

## Course Overview<sup>1</sup>

# CS61B: Data Structures and Advanced Programming

## 1 Introduction

In this course you will learn many data structures, algorithms and software engineering principles that are used daily by professional software developers. There will also be more of a focus on learning good and more “real world” programming practices through the completion of more involved and larger programming assignments. Assignments will all be completed in (and you will learn the fundamentals of) Java, a language that is used widely in industry. For these reasons, many students find this course to be the most practical of the lower-division sequence in preparation for programming-heavy upper-division courses and a basic programming job. Simultaneously, formalisms for analyzing said data structures and algorithms will also be introduced, which will prepare you (along with a course in discrete mathematics) for CS170 or equivalent introductory upper-division course in algorithms. Although you will probably find many of the concepts introduced in this course to be more straightforward than those in 61A, you will probably find the programming workload heavier.

Because the same amount of material will be covered in lecture and the same amount of homework and projects assigned during the summer in 8 weeks as would normally be done in 15 weeks during the spring or fall, this course will move quickly. Although CS61B is known as having a heavy programming workload during the semester, in the summer each homework will be about twice as long and the projects need to be done in roughly half the time. Despite this extra work load, many students in the past preferred taking the course in the summer, because they were able to focus on 61B and not be distracted by other demanding courses.

On the course website <http://inst.eecs.berkeley.edu/~cs61b>, throughout the semester you will find recent announcements as well as all course handouts (including this one).

## 2 Goals

By the end of the course, students will...

- be familiar with programming in the more strongly typed language, Java, and with Object Oriented Programming (OOP) concepts,
- be able to design, write, and debug moderately sized programs,
- understand the basics of algorithm analysis with regard to its formalisms, its conceptual nature, and its practice in software,
- and be familiar with many standard data structures and algorithms.

## 3 Prerequisites

CS 61A or E77 (or an equivalent) is formally a prerequisite for this course. While during the fall and spring semesters, this requirement is strictly enforced, during the summer anyone is permitted in the course. I will assume a knowledge of 61A from students, but if you do not have this background you should still be able to get an A+ in this course. Because this is the most likely course in Cal’s lower-division sequence to have an equivalent at another university, students visiting for the summer may have the best chance of transfer equivalency with this course (of course, consult your home university). While Cal students who intend to study EECS or CS and who have not taken 61A are urged to take that course instead, we will do nothing to stop you from taking this course.

---

<sup>1</sup>Thanks to Barath Raghavan, Mike Brudno and Paul Hilfinger for providing versions of this document in previous semesters, from which I have drawn both inspiration and text.

## 4 Staff and Schedule

**Instructor:** Jeffrey Schoner, jschoner @ cs .dot. berkeley .dot. edu

Note: Please call me “Jeff”.

**Teaching Assistants (TAs):** Steven Kulaso, Sarah Moussa, David Turner

**Readers:** Munan Gosalia, Reader # 2

**Lab Assistants (LAs):** Melody Hsu, Daniel Kim, Colleen Lewis

This course takes place during Summer Sesion C, which runs from June 21 to Aug 13.

All meeting times begin ten minutes after the hour (so-called “Berkeley time”). Lecture meets Monday through Thursday, 11–12:30 (6 hours per week) in 2050 Valley Life Sciences Building (VLSB). All discussions (2 hours per week) meet in 320 Soda, all laboratories (4 hours per week) in 275 Soda according to the following table.

Section	Lab	Discussion	TA
101	MW 1-3	TuTh 1-2	David
102	MW 3-5	TuTh 3-4	Steven
103	MW 5-7	TuTh 5-6	Sarah

No labs or lecture will take place on Monday, July 5th, due to the observance of Independence Day.

Up to date staff office hours as well as a tentative syllabus with all lecture topics, due dates, exam dates, etc. can be found on the course website.

