larger programming projects. Projects are individual efforts, with the same collaboration rules as HW. You'll have later classes in which 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.

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.

Each project will have three graders: 'basics', 'release', and 'secret'. The basics autograder will be running as soon as the assignment is out, and will provide verbose output. Successful completion of the basics autograder before the basics deadline will yield a point of extra credit.

The release autograder will also be running when the assignment is released, but it will provide less detailed feedback. The results of the secret autograder tests will not be given out, and are only used for grading purposes.

Projects 0, 1, 2, and 3 will be worth 10, 18, 24, and 18 points respectively. Projects 2 and 3 will each have 5 points worth of gold points. These will be especially challenging parts of the assignment. If you opt not to do the gold point parts of the projects, we will reweight your project by 14/13ths.

Exams

There will be two evening midterms on February 18th and April 1st, and a final exam on May 12th. You will be allowed to bring one page of handwritten notes (front and back) to midterm 1, two pages to midterm 2, and three pages to the final. Pages must be of reasonable physical dimensions. 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).

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.

Midterm (both MT1 and MT2) grades can be "shadowed" (aka a weaker version of "clobbered") by the final. 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 his MT1 score would be replaced with the equivalent of 0.6 standard deviations (1.1 - 0.4 = 0.6) above (pre-shadowed) the original mean. Bilbo's MT2 score remains untouched (1.1 - 0.9 = 0.2, which is not greater than 0.5). In conclusion, you can shadow a bad midterm if you show substantial improvement, but that shadowing can only help you to a certain degree.

With the obvious exception of emergencies, you must arrange alternative exam times well in advance. Information will be provided on how to request alternate times before each exam.

Grades

Your letter grade will be determined by the total points out of the possible 233. 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. Grading bins are as follows:

Category	**Points**
Projects	80
Homework/Labs	23
Midterms	70
Final Exam	60
Total	233

A+	A	A-	B+	B	B-	C+	C	C-	D+	D	D-	F
226	210	196	184	170	161	151	138	126	115	?103	90	< 90

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 60% 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

There will be a total of 8.1 points of extra credit in the course:

1. 1 point each for projects 0, 1, 2, and 3 basics autograders.
2. 1 point for mid-semester survey.
3. 3 points for taking the HKN survey.
4. 0.1 points for fast put operation in project 1 (spec error)

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.

In CS61B, we have three types of assignments: homeworks, labs, and projects. The entire point of homeworks and labs is to learn. You should feel free to collaborate with others, though keep in mind that greater independence is likely to give you a better learning experience (as long as you aren't totally stuck).

By contrast, the projects were designed for learning (particularly how to be self-reliant in the context of large unfamiliar systems), as well as 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.

Cheating on a project will earn you a negative score for that project, i.e. if a project is worth 127 points of your course grade, you will instead lose 127 points from your final point total. Cheating a second time or on an exam will earn you an F in the course. 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 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 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)! 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. 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.

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.

Submissions that are someone else's work, but which are appropriately cited, will be given zero points (as opposed to a negative score and a referral to the 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.
The above rules apply for projects. When it comes to HW and labs, you should strive to do as much as you can alone, but it is OK to work with others (even sharing code, although we recommend that you do this very sparingly) if you feel that you'll learn better that way.

Nonetheless, you should not submit other student's HW or lab code as your own, and identical or near identical submissions will be treated as plagiarism. Though we have not enumerated a specific penalty for plagiarising HW and labs in this document, we reserve the right to enact an appropriate penalty from the departmental policy on academic dishonesty.

Lateness

We will give no credit for homeworks or labs after the deadline. Your lowest two scores on hw/labs are dropped (for a total of two drops, not four). This is intended to handle any contingencies. With 1200 students, we cannot afford to handle individual cases, though if you have a more extreme situation that warrants our attention, contact one of the head TAs.

On programming projects, we'll penalize an assignment 5/12 percent (0.417%) for each hour it is late, rounded off in some unspecified fashion.

For each of your four projects, you'll receive 24 grace hours. Any hours unused will roll over to the next project. Thus, if you don't use any of your grace hours until the final project, you'll have 96 grace hours to use on that project. What a deal!

Acknowledgements

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