Introduction to Computer Science II
Fall 2024
Course Information

Prof. Kaplan
Last updated: 2024-Aug-20

Be sure to read **all** of this document!

1 The topics

1.1 Is this course for you?

I will assume that students in this clear already know the following basics:

- data types and variables
- arithmetic and logic operations
- conditionals (*if-then[-else]*) statements
- arrays (single- and multi-dimensional)
- iteration (loops)
- writing and calling methods/procedures/functions

I assume that you've used the above in a few combined structures and patterns (e.g., *nested loops* and basic *recursion*). If you are familiar with this material—by having taken *Intro I* or some equivalent course, or through your own self-taught programming experience—then you are ready for this course.

1.2 The focus of this course

Every computer program is a composition of the basic building blocks listed above. However, in solving any kind of interesting problem, constructing programs with these building blocks raises issues of scale, structure, and clarity. That is, to solve larger problems, there must be *more*: more algorithmic patterns; more programming structures; more data types and data structures. If you are like nearly any normal human, you began to find that your own code, data, and solutions became confusing to you—a big pile of small values and operations that was hard to track in your head.

Moreover, although the programs we use—web browsers, word processors, spreadsheets, movie streaming services, GPS-based driving maps—are made of these same programmatic

structures, it is not obvious just how they are combined to make those kinds of programs work. The small class assignments that you solved in first learning to program seem to bear little resemblance to the full-sized programs that you routinely use.

This course covers *higher-level abstractions* that programmers use to organize code and data on a larger, deeper scale. Learning these abstractions and structures will allow you decompose and solve more complex problems, use existing code that solves small-scale problems, and approach a much wider range of algorithmic problems and solutions.

More concretely, here is a list of topics that this course will cover:

- Java review
- Defining and creating new object classes
- Inheritance and the class hierarchy
- Overloading and overriding
- Abstract classes and methods
- Interfaces and generics
- Exceptions and error handling
- File input and output
- Linked lists
- Stacks and queues

This course will be project-intensive. Much of the material will seem easy enough to comprehend when presented in class, but the only way to understand this material thoroughly is to use it. That is, to truly understand a concept in depth, you must formulate an algorithm to solve a given problem and then write that algorithm in a programming language. In this manner, our projects will require you to address these concepts in detail.

2 Lectures, labs, and help

Lectures and labs: The **lectures** for this class are on **Wednesday and Friday** of each week, from **1:00 pm to 1:50 pm** in **SCCE A011**.

The **labs** occur on **Monday** of each week in **SCCE A331**, at the following times (by section):

- **COSC-112L-01:** 12:00 pm to 12:50 pm
- **COSC-112L-02:** 1:00 pm to 1:50 pm

You are expected to be present for **all of the lectures and labs**; missing either is strongly discouraged. I will not teach material twice, so if you miss a lecture or a lab, then you're on your own. If you must miss lecture or lab due to an illness, a curricular conflict (e.g., a Geology field trip), or an emergency situation, contact me. **If you have a extra-curricular conflict** with a lecture or lab—for an athletic event, for a (non-curricular) musical or theatrical performance, to depart early for or arrive late from a vacation, or for any other non-emergency—then **the choice is yours to miss or to attend**. If you choose to miss the class meeting, then you must be prepared to obtain and learn on your own the material that you missed. I recommend that you choose to attend the class meeting when these conflicts arise. Do not underestimate the willingness of those who run extra-curricular programs to make accommodations for your academic priorities.

Office hours and meetings: If you seek assistance, reinforcement, review, or other opportunities to discuss the course material or assignments, you should see me. There is a link on the course web page for scheduling times for meetings. Please use it; talking with me outside of class is one of the reasons you came to a small college.

Announcements and questions: For communication outside of classes and labs, we will use `Slack`. You should download and install it. There will be a class channel, `#cosc-112-01-24f` where I will post class announcements, updates about assignments. That channel will also be the place for you to post questions that others may wish to see, and to get those questions answered by the TAs and by myself. Finally, via direct messaging, Slack will be the quickest way to send me a question and get an answer.

There is, of course, also a class website: `bit.ly/cosc-112-01-24f`. It will also have announcements, as well as all the of the assignments and documents needed for the class.