## 5 Enrollment

First, we do not care too much about what Telebears says. You simply need to be enrolled (not on the waitlist) in the course somewhere and have logged into your instructional account. Students are urged to get off the waitlist by switching to an open section. Do not use Telebears to move to another section, except to get off the waitlist! If you would like to switch sections, get in contact with the TA assigned to your enrolled section as well as the one assigned to the section you would like to change to.

For discussion sections, space is not usually a problem even for overenrolled ones. Nonetheless, it is important to remember that you will get more personal attention if you’re attending a smaller section. For laboratory sections, there are only so many computers in the room, hence we have to limit enrollment. Remember as well, that in a smaller section, you will get help and your labs checked off more quickly. We highly recommend that you attend a corresponding laboratory/discussion pair, but you are only required to attend the same lab every week. For these reasons, as soon as possible you need to confirm with the TA for your desired lab section that you are on their roll.

## 6 Course Materials

Basically, you will benefit from having some data structures texts to support the lecture material. Depending on your prior familiarity with Java or related languages like C++, a good book (of your preferred style) on Java may be helpful. While the lectures will cover all of the material for the course, these books go into more detail and can be consulted before and after lecture for added clarity. Furthermore, details in the assigned reading, which were not covered in lecture are still fair game for exams.

**Available through your favorite local textbook shop or online bookseller**

- *Data Structures and Algorithms in Java* by Michael Goodrich and Roberto Tamassia, third edition (2003), published by Wiley Text Books (ISBN: 0471469831). GT is the primary text book for this class each semester. Officially, it is **required**, although I will only give reading assignments (and not assign problems) from it. Old editions are ok, but I believe they may have different page and section numbering. Additionally, different editions

cover slightly different topics. Perhaps counterintuitively, some of the older editions better cover material from this course than the newest one.

- *Introduction to Programming Using Java* by David Arnow, Scott Dexter and Gerald Weiss, second edition, published by Addison-Wesley (ISBN: 0321200063, but other formats are available too). ADW is **optional** (despite it being designated “recommended” by the bookstore). It is a fairly solid Java introductory text book, but we won’t use most of it. You’re likely to be able to get the same information out of another (cheaper) introductory Java book. Some older editions of this book (make sure they are for Java 2) are also ok, but are written only by Arnow and Weiss.

#### Available through northside Copy Central (2483 Hearst)

- *Data-structures (into Java)* by Paul Hilfinger. DSiJ supplements GT in that it covers more of the material from this class and is more formal and detailed. It is essentially **required**, since I will be assigning reading from it. It costs \$18.67. Unfortunately, there was a mistake made with the title on the cover. The cover says “Computer Science 61B (Java)” although it is the data structures text. It is also listed as reader number 8 (upper right hand corner of the cover).
- *Programming into Java* by Paul Hilfinger. PiJ is a **recommended** textbook on Java. It tends to be more formal than AW, but is more attuned to this course. The cover says “Computer Science 61B Programming” and it is reader number 7. It costs \$20.33.

Various versions of these two texts exist from previous offerings of this course between 2000 and 2002 that are fine, although there are some mistakes that have been corrected in more recent versions. Up to date versions are also available in PDF form from the course home page, but please do not print them out (they are around 500 pages combined). Save yourself time, energy, and toner/ink cartridges (therefore money) and buy the printed copies instead. Despite the title errors on the cover, the first page of both texts contains the correct title.

#### Other books of interest

- *Thinking in Java* by Bruce Eckel, third edition. This is one particular comprehensive book (of many out there) for learning Java. At 1119 pages, TIJ is a tome that thoroughly covers a broad swath of Java (and OOP). While the print copy (ISBN: 0131002872) is quite affordable, it is also available in a free to download version at <http://www.mindview.net/Books/TIJ/> for the true budget shopper.
- *Introduction to Algorithms* by Thomas Cormen, Charles Leiserson and Ronald Rivest, published by MIT Press (ISBN: 0262032937). CLR is actually the book for CS170 and has no official assignment for CS61B. However, it does cover most of the data structures and algorithms material for this course (plus much, much more). In fact, it covers some course material that is not in the other texts at all. Furthermore, it is well written, detailed and concise, but assumes some more mathematical sophistication.

