Comp 127: OOP and Abstraction

Fall 2025

Instructor Paul Cantrell pcantrel@macalester.edu
he/him or they/them cantrell@pobox.com
Addressing me as "Paul" is fine Note: only **one L** in Macalester email!

Class

MWF 10:50 – 11:50 AM

Final project celebration

Mon, Dec 15, 10:30 AM – 12:30 PM

Contact / Help / Support options

- Use preceptor office hours (schedule on course site)
- Contact me online via Slack or email
- Self-schedule an office hours appointment

Course Description

What happens as programs grow large and complex? How do we break a large program into manageable pieces? How do we make those pieces reusable beyond their original context, and their original author? Programming is a deeply complex, collaborative, and human endeavor. How can we humans, with our human minds, work with computers and each other to create something of value?

This course is an introduction to the way software developers create structure in code. It provides **practical experience**: it is a heavily hands-on course, and you will spend time building the skills necessary to create software using many of the same tools that professionals use to build it.

However, at least as importantly, this course is a grounding in **underlying principles**. In the fast-changing world of software, no matter what you learn today, rest assured you will need to learn something unfamiliar tomorrow! This course will give you groundwork to recognize the familiar in the unfamiliar, and stay adaptable by anchoring your learning in things that have remained present through programming's many topsy-turvy years.

In this class, you will learn about subject matter in three general categories:

- **Concepts:** These are the ideas of which code and code structures are made. They take on many forms, appear in many contexts, and apply across many programming languages and kinds of software. Examples include: abstraction, classes and objects, encapsulation, type systems, ASTs, polymorphism, closures, APIs, separation of concerns, contracts, immutability, and parallelism.
- **Practices:** These are the processes and habits developers use to build software. They are the heart of the software practitioner's craft, and are not so much about what the code is as they are about what the programmer does. Examples include refactoring, testing, pairing, versioning, diagramming, writing for readability, documenting, and debugging. Collaboration and communication are recurring themes through all of these.
- **Tools:** We will work with many tools professional software developers use on projects out in the wild, most notably Java, Git, GitHub, Visual Studio Code, JUnit, and Slack. Each of these tools is practical and useful in its own right! However, each is also a path to deeper learning about concepts and practices. Thus, you should not think of this as a "Java course;" it is a course that uses Java to teach the important concepts and practices of software development.

You can find a detailed overview of course content in our **Comp 127 Learning Goals**.

A Word of Warning, and of Encouragement

Programming is fun! But programming can also be extraordinarily, crushingly frustrating. (Yes, for everyone, even your instructor!) Sooner or later you will get stuck. This never goes away; even the most experienced developers spend lots of time stuck and frustrated. The difference is that the experienced developer knows how to work through it, both technically and psychologically. Learning to deal with frustration is one of the most important parts of being an effective software developer.

The secret to learning? **Seek help early and often!** If you get stuck, yes, do wrestle with it on your own for a little while, but remember that the assignments are designed to make you need help. Look at the **How to get help** section on Moodle. It's at the top for a reason!

Talk with each other, share programming tips, and cheer each other on! (However, see notes below on collaboration vs. copying and what constitutes cheating.)

Coursework

Readings and Course Tasks 5%

There will be approximately 20 reading assignments and course tasks. Sometimes they will include questions for you to reflect on. Sometimes there will be a job you need to complete.

Readings are **due one hour before class.** To receive credit for the readings, **post questions or observations** about the readings in our Slack channel, #comp127-f25. For full reading credit, **post at least one question or thought about the readings each week**, at any time during the week. You do not need to post a question or thought in response to every single assigned reading, though I encourage you to ask as many questions about the reading content as you like. I also encourage you to read the questions other students asked, and the responses to them.

Among the course tasks I will ask you to complete is to **attend at least three MSCS department-sponsored events** during the semester. This can include department seminars, Beyond Mac, coffee breaks, and capstone presentations. Instructions for receiving credit are on Moodle.

In-class activities and take-home exercises 15%

We will spend much of our in-class time doing practical programming activities. No matter how many concepts you understand, you can only learn programming by doing it.

The activities are for practice, what matters with these is **effort**. Any reasonable attempt at most of the activity counts. These are much like practicing a musical instrument, or doing a daily workout: simply putting in the time and energy is the main point. We will use pair programming for all activities, because communication with humans is a *crucial* part of software development.

