

**Course:** CS 2110 —also ENGRD 2210 (engineers should sign up for ENGRD 2210)

**Course website:**

[www.cs.cornell.edu/courses/cs2110/2018fa/index.html](http://www.cs.cornell.edu/courses/cs2110/2018fa/index.html)

**Instructors:** Anne Bracy, David Gries

**CS 2111:** A 1-credit S/U course for students who want more contact hours. Only for students in CS2110. It gives more explanation of core ideas in Java, programming, data structures, assignments, etc., and provides more opportunities to ask questions. Requirement: Attend one 1-hour session each week. We recommend it!

**CS 2112:** An honors version of 2110, given at the same time. Don't take it if you don't know Java.

**Academic Excellence Workshop (AEW):** A 1-credit S/U course for students in CS2110 in which students work together in a cooperative 2-hour session each week. Requirement: Attend weekly 2-hour session. For info, visit Olin 167.

**CS 2110 Description:** Intermediate programming in a high-level language and introduction to computer science. Topics include program structure and organization, object-oriented programming (classes, objects, types, sub-typing), graphical user interfaces, algorithm analysis (asymptotic complexity, big “O” notation), recursion, data structures (lists, trees, stacks, queues, heaps, search trees, hash tables, graphs), graph algorithms. Java is the principal programming language.

**Course outcomes:**

- 1: Be fluent in the use of recursion and object-oriented programming concepts (e.g. classes, objects, inheritance, and interfaces).
- 2: Be able to design and implement parts of nontrivial Java programs (roughly 1000 lines of code), starting from an English language specification.
- 3: Understand graphical user interfaces.
- 4: Understand asymptotic complexity of algorithms and be able to analyze programs to determine their running times.
- 5: Understand basic data structures taught in the course and be able to implement them and use them in programs.

**Prerequisite:** CS 1110 or CS 1112 or equivalent course that provides a solid introduction to a procedural programming language. See the back of this page. *Knowledge of Java or OO programming is not a prerequisite.*

**Lectures:** Tues/Thurs 10:10–11:00, Statler Auditorium.

**Sections:** Sign up for one 1-hour section. Attend every week. Sections teach new material, review material, give help on assignments, etc. Permission not needed to switch sections, but register for the one you attend regularly. Some sections given at the same time are unbal-

anced with regard to enrollment. Please move to a different section to balance the load, if you can.

**Electronics Policy:** Cell phone usage is prohibited during lecture. Please limit laptop use to 2110-relevant notetaking. Research shows that laptop use hinders classroom learning for both users and nearby peers.

**Coursework and grades:** For an extensive discussion of grades, visit the course website and click on “Course info” in the left part of the top horizontal navigation bar.

**Programming Assignments:** The 9 programming assignments (the first, A0, is trivial) come in two flavors:

- Vanilla: specific task to learn and practice what's being taught. We tell you exactly what to do.
- Chocolate: more open-ended project, in which you get to do design, etc.

For the chocolate assignments, we will leave varying parts of the design to you—it's more fun but more challenging because you get to make design decisions.

**Course CMS:** The system used to submit assignments, record grades, etc. Its URL is on course website. It is populated with enrolled students and gets updated every 2-3 days.

**Piazza:** The website we use to communicate questions, answers, etc. Use it almost daily, both asking questions and answering questions that other students ask.  
<https://piazza.com/cornell/fall2018/cs2110/home>

**Lunch with instructors:** Each instructor will generally eat lunch once a week with up to 7 students in the Straight Oakenshield. Sign up beforehand on a doodle poll linked to on a Pinned Piazza note.

**Exams:** The two prelims and the *optional* final are given at the times shown below. Put them on your calendar *now*. Makeups are generally not allowed except in really exceptional circumstances. If you are out of town, arrangements may be made for you to take it while out of town. For more info, including about the optional final, visit the course website, click on *Course Info*, and click on *Policies*.

Prelim 1: Thu 27 Sep. 7:30-9:00PM or 5:30-7:00PM  
Prelim 2: Thu, 15 Nov. 7:30-9:00PM or 5:30-7:00PM  
Optional Final: TBA. 7:00-9:30PM

**JavaHyperText:** You don't have to buy a book! Instead we have extensive online material. Use this online resource regularly to find quick answers to questions. We believe it is far better than the paper textbooks.  
<http://www.cs.cornell.edu/courses/JavaAndDS/>

**Lecture slides:** We generally make slides available on the course website (click *Lecture Notes*) the day before a lecture. Please download them the day before the lec-

ture, scan through them to get an idea what the lecture is about, and bring a copy to class (paper, on laptop, on tablet, whatever).