A lot of people would say it’s one of those books that every aspiring computer scientist should own. You may want to consider consulting this book occasionally on certain topics. The engineering library has some copies or you may have friends you can borrow it from. If you have money to burn, are taking CS170 soon or something like that you could also buy a copy, but it’s really in no way necessary for this course. The second edition (from 2001) has some additional material (relevant to this course on randomized and amortized analysis) lacking in first edition (from circa 1990).

We won’t be using GT or DSiJ for about two weeks, but you should get your hands on a Java book as soon as possible.

## 7 Computer Issues

### 7.1 Computer Accounts

Students will receive instructional computer accounts from their TA in lab during the first week of classes. They are required for all students, as all assignments must be submitted through these instructional accounts. 275 Soda is reserved for 61B during scheduled lab times. While you may use computers in any of the UNIX instructional labs

(such as the others on the second floor of Soda Hall) for completing your assignments, please be cautious about using a lab while it reserved for a different class (such as 271 Soda for CS61C). If other labs are full and the reserved lab is not, ask the TA in charge if it's ok that you work there. You must cede your workstation in the case where a reserved lab is full and a student in the class for which lab is reserved does not have a fully functional workstation. Similarly, if you come to a 61B lab and there are no available workstations, kindly ask students not in the course to leave (or have the TA do it for you, if you don't feel comfortable doing that yourself).

We will assume that you know the basics of using UNIX for this course. If you are not familiar with it, the instructor and TAs can recommend resources for getting you up to speed.

## 7.2 Lab Access

The EECS instructional laboratories are open from 7am to 6:30pm on weekdays and are locked all day on weekends and holidays.

Once you have an instructional computer account and are enrolled in the course, you are eligible to get a key card, which will give you 24 hour access to the labs. To get one, bring your student ID and \$20 (\$15 of it is a refundable deposit) to 387 Soda or 253 Cory (open weekdays 8-noon, 1-4:30). If you have a card from a previous semester, you may need to visit the above office to have it reactivated for the summer.

**I highly recommend getting a key card.** Most students will find it convenient to do work outside of the hours the lab is open.

See <http://inst.eecs.berkeley.edu/usr/pub/cardkey.help> for more information.

## 7.3 Working from Home

You can of course work at home, so long as you have a some sort of Java development environment installed on your computer (download J2SE 1.4.2 from <http://java.sun.com> if you want the same environment we'll be using in class). However, you should note a few points:

- Many students find it easier to focus when working in the lab.
- You can ask other students in the lab for immediate help if you're having trouble with an assignment.
- How your program runs on the instructional machines in 275 Soda is all that matters. Should you develop a solution at home, leave plenty of time for testing it on the lab machines.

## 8 Coursework

Because this course is offered during the summer in a compressed format, it is especially important that you start on all assignments as soon as possible. This is particularly true for projects, since they will require a non-trivial amount of work.

All handed in work will be graded for correctness. Where possible, we will use automated testing procedures (also known as "the autograder") for determining program correctness. Because of this, you should pay close attention to specifications given in the handout for each assignment. While a human reader will examine the results of all automated testing and do further assessment of your program for factors such as documentation and commenting, they will not give special consideration in cases where your program cannot be automatically tested due to you not following the specification. Where possible, we will instate basic tests that will be run at submission time. You will immediately obtain the results of these tests, so that you can make changes to program and resubmit before the deadline. However, these tests will not be comprehensive, but rather aimed at making sure your program does not contain errors that result in total program failure during testing.

