Syllabus

Course Meeting Times

Lectures: 2 sessions / week, 1.5 hours / session

Recitations: 1 sessions / week, 1 hour / session

Course Description

6.005 Software Construction introduces fundamental principles and techniques of software development, i.e., how to write software that is safe from bugs, easy to understand, and ready for change. Topics include specifications and invariants; testing, test-case generation, and coverage; abstract data types and representation independence; design patterns for object-oriented programming; concurrent programming, including message passing and shared concurrency, and defending against races and deadlock; and functional programming with immutable data and higher-order functions. Includes weekly programming exercises and larger group programming projects.

The 6.005 website homepage from Spring 2016, along with all course materials, is available to OpenCourseWare users.

Class

Class meetings

There are two 90-minute class meetings each week on Monday and Wednesday and one 1-hour class meeting each week on Friday. You are expected to attend all class meetings and to participate actively in exercises and discussions.

Laptops required

Classes will include multiple-choice questions and programming exercises that require a laptop.

Readings

Most classes will have a reading that you must read before coming to class. There is no course textbook.

Nanoquizzes

Every class meeting will begin with a short quiz on the required reading for the class, plus recent class meetings. Nanoquizzes are closed-book and closed-notes, with a 3-minute time limit. There will be approximately 25 nanoquizzes. Your lowest 5 nanoquiz grades will be automatically dropped, and you can make up missed nanoquizzes or low grades. More info can be found on the nanoquiz grading and makeup page.

Project

You will complete a group software development project at the end of the semester. The project will be done in teams of three students. Each team member is required to participate roughly equally in every activity (design, implementation, test, documentation), and we may ask for an accounting of what each team member did. A single grade will be assigned to all members of the team.

Team Meetings

During the project, you and your project team will meet with your Teaching Assistant to discuss the work. Your TA will assign a grade based in part on this meeting. Team meetings will usually be scheduled during class times that will be reserved for this purpose.

Quizzes

There will be two quizzes, on dates specified on the course calendar. Each quiz will be comprehensive, drawing on any topics covered up to that point in the course, so e.g. Quiz 2 may include topics that were already covered on Quiz 1.

Quizzes are closed-book, but you may bring a single 8.5×11” double-sided page of notes, readable without magnification, that was created by you. Since the process of creating a crib sheet is most of its benefit, you may not share these notes or use someone else’s.

An archive of past quizzes is available.

Grading Policy

Grades will be roughly computed as follows:

ACTIVITIES	PERCENTAGES
Quizzes	30%
Problem Sets	50%
Project	10%
Code Review	5%
Participation	5%

Letter grades are determined at the end of the semester. The default cutoffs are: a final average of 90 and above is an A, 80 and above is a B, 70 and above is a C. These boundaries may be adjusted downwards if necessary because of the difficulty of the assignments or quizzes, but the boundaries will never be adjusted upwards, so a final average of 90 is guaranteed to be an A. The boundary adjustment is done heuristically, and there are no grade quotas, no grade targets, and no centering of the class on a particular grade boundary.

Every student is considered individually in the final grading meeting, judging from their entire performance in the course. A single bad mark in an otherwise consistent record will often be discounted.

Problem Set Grading

The overall grade for a problem set will typically be computed as follows:

overall grade = 40% × beta-autograde + 45% × final-autograde + 15% × manual-grade

where beta-autograde and final-autograde are determined by automated tests, and manual-grade is determined by graders reading the code. One part of manual-grade is fixing code in response to code reviews.

The breakdown may vary by 5-10% from problem set to problem set, depending on how much of the problem set can be autograded and how much requires human eyes. Each problem set’s breakdown will be announced when final grades are released.

Problems Sets, Beta & Final Problem Set Deadlines, and Slack Days

Problem Sets

To consolidate your understanding of the ideas from class, you will do five problem sets, PS0 to PS4, involving both design and implementation work. Problem sets will be done individually. Please review the page on collaboration on problem sets.

Code review

As part of each problem set, there will be a 2-day code-reviewing period when other students and staff will give you feedback about the code you submitted, using a web-based system. You will be expected to participate in this process by reviewing some of your classmates’ code. More details about objectives and guidelines for the code reviewing process are available.

