Welcome and Syllabus! CS 136 (Data Structures)

Sam McCauley

February 6, 2026

Hi!

- I'm Sam; you can call me Sam
- Office: TPL 304
- Email: `sam@cs.williams.edu`
- Not *quite* accessible; let me know if that's ever an issue

Course Materials and Tools

- Course website
  - `http://cs.williams.edu/~cs136/index.html`
  - Syllabus, schedules, office hours, *readings*, links to (virtually) all content
- Gitlab
  - `evolene.cs.williams.edu`
  - Used for git: starter code, helping you keep track of your work
- Gradescope
  - `gradescope.com`
  - Used for all grading/feedback

Syllabus

- Let’s look at the syllabus

- And the website

Honor Policy

- Academic dishonesty is often *overstated* in students' minds—most students are honest
- I'm expecting to have some honor code cases this year
- A "reverse lottery:" cheating gains you a little each time, but you risk losing a lot

FAQ: LLMs are extremely useful. Why can’t we use them?

LLMs for Coding

- LLMs are currently great for jobs that you COULD do, but would be a bit of a pain to actually do
- Goal of this class: get you to the point where you *can* code up these simple tasks yourself
- Best way to learn is by doing!

LLMs for Learning/Studying/Searching

- LLMs are good at this. I use them myself sometimes
- They are quick and easy, and often work OK
- But they do not always work, and it's not always easy to tell why
- In this class: use the resources we provide. It will help you in the long run!

Help Hours

- Start on *Monday*
- Posted on the website

Tools for Code

- VSCode
  - Powerful, useful, free, easy to install, often used in practice
  - *Occasionally* too much for our purposes.
- terminal
  - You'll be using to compile and run Java programs
  - Built in to VSCode! Works on Windows, Mac, etc.
  - Learning to use the terminal is a part of the course

Why Take CS136?

- To learn about:
  - Data Structures
    - Effective ways to store and manipulate data
  - Advanced Programming
    - Combine data structures, programming techniques, and algorithmic design to write programs that solve interesting and important problems
  - Basics of Algorithm Analysis
    - Measuring algorithm complexity
    - Establishing algorithm correctness

Goals

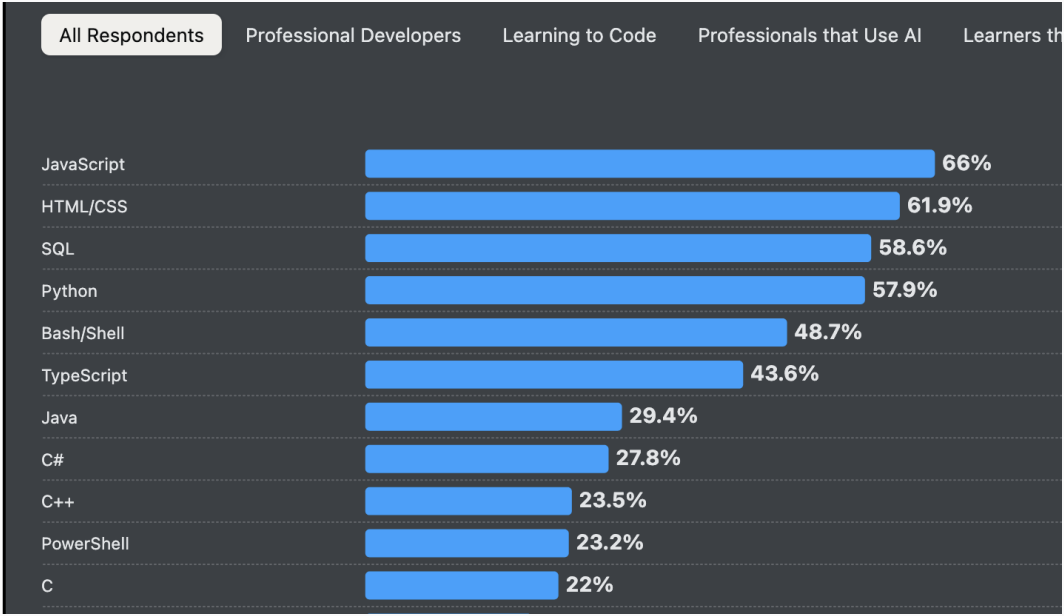
- Identify basic data structures
  - list, stack, array, tree, graph, hash table, and more
- Implement these structures in Java
- Learn how to evaluate and visualize data structures
  - Linked lists and arrays both represent lists of items
  - Different representations of data
  - Different algorithms for manipulating/accessing/storing data
- Learn how to design larger programs that are easier to modify, extend, and debug
- **Have fun!**

Course Outline

- Java review
- Basic structures
  - Lists, vectors, queues, stacks
- Advanced structures
  - Graphs, heaps, trees, dictionaries
- Foundations (throughout semester)
  - Vocabulary
  - Analysis tools
  - Recursion & Induction
  - Methodology

Why Java?