## 8.1 Homework Assignments

There will be a homework assignment due at 9pm every Sunday (starting the Sunday after the first week of class), except for the Sunday that falls on July 4th, in which case the assignment will be due the next day. Topics covered in lecture the preceding week will form the basis of the homework exercises. Most problems will be programming exercises, but there will be other exercises which will require answers in English or as a formal proof. All homework assignments must be done individually. Normally, they must be turned in electronically using the submit program; email copies are not acceptable. If there are assignments (or parts thereof) which require or permit paper responses to non-programming exercises, this will be explicitly stated in the assignment handout.

## 8.2 Projects

Each project will be due on a Friday at 9pm. Some projects may have an earlier priority deadline, which if met, will allow you to get automated testing feedback on your submission before turning in an improved version for the final deadline. The goal of doing projects is for you to get experience designing, writing, and debugging larger programs. There will be 3 projects over the course of the semester. There will be at least 2 weeks between the assignment and due date of a project, as well as 2 weeks between the due dates of any adjacent projects. They must be turned in electronically using the submit program; email and paper copies are not acceptable. If students are allowed to work in pairs or not will be specified on all project handouts.

## 8.3 Exams

There will be 3 exams: 2 midterms and a final. All exams will be given in the evening to allow plenty of time. Currently, the midterms are scheduled for July 6th and July 27th from 6pm to 8pm and the final is scheduled for August 12th (time TBA). All of the exams will be open notes and open book, but electronic devices are not allowed. Please arrive at the exams slightly ahead of the scheduled time and make sure to bring a Cal student ID or government issued photo ID.

If you require special accommodations for an exam, please contact me about one week beforehand. If you have a conflict with these times, please try to reschedule your conflicting event if at all possible. If that's not possible, please let me know a week in advance, so that something can be arranged.

Exams will be returned in your laboratory or discussion section. Every effort will be made to grade all exams within two days. If you feel a mistake was made while grading your exam, please return it along with a note explaining your concern to a TA. We reserve the right to regrade the entire exam.

## 8.4 Lab Assignments and Sections

There will be required laboratory assignments (labs), which must be checked off by your TA during your scheduled lab section for full credit. The lab for a given week is due by the end of the last meeting of your lab section that week, but TAs will be available to check off and offer help during both meetings. There will be no labs during the final week of class. Most labs will be fairly straightforward and are designed to prepare you for the projects and homeworks. They can be done alone or in a group of two, but both group members should be prepared to answer questions about all of the lab.

## 8.5 Discussion Sections

While each TA is more or less free to run their sections as they see fit, in general the focus will be on helping the students understand the lecture material for exams. This will likely involve answering student questions about assignments or lecture material, as well as leading the section through sample exam-style problems.

## 8.6 Office Hours

Students should use instructor and TA office hours for asking any questions they have about the course such as those dealing with lecture material, additional reading, assignments, exam question prognostication and grading. You may visit the office hours of any TA, not just the TA for your section.

## 9 Lateness

You have 48 hours of lateness (or so-called “slip hours”) for the entire semester. You may use these in 1 hour chunks for homework or projects to avoid having your score reduced (with restrictions outlined below). However, you will not get any extra credit or a big prize if you don’t use them all up.

Homework assignments are due Sundays (except for the July 4th holiday, when HW2 is due Monday) at 9pm. **Late homework will not be accepted after the start of the next lecture at 11am.** You may use as many slip hours as you need up until that point. However, if you do not have enough to cover the number of hours late, you will only get half credit for the assignment and your slip hour count will not be reduced.

**Late labs will not be accepted.** I strongly encourage you to start the labs early and get them checked off before the second lab if possible.

Projects are due Fridays at 9pm. You can use as many of your remaining slip hours (in 1 hour chunks of course) as are necessary for any project. After they run out, your reduced score,  $s(h)$  ( $h$  is the number of hours late reduced by any slip hours used), will be defined as follows,

$$\forall h > 0, \quad s(h) = \lfloor s(0) \cdot 2^{-\lceil h \rceil / 24} \rfloor,$$

where  $s(0)$  would be the score given if no late penalty had been applied. In plain language, more or less your score exponentially decays (cue bad jokes about the nuclear meltdown of your grade) with respect to lateness, losing half credit every 24 hours. **Late projects will not be accepted after the start of the next lecture on Monday morning at 11am.**

With all of the above times, there is a 5 minute grace period. So, if you need to turn something in at 21:00 for full credit, 21:03 is still full credit, but at 21:05:01 is late.