Course Info

INSTRUCTORS
Prof. Robert Miller
Dr. Max Goldman

DEPARTMENTS
Electrical Engineering and Computer Science

AS TAUGHT IN
Spring 2016

LEVEL
Undergraduate

TOPICS
Engineering
Computer Science
Algorithms and Data Structures
Computer Networks
Software Design and Engineering

LEARNING RESOURCE TYPES
Problem Sets
Exams with Solutions
Online Textbook
Programming Assignments

Download Course

Feedback

Beta and final submission

Each problem set will have beta and final submission. The beta submission will be graded by an automated tester, and will be subject to code review. The final submission is due a week later. You will be expected to fix any failed test cases and revise your code based on code review feedback.

The final submission must address all the code reviews received by the beta submission – all the human comments plus all the automated checkstyle comments that are marked #important. You can address a code review either by changing the code to reflect the review or by including a source code comment in your code that explains why the code wasn't changed. If code reviews are unclear, you can discuss them with the reviewers, but you still must edit your own code in reaction to the review. A grader will check the submission and deduct points if it hasn't addressed the code reviews.

Since the final submission inevitably happens after code review for the problem set, it's understood that you've looked at other students' written solutions, and been inspired by other ways to solve the problems. You must be exceedingly scrupulous, therefore, in not using those written solutions during your revision. Both your original code and your revised code must be your own. Looking at other students' answers to the problem set while you are revising your solution will be considered a violation of the collaboration policy.

Collaboration and Public Sharing

In line with MIT's policy on academic honesty, please review the detailed page on collaboration and public sharing.

FAQ

Why is this course structured the way it is?

Practice and feedback are key to learning, and the course is structured to provide as many opportunities for practice and feedback as possible. That means we don't want to spend class time on lectures, we want to spend it on exercises to practice the concepts and skills of building software.

Citations: Wieman et al., *Course Transformation Guide* · Deslauriers et al., *Improved Learning in a Large-Enrollment Physics Class*

Why is attendance in class required?

Class meetings are all about practice and feedback . The individual, pair, and small group questions, exercises, and coding problems we work on in class, and the discussions and feedback led by instructors, are a required component of this class.

Class is like swim/judo/math team practice: you don't get good unless you show up, and practicing with others is a necessary complement to practicing alone.

Why are we sometimes asked to close laptops during class?

Your laptop is a necessary tool for in-class programming, but it also presents a huge opportunity for distraction. The price of that distraction is paid not only by you, but by all those around you who can see your screen. In one study linked below:

- For note-takers with laptops, multi-tasking led to a 11% drop in comprehension test scores.
- For note-takers without laptops, merely having a laptop multi-tasker in their field of view led to a 17% drop in comprehension test scores!

If you want to use a smartphone in your lap, so that the screen is not visible and not distracting to others, we have no objection, but you'll still be hurting yourself.

Citations: Sana et al., *Laptop multitasking hinders classroom learning for both users and nearby peers* · Mueller and Oppenheimer, *The Pen Is Mightier Than the Keyboard*

How is participation in class graded?

In general, participation in class is graded based on whether you attempted the questions, exercises, coding problems, etc., not on correctness.

Programming exercises are graded based on whether you've attempted the exercise and made some progress, even if you don't complete it. You are not expected to complete unfinished exercises after class, but TAs and LAs will be happy to help you review or finish them.

Why do I need to read the readings and complete the reading exercises the night before coming to class?

Reading the material before class prepares you to spend class time practicing the concepts and skills you're learning. Reading in advance gives you time to think and ask questions, and repeated exposure to material spaced out over time improves learning.

See: Spacing effect

Why does the class have nanoquizzes on topics before we practice them in class?

Nanoquizzes assess whether you did the reading and practiced with the reading exercises before coming to class, and they provide feedback to you on your comprehension. Nanoquizzes are themselves part of the practice we do in class: recalling information from the readings benefits learning more than just re-reading or re-hearing it.

See: Testing effect