There are about 6 take-home exercises in the course. These are more in-depth solo programming efforts. I grade them for **completion, not correctness**: reasonable effort receives full credit, even if there are some problems with your solution. You should complete each take-home exercise **within one week** of when the exercise was originally assigned.

Homework 40%

There will be approximately 6 homework assignments in the course. We (preceptors and instructor) grade these for **correctness and quality**. With the exception of the first homework, (which is toy-sized, just for everyone to practice the process), the homeworks involve larger, more complex coding tasks than the activities and take-home exercises. Larger homework assignments will be worth more points. Preceptors help grade the homework. Because of the labor involved in homework grading, it is especially important to respect the homework due dates. If you will need extra time for an assignment, talk to your instructor **in advance** (see "Late work" below).

Conceptual mastery puzzles 15%

You are not only learning to build software, but also learning to *think* about software. For some aspects of software development, it is good to treat the computer as a partner and let it assist you. However, there are some fundamental concepts you should be comfortable reasoning about *in your own head*, without leaning on the computer for assistance.

To help you build this conceptual mastery, you will do puzzles of different types, each type associated with a specific conceptual mastery goal. We will give you software that can generate an infinite stream of puzzles (and solutions) of each type. You can practice these puzzles as many times as you want, whenever you want.

When you feel you are ready, you can attempt one of these puzzle types for credit. When you make this official attempt, you can use notes but *not* your development tools. Remember, no leaning on the computer for assistance! I will grade your attempt for **precise correctness**. You can receive full credit, half credit, or no credit.

Here's the catch: **you can attempt each goal repeatedly**. Your final score for each conceptual mastery goal is the best score your received. This means that you have many chances to do well in each goal. If you do badly on the first try, don't panic! That bad attempt doesn't permanently bring down your final score. It just means you need to try again.

Course Project 25%

At the end of the class, you will create a programming project of your choice with a partner or two. This is your chance to stretch your legs, apply what you have learned, be creative, and have fun building some software.

Policies

Copying code

Programming is a social activity, and I encourage you to seek out help from your instructor, your preceptors, your fellow students, and the web. Collaboration is good; however, cheating is not. **Copying any code you did not write without attribution** is cheating, and under many circumstances, copying code is a federal crime. (This is true even if you modify what you copy.)

It can be difficult to understand the line between copying and assisting, so I'll give you the following guidance:

- Do not share your code for take-home exercise and homeworks with other students. (Sharing and comparing code for in-class activities is OK.)
- Do not look at somebody else's source code for a homework solution (except for the one exception noted below).
- Do not communicate with each other during exams.

The one exception to these rules is that **you may help somebody debug their code** for a take-home exercise or a homework problem which you have already solved yourself.

If someone helped you with a problem, acknowledge their help on the assignment with a comment in the code. I will **never dock points for acknowledging the help of others**, and crediting help you received helps mitigate any suspicion of cheating. So that I have a sense of how many students are reading this syllabus, please email me a picture of a hydrozoan.

If you copy code from a public reference (e.g. Stack Overflow, blog, tutorial, docs), you must: make sure that you **have the legal right** to copy it, and **add attribution** the code's origin. (In most cases, it is sufficient to add a comment in the code with the URL of your source.)

The act of turning in copied code without attribution is a violation of Macalester's academic honesty policy, and is subject to disciplinary action according to college rules. The disciplinary process is not fun for either of us. Please don't.

Using AI / LLMs

In *this* class, **using any form of AI-generated code** on in-class activities, take-home exercises, homeworks, or conceptual mastery puzzles **also counts as cheating**. This includes but is not limited to code generated by GPT and Copilot. Don't.

Why? The exercises and homeworks in this class ask you to puzzle out for yourself things that many, many other people have puzzled out before. The learning comes from **you doing the puzzling**, not from you producing correct output.

AI can easily regurgitate other people's solutions to familiar problems; for familiar questions, it often ends up functioning as a plagiarism tool. If AI does that for you in this class, it robs you of your learning. The activities, take-home exercises, and homeworks in this class are a bit like lifting weights: other people have done it many times before, and it's easy to make a machine do it; it is *you* doing it that matters. If you had a machine lift weights for you, would you expect to build muscle?

You *are* allowed to try out AI-generated code for your course project, although I don't particularly recommend it. I will provide more guidance when assigning the projects.