### 9.1 Project Lateness Examples

- handed in Friday night at 21:30 ( $h = 0.5$ ): 1 slip hour and no penalty *or* 0 slip hours and 3% off
- handed in Friday night at 23:15 ( $h = 2.25$ ): 3 slip hours and no penalty *or* 0 slip hours and 9% off
- handed in Saturday morning at 9:30 ( $h = 12.5$ ): 13 slip hours and no penalty *or* 0 slip hours and 32% off
- handed in Sunday morning at 8:00 ( $h = 35$ ): 35 slip hours and no penalty *or* 0 slip hours and 64% off *or* (for example) 24 slip hours and 28% off
- handed in Monday night at 21:00 ( $h = 72$ ): no slip hours lost and no credit

## 10 Getting Help

Of course, the instructor and the TAs are there to help you if you have problems or questions. Naturally, we cannot give you the answers to assignments before you turn them in, but we can help you help yourself. Students should feel free to contact any of the course staff in person or via email for questions regarding assignments, lecture material, or other problems or questions.

While you can email the entire CS61B staff at `cs61b@imail.eecs.berkeley.edu`, you are encouraged to use the newsgroup `ucb.class.cs61b` for questions that are not private in nature. With the newsgroup, both the staff and your fellow students can answer questions, where everybody can see the answers.

## 11 Grading

### Tentative breakdown

7 homework assignments	× 5 points each	=	35 points
3 project assignments	× on average 25 points each	=	75 points
2 midterm exams	× 20 points each	=	40 points
1 final exam	× 50 points	=	50 points
total		=	200 points

**Important Note on Labs** Here's how the 7 laboratory assignments I expect to give this semester figure in. The stick: you can only get an A- (respectively B+) or higher, if you successfully get at least 5 (respectively 4) of the 7 labs checked off. The carrot: if you get all 7 labs checked off, your grade will be bumped up to the next half-grade *if you are within 5 points of it* and if you get at least 6 labs checked off, your grade will be bumped up to the next half-grade *if you are within a point of it*. People who are in the A+ range and complete all the labs get a warm fuzzy feeling of accomplishment. Since you never know how close you will be to the next grade, it is worth doing all the labs grade-wise.

**Grading Scale** With that in mind, grades will primarily be determined according to the following table showing the minimum number of points required for a corresponding grade.

A+	A	A-	B+	B	B-	C+	C	C-	D+	D	D-	F
190	175	165	155	145	135	125	115	110	105	95	85	

With this scale, a student receiving near 100% on homeworks and projects, but 50% on exams will receive a B or B+ (depending on how many labs are completed). Conversely, a student earning 50% of the homework and project points, but who earns 90% on the exams will receive a B-.

Note that there is no curve in the class; you are not directly competing with other students for your grade. However, while this scale is not fixed, if it is changed it will only be in your favor (some of the cutoffs may be lowered). This means that the above ranges are simply the maximum cutoffs.

Please keep in mind that it is likely the average score for an exam will be somewhat lower than an average score on a homework or project. In other words, getting perfect on a homework should not be terribly difficult or unlikely for most students, while probably nobody will get perfect on the final exam (prove me wrong!).

## 12 Policy on Collaboration and Cheating

We feel that collaboration amongst students in this class is one of the most important resources available for understanding the material. However, it is necessary to draw a line between reasonable collaboration and outright cheating (academic dishonesty in all forms). You should feel free (and are encouraged!) to discuss potential solutions to assignments and questions about lecture material with other students in the class. **If you have a question about the legality of a form of collaboration, don't hesitate to ask the instructor or a TA.**

As a guideline, it is fine to discuss a general plan of attack or ideas for solving a problem with others. However, after one of you has begun actually coding or writing an assignment, you need to do your own work. Another major guideline is known as the "no code rule". It says you should never be in possession of another student's code, nor should your code be in the possession of another student.