Java is Still Popular!



Why Java?

- Java is (still) popular!
- Object-oriented—good for large systems
- Good support for abstraction, extension, modularization
- Automatically handles low-level memory management
- Very portable

How to Think about Java

- Containers lead to scalability, flexibility, portability
- But they come with overhead!
- Java is the same way
- This class is all about *scalang* your CS knowledge

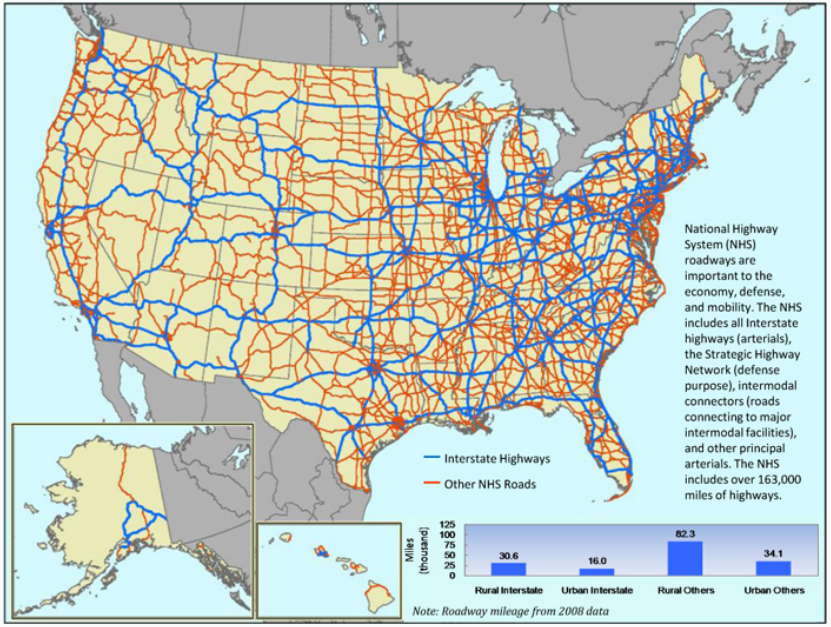
Java Experience

- It’s OK if you’re new to Java, or very rusty
- (I am also quite rusty)
- Let me know if you see any mistakes!
- Goal for this class: figure it out together

Common Themes in This Course

- Identify data for problem
- Identify questions to answer about data
- Design data structures and algorithms to answer questions *correctly* and *efficiently*
  - Note: not all correct solutions are efficient
  - And vice versa!
- Implement solutions that are robust, adaptable, and reusable
- Example: Shortest Paths in Networks

Example: Shortest paths



Finding Shortest Paths

- The data: road segments
  - Road segment: Source, destination, length (weight)
- The question
  - Given source and destination, compute the shortest path from source
- The algorithm: Dijkstra's Algorithm
- The data structures (spoiler alert!)
  - Graph: holds the road network in some useful form
  - Priority Queue: holds not-yet-inspected edges
  - Also uses: Lists, arrays, stacks, ...
- A demo....

Java Intro

Let’s take a look at Java programming

- Next couple lectures: (re)introduction to Java
- Some of you have Java experience; some don’t
- Goal: quickly get everyone up to speed for the first couple labs
- If you are less comfortable with Java, be sure to keep up with readings and ask questions early on!

Elements of a Java Program

```
1 public class Hello {
2       public static void main(String[] args) {
3
4
5
6
7       }
8 }
```

- First line: the “class” the program lives in (basically the box we’re working in). Must match the name of the file!
- Second line: the “main” function where execution starts

Hello World

```java
/*
 * Hello.java
 * Author: Sam
 * Prints a welcome message to the terminal
 */
public class Hello {
 public static void main(String[] args) {
     System.out.println("Hello, CS136!") ;
 }
}
```

Let’s test out the hello world program

- Open it in VSCode
- To *compile*:

```
> javac Hello.java
```

- To run:

```
> java Hello
```

Compiling and Running Code

- You need to compile the code every time you use it.
- Long story short: the computer does a pass over the code, translating it into a form that's more useful for it internally
- In pairs: What are some disadvantages of compiling code before it is run? What do you think some advantages are?

Playing around with Hello World

- How can we change what is printed?

- What happens if we make a mistake?

Notes on Java Syntax

- Multi-line comments: `/*` and `*/`
- Single-line comments: `//`
- Code is wrapped in a *class declaration*
  - Everything is (in) a class in Java
  - File name must be same as declared class name
  - System is a Java class holding an object called out

Semicolons

- Every command needs a semicolon at the end
- (Don’t need at the end of loops, function declarations, classes, etc.)
- It’s annoying. The compiler will tell you if you missed one