Help sessions and tutors: If you are struggling with some aspect of the class, your first line of defense should be to **see me** (see above). However, spending additional time with a peer tutor to go over the material and assignments can be a valuable aid.[1]

[1]Be cautious about project work with tutors! It is critical that they provide only conceptual feedback or hints about structuring or debugging your code. To see actual code written by the peer tutor to solve a part (or all) of a project is *plagiarism*.

Additionally, each lab section will have two *student teaching assistants (TAs)* to help answer questions and provide guidance in the labs. There will also be weekly *evening help sessions* led by TAs. More information about these will be forthcoming.

3 Texts and materials

This course will use the textbook *Building Java Programs: A Back To Basics Approach* by Reges and Stepp, edition 5e. This textbook will be provided through the college's textbook program. We will not follow the book explicitly, but use it as reinforcement of the ideas presented in class and practiced during labs and projects.

All other tools for this course—all of the software and documentation—will be provided. The first lab will include instructions on how to get the necessary coding tools installed and configured on your computer.

4 Assignments, deadlines, and extensions

There will be a number of programming assignments. The deadline for each will be stated clearly on the assignment, as will the manner of submission. **Late submissions may receive failing grades**. Turn in what you have, and do so on time.

An extension for any assignment **must be requested in writing** (email counts as *writing*), **at least 48 hours prior to the deadline**. The determination as to whether or not a particular situation merits an extension will be made on a case-by-case basis. Scheduled events and moments of heavy curriculur workload are **not** sufficient reason to warrant an extension. Rather, extensions are intended for unusual circumstances that prevent you from planning your time well in order to meet the deadline. Note that a sudden onset of illness or other emergency situation that occurs less than 48 hours before a deadline will be treated as a special case.

Note also that extensions will be short. This course will present a constant stream of assignments, and falling behind would create an intractable situation from which students rarely catch up. Therefore, extensions will be only so that you can recover from a temporary difficulty and submit *something*. If you fall behind, I am likely to direct you to skip the missed assignment and instead focus on the current one. Keeping things moving will be critical.

5 Exams

There will be **one mid-term exam** given during a regular class lecture hour; there will also be a **comprehensive final exam** given during the final exam period at the semester's end. The mid-term exam will be given during week 7 of the semester; the final exam will be a 3-hour, scheduled exam. Its time and location will be announced when the Registrar's office posts the final exam schedule, usually a few weeks into the semester.

6 Grading

I do not use a fixed formula to determine grades. Instead, I will offer the following guidance on how your work will be evaluated:

- **Labs** are small programming assignments and are considered *practice*. You are expected to complete and submit them all. Missing labs can have a negative effect on your final grade. The feedback on these labs will simply indicate whether there is a problem with the work to which you should bring your attention.
- **Projects** are larger programming assignments and consitute both deeper practice with the concepts **and** a chance for me to evaluate your understanding. These will be graded more thoroughly and given a grade that reflects how well you seem to have understood and solved the problem. Their contribution to your final grade is still quite modest, but the quality of the work will matter.
- **Exams** are hand-written, in-person, time-limited evaluations and will be the primary determinant of your final grade. Note that the final grade should reflect your understanding and mastery of the material at the *end* of the semester. Thus, the final exam is the most significant component in your final grade.

7 Academic dishonesty

You will be expected to do **your own work** on all assignments and exams in this course. While I encourage you to interact with your classmates and discuss the material and assignments, there is a limit to the specificity of such discussions. We seek to make that limit clear here.

It is acceptable to discuss any assignment for the class with a classmate. You may even discuss your approach to a particular problem, or review relevant material for a problem with another person. However, you **may not show another student your code, nor see another student's code. If in doubt, contact me and ask for clarification**.

Copying and submitting code from another student, a web site, or an AI system (e.g., ChatGPT) is misrepresenting that code as your own work, and is therefore plagiarism. Plagiarized submissions will yield a failing grade for the course.

Regarding the use of AI's or other internet sources: Don't. Remember that those assignments have a small effect on your grade; they are intended to be *practice*. If you take shortcuts with that practice, then your understanding and development will suffer, and the results will show on the exams. So don't. Just don't. You don't need it. Use the help sessions, the labs, the Slack channel, the office hours, and develop the skills.