Welcome

CSCI 245
Wheaton College
Thomas VanDrunen
Spring 2026

Please keep phones and other devices silenced and put away—not seen or heard (or used) any time during class.

- Object-oriented programming in Java. (CSCI 235)
  - Review: classes, interfaces, polymorphism, and other basic OOP features.
  - Review: Java Collections.
  - Class extension (subclassing, inheritance).
  - Nested classes.
  - Generics.
  - Enum types.
- Software development. (CSCI 335)
  - Software lifecycle.
  - Documentation.
  - IDEs.
  - Revision control.
- Object-oriented design. (CSCI 335)
  - Elements of good software design.
  - UML.
  - Refactoring.
  - Design patterns
- Analysis of algorithms. (CSCI 345)
  - Loop invariants.
  - Analyzing an iterative algorithm.
  - Big-oh notation.
  - Analyzing recursive algorithms.
  - Empirical measurements of performance.
  - Sorting.
- Abstract data types and data structures. (CSCI 345)
  - Linked lists and other linked structures.
  - Abstract vs concrete types.
  - Stacks.
  - Queues.
  - Binary trees.
  - Hashing.
- C programming. (CSCI 351)
  - C basics.
  - Functions and prototypes.
  - Compiling, linking, preprocessing.
  - Structs.
  - Pointers and dynamic allocation.
  - C-style Strings.
  - Bit operations.
  - Function pointers.
- Computer systems. (CSCI 351)
  - Model of computer memory.
  - Model of execution.
  - Programming in pseudo-assembly.
  - Model of function call and return.

Tier 0.
Recommended.

These are your go-to resources for learning and getting help.
The course textbooks; your notes; other class material (handouts, slides, etc); code from labs; the official Java and C documentation (see links on Canvas); your instructor (office hours, email); your TA (Daniel Kao); evening lab assistants.

Tier 1.
Use with caution.

There may be some, limited use of these, but prefer Tier 0.
A classmate's notes; classmates themselves; upperclass Wheaton CSCI majors and minors; tutors; unofficial Java references and guides; other textbooks that teach programming in Java or C.
See academic integrity policies in syllabus; do not share or look at working code for the projects; inform your instructor if you use a tutor.

Tier 2.
Not recommended.

Avoid using these.
StackOverflow and similar websites; the "AI overview" from Google and similar search tools; online tutorials; people outside of Wheaton College's CSCI program.
If you get any information from a Tier 2 resource, *verify that information with a Tier 0 resource*. If you find you receive crucial help from any such source on a project, then cite the source.

Tier 3.
Disallowed.

Do not use these under any circumstance.
Classmates' code; code from students who took this course in past semesters; AI-assisted tools including IDE plugins and LLM chatbots; anything that provides a solution to a project or similar problem.

Spirits of the Computer

We are about to study the idea of a *computational process*. Computational processes are abstract beings that inhabit computers. As they evolve, processes manipulate other abstract things called *data*. The evolution of a process is directed by a pattern of rules called a *program*. People create programs to direct processes. In effect, we conjure the spirits of the computer with our spells.

A computational process is indeed much like a sorcerer's idea of a spirit. It cannot be seen or touched. It is not composed of matter at all. However, it is very real. It can perform intellectual work. ...

Abelson and Sussman, *Structure and Interpretation of Computer Programs*, 1984 and 1996.

The Compulsive Programmer

The computer programmer... is a creator of universes for which he [or she] alone is the lawgiver. So, of course, is the designer of any game, ... [but computer] systems so formulated and elaborated *act out* their programmed scripts. They compliantly obey their laws and vividly exhibit their obedient behavior. No playwright, no stage director, no emperor, however powerful, has ever exercised such absolute authority to arrange a stage or a field of battle and to command such unswervingly dutiful actors or troops.

One would have to be astonished if Lord Acton's observation that power corrupts were not to apply in an environment in which omnipotence is so easily achievable. It does apply.

Joseph Weizenbaum, *Computer Power and Human Reason*, 1976

Reflection of God's Creativity

I think people who write programs do have a glimmer of extra insight into the nature of God... because creating a program often means that you have to create a small universe.

Don Knuth, *Things a Computer Scientist Rarely Talks about*, 2001

The Joys of the Craft

Why is programming fun? What delights may its practitioner expect as his reward?

First is the sheer joy of making things. As the child delights in his mud pie, so the adult enjoys building things, especially things of his [or her] own design. I think this delight must be an image of God's delight in making things, a delight shown in the distinctness and newness of each leaf and each snowflake.

Fred Brooks, *The Mythical Man-Month*, 1995

The Joys of the Craft (cont'd)

Second is the pleasure of making things that are useful to other people...

Third is the fascination of fashioning complex puzzle-like objects of interlocking moving parts and watching them work in subtle cycles, playing out the consequences of principles built in from the beginning...

Fourth is the joy of always learning, which springs from the non-repeating nature of the task...

Fred Brooks, *The Mythical Man-Month*, 1995

The Joys of the Craft (cont'd)

Finally, there is the delight of working in such a tractable medium. The programmer, like the poet, works only slightly removed from pure thought-stuff. He [or she] builds castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures. (As we shall see later, this very tractability has its own problems.)

Fred Brooks, *The Mythical Man-Month*, 1995

The Joys of the Craft (cont'd)

Yet the program construct, unlike the poet's words, is real in the sense that it moves and works, producing visible outputs separate from the construct itself. It prints results, draws pictures, produces sounds, moves arms. The magic of myth and legend has come true in our time. One types the correct incantation on a keyboard, and a display screen comes to life, showing things that never were or could be.

Programming is fun because it gratifies creative longings built deep within us...

Fred Brooks, *The Mythical Man-Month*, 1995

Coming up:

Read from McDowell, read ch 1, sec 2.1, sec 3.1, and sec 6.2.

Stay tuned for a prelab reading, quiz, and Project 0.