Obvious cases of cheating include (but are not limited to) turning somebody else's (this person does not have to be in the class) code as your own, giving your code to somebody else or working with another student to develop the same solution and handing it in under each of your names. Perhaps less obvious ones include debugging another student's program (although explaining an error or how to use a piece of software is fine) and editing another student's program in any way for them.

For all assignments, we will specify the number of people that are allowed to work together in a group. In the case of assignments where groups are allowed, members of each group are of course allowed unlimited collaboration with

each other for that specific assignment. However, groups must abide by the same rules as individuals (for example, no sharing code between groups).

Rest assured that we will use sophisticated software to check your submissions for copying.

Possibly the most dispicable cases are those of sabotage, which include intentional malicious modification of another student's solution as well as intentional malicious damage to or obstruction of access to (physically or otherwise) facilities or equipment which limits learning opportunities for other students. On the less severe side of things, this includes using a screen lock on a terminal for an extended period of time, while performing a denial of service (DOS) attack on instructional equipment is obviously more severe.

Students caught cheating will be punished. At the very least, a negative maximum score will be given for cheating on a homework (for example, -5 points). Cheating on an exam or project will likely result in a failing grade in the course. I reserve the right to report any cases of cheating to the relevant authorities.

## 13 Frequently Asked Questions (FAQ) with Answers and Frequently Overheard Statements (FOS) with Responses

**Q:** I'm only 1 point away from a B+! Can you *pleeease* raise my grade?

**A:** Generally no. That's what the lab incentive is for.

**Q:** I'm currently on the waitlist. What should I do?

**A:** If possible, enroll in another section with space. If not, keep doing the assignments and going to class as if you were enrolled. I hope to enroll everybody at some point.

**Q:** Why are the late penalties so harsh?

**A:** First, you have slip hours already, so penalties after those are used up need to be a bit more significant. Second, because this class is during Summer Session, readers have very little time to grade assignments. Without any lateness, they have on average about 5 days to grade each of the 7 homeworks and 3 projects. Any extra allowed lateness would cause the readers to get further behind in their duties.

**Q:** I had *this thing* to do last week and couldn't turn the assignment in on time.

**A:** First, unless this is something severe and unplanned (death in the family, emergency hospitalization of you, etc.), this is what slip hours are for. Don't ask for extensions otherwise, because I won't give you one. Second, you should begin all assignments (but especially projects) early in case *things* do come up.

**Q:** Why are assignments due at 21:00?

**A:** Pulling all nighters has a cherished history in CS courses. In my day, I even knew several students who were somehow proud of their ability to complete an entire assignment the 12 hours directly proceeding a noon deadline. Frankly, this behavior doesn't show you're "hardcore", certainly not that you're a good student, but rather that you are a major procrastinator. So, I intend to discourage it, particularly when there is lecture the next day. If you want to stay up all night working on the projects, you could do it for all three and still have slip hours to spare, but, hey, it's your Friday night.

**OS:** I've heard classes are easier in summer school than during the semester.

**R:** It depends. We cover the same material and do the same amount of coursework in half the time, so in that way it is more intense. On the other hand, you probably aren't taking any other classes, so you should have more time to focus just on this course, which may make things less hectic overall.

**OS:** I'm taking this class now, because Professor Hilfinger is teaching it next semester and I hear he's really hard.

**R:** See the response to the previous FOS. Although I doubt taking this class will earn you the same bragging rights, I would like to add that I hope to teach this class with a similar rigor as the dear professor. Plus, *real* CS students aren't afraid of a little hard work, are they? Most students notice that if they work hard, they learn more and leave with a sense of accomplishment.