**VideoNote:** A year ago, our lectures were videotaped, indexed, and placed on the web. We'll discuss in class whether these will be available and how.

**Java:** You need Java 8 (also called 1.8). If necessary, download and install it using instructions on the "Resources" page of the course website.

**Eclipse:** A free IDE (Integrated Development Environment) that we use to write, debug, and run programs. The "Resources" page on the course website has instructions for downloading and installing it.

**DrJava:** A free, extremely basic IDE, which we sometimes use to demo things easily. Use it to try things out, but don't use it for assignments. Download from [www.drjava.org](http://www.drjava.org). Download a "jar file", not the app.

**codingbat.com:** A website where you can practice writing small Java segment and see results immediately. Practice with boolean expressions, strings, arrays, recursion, etc. See the course website on Resources page.

**Academic Integrity:** On any programming assignment, it is a violation of academic integrity to:

1. Look at or be in possession of the code of another CMS group (most assignments can be done in groups of two), in this semester or a previous one with a similar assignment.
2. Show or give your code to another student.
3. Post code on the Piazza.

Naturally, you may discuss assignments with others, but the discussion should not extend to writing actual code, picking variable names, agreeing on specifications or comments, etc.

If you do an assignment with another person (in a Group), you must both sit at the computer together, working together. It is a violation of the code to split the work into parts, do the parts independently, and then merge.

### Programming Proficiency

A CS 2110 student should be able to write simple programs in some programming language. They should have fluency with strings, arrays, two-dimensional arrays, simple conditional tests, loops, and functions and procedures as well as calls on them.

You may find it helpful to watch our three videos on "stepwise refinement", to see how we advocate developing programs. From the course website, click on "JavaHyperText" in the top horizontal navigation bar. On the page that opens, click on "Stepwise refinement" in the horizontal navigation bar.

Here are examples of the sorts of programs you should be able to write, in whatever language you prefer. To test yourself, consider launching the editor you used when you learned to program and actually write, test, and debug them.

1. Write a function that returns true if its string parameter is a palindrome (false otherwise). A palindrome is a string that reads the same backward and forward, e.g. "Madam, I'm Adam." Actually, this string would fail the test because it contains white space and punctuation. With parameter "madamimadam", the function would return true.

2. Write a function that returns its string parameter but with punctuation and spaces removed and letters turned into lower case. If you call your function from problem 1 with the output of this new function, "Madam, I'm Adam." would pass the test.

Ideally, use some existing string function in the language you are familiar with to test for white space and punctuation and to map upper case to lower. No need to reinvent the wheel.

In CS 2110 we prefer to use the provided language features, including prebuilt library methods, to full effect. Better programmers are effective in using the tools available to them: they write less code, and their code is more expressive and more exact, so they make fewer mistakes.

3. Compute the median of a one-dimensional array  $x$  containing integers, or count the number of zeros in  $x$  (each of these actions would be a separate method, returning an integer value). Compute the mean as a floating point number.

4. Given integers  $b$  and  $c$ , where  $0 < c$ , compute  $b/c$  as an integer (rounded to the nearest integer).

5. Count the number of zeroes in a rectangular matrix  $y$ . For a square array  $\text{square}$ , determine whether all the diagonal elements have the same value.

6. Define the "balance" of a rectangular matrix  $y$  to be the number of elements larger than the mean value (rounded to an integer using the method of question 4) minus the number of elements smaller than the mean. Given an integer matrix, compute its mean and balance.

7. (Binary search). Given a sorted integer array segment  $b[h..k]$  and an integer  $x$ , find the position  $j$  such that  $b[h..j-1] \leq x$  and  $b[j..k] > x$ . ( $b[h..j-1] \leq x$  means that all values of  $b[h..j-1]$  are  $\leq x$ ). Your program should run in time proportional to the log of  $k+1-h$ . (Did you have binary search in your previous course? If so, this should be easy. If not, don't worry; we will teach it to you.)