Late work

You will hand in all your homework online. If technical troubles are preventing you from handing in your assignment, contact me *immediately*, even if it is the middle of the night! I will respond when I am able.

You may also have non-technical troubles preventing you from handing in an assignment. We all have lives, and sometimes things come up. I understand. If circumstances will force you to do work late, **talk to me before the assignment is due**. My goal is to make this course the best it can be for all of us, students and instructor, within the constraints that we have. That means I will happily do what I can to accommodate you if you are facing a difficult situation or a time crunch, but that can only work smoothly if you talk to me in advance. Late work impacts the preceptors, your instructor (me), and your own learning. Late work thus requires planning. Think ahead! Communicate!

If you have not been in communication with me about late work, then assignments incur a late penalty in proportion to the time past due, with credit **continuously decreasing to zero** over the course of one week. This late policy means that it is in your interest to hurry up and finish your assignment even if it is already late. Don't delay! Don't feel guilty! Just get it done! Note that this policy means that if you are going to miss the deadline by a few hours, it is far better to finish your work properly than to hand it in half-done.

Questions about grades

Preceptors typically grade exercises and homework assignments via Github Classroom using Pull Requests (PRs). You may receive notifications of comments during the grading process. It is critical for preceptors to finish their grading process uninterrupted. We ask that you **wait until the whole assignment is graded** before asking any questions regarding grading. Once you have complete feedback on the whole assignment, you are welcome to contact the **instructor** for clarification. **Do not contact preceptors about grading questions.** Do not contact the instructor through multiple redundant channels.

Classroom Inclusion and Community Guidelines

Macalester College values diversity and inclusion. We are committed to a climate of mutual respect, free of discrimination based on race, ethnicity, gender identity, religion, sexual orientation, disability, and other identities, in and out of the classroom. This class strives to be a learning environment that is usable, equitable, inclusive, and welcoming.

To help support these goals, we expect you to follow the MSCS Community Guidelines. These guidelines were created by the MSCS faculty and staff in our ongoing efforts to create a community that is more welcoming, supportive, and inclusive. They describe our expectations and how to respond to or report issues that arise. If you anticipate or experience any barriers to learning, please talk to your instructor, or see the guidelines document for how to raise your concerns.

Support and Well-Being

All of us — you, me, your other professors, your fellow students — are human beings. All of us carry our own experiences, thoughts, emotions, needs, and hopes with us, including in class. All of us have bodies and minds that need care. Please do what you need to do to take good care of yourself, and make our class a good experience for yourself and your fellow students. In the classroom, eat if you are hungry, drink if you are thirsty, and step out if you need a break. Outside of the classroom, take the time to sleep well, eat well, and connect with others — even when life is busy and the pressure is high. If you are feeling sick, stay home. We keep each other safe! And when you need help, reach out.

I put the paragraph above in my syllabus every term. It is especially important right now, during a time of multiple ongoing crises in the larger world. We urgently need to be present for each other, and present for ourselves. Our physical and mental health depends on *all* of us taking care of ourselves and each other.

I am committed to providing assistance to help you succeed in this course. I love programming, and I hope you will love it too. I am living through these difficult times preserving my spirit as best I can, and I want to help you do the same.

If **anything** poses a risk of interfering with your ability to thrive in this course or on this campus, please discuss it with me. This includes health issues (both mental and physical), disabilities (both documented and undocumented), schedule conflicts, competing obligations, life changes, and personal challenges. We will talk about what you need, and form a strategy for this course together. You don't need to share any more personal detail than you are comfortable sharing. Just **let me know that you need support**, and I will do my best to support you.

Official accommodations are available for students with documented disabilities. Contact disabilityservices@macalester.edu to make an appointment, and visit Macalester's Disability Services site for more information on the accommodations process. Note that **even if you do have an official accommodation, it is still your responsibility to contact me**.

Please communicate with me. No matter whether you have a sticky personal situation, a vague concern, or a struggle that is impacting you and your work, no matter whether it is officially recognized by the college or not, please communicate with me. I will try not to ask for details you are not comfortable sharing, but I do want the chance to help you out. The golden rule:

Never suffer in silence!

Never. I am here to support you. I know it can be hard to ask for support when you most need it. When that time comes, remember that **I *want* you to ask.**