Introduction to CS 2110

Spring 2024


How long is the shortest route from LA to NY?
Calgary
Winnipeg
Montréal
Boston
Toronto
Duluth
Portland
Helena
Pittsburgh
Chicago
Omaha
Salt Lake City
Denver
Saint Louis
Kansas City
Raleigh
Nashville
San Francisco
Oklahoma City
Atlanta
Charleston
Santa Fe
Las Vegas
Little Rock
Los Angeles
Phoenix
Dallas
El Paso
Houston
New Orleans
Miami
USA

How long was the shortest route from LA to NY?

A. 19
B. 20
C. 21
D. 22
E. other

Outcome of CS 2110

- Write bigger, better software
  - Write programs with fewer bugs
  - Write programs that run faster and solve bigger problems
  - Leverage software that's already been written
  - Represent real-world data and problems in ways computers understand
- Read and write Java code

Demo: final assignment

Computer science is about sewers?

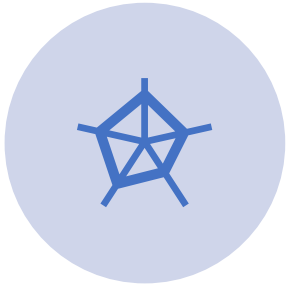
- Character navigates a maze
  - How to represent mazes?
  - How to generate solvable mazes?
  - How to navigate a maze efficiently?
- Program is graphical
  - How to respond to interactive events?
- Character seeks treasure, but has limited time
  - How to maximize reward within constraints?

Course themes

Programming languages and paradigms

Producing correct & maintainable software

Organizing information in memory



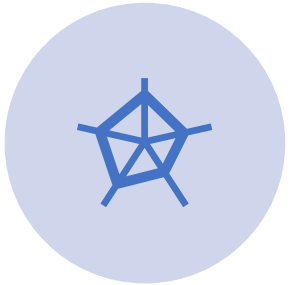
Comparing algorithm performance

Course themes

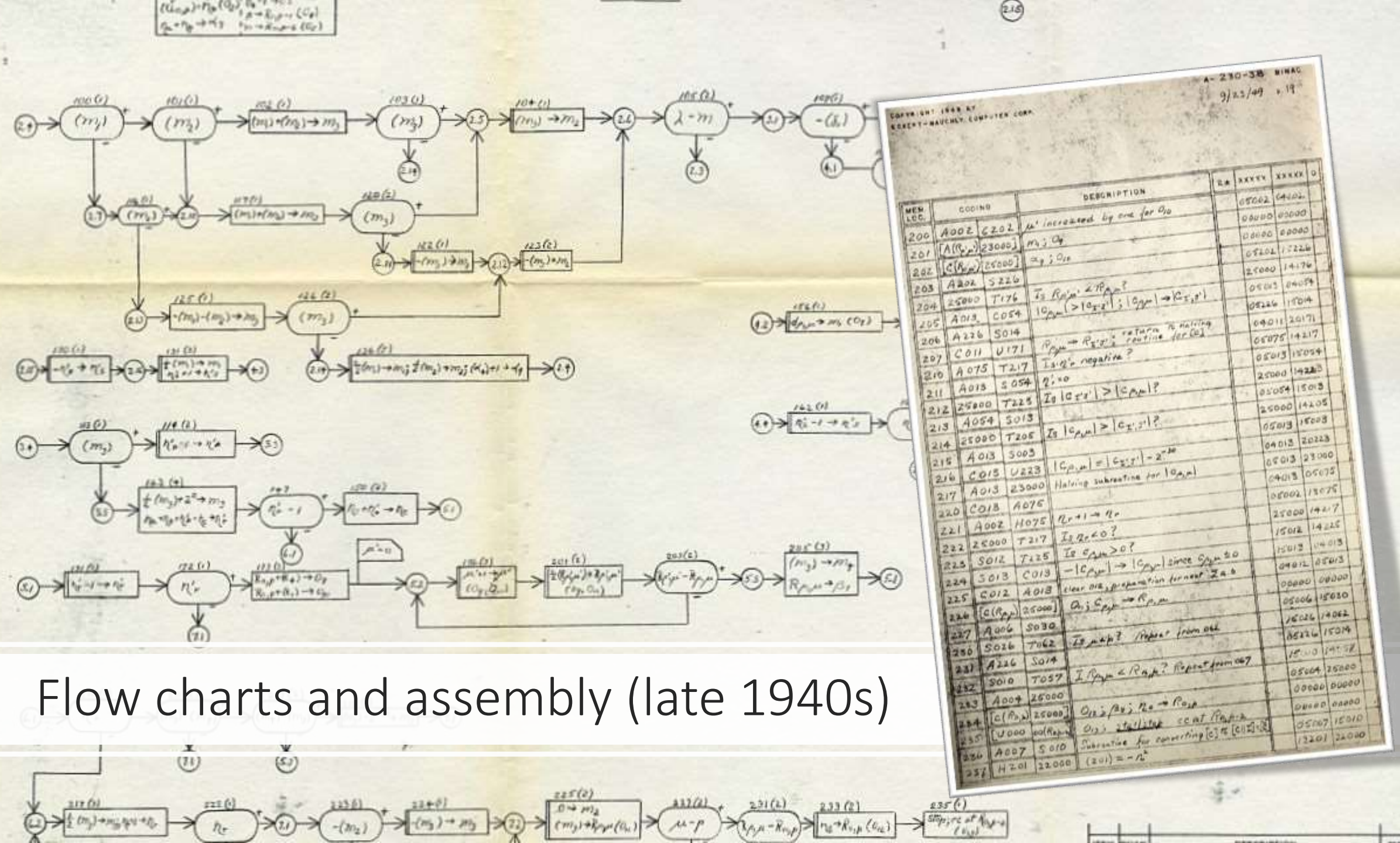
Programming languages and paradigms

Producing correct & maintainable software

Organizing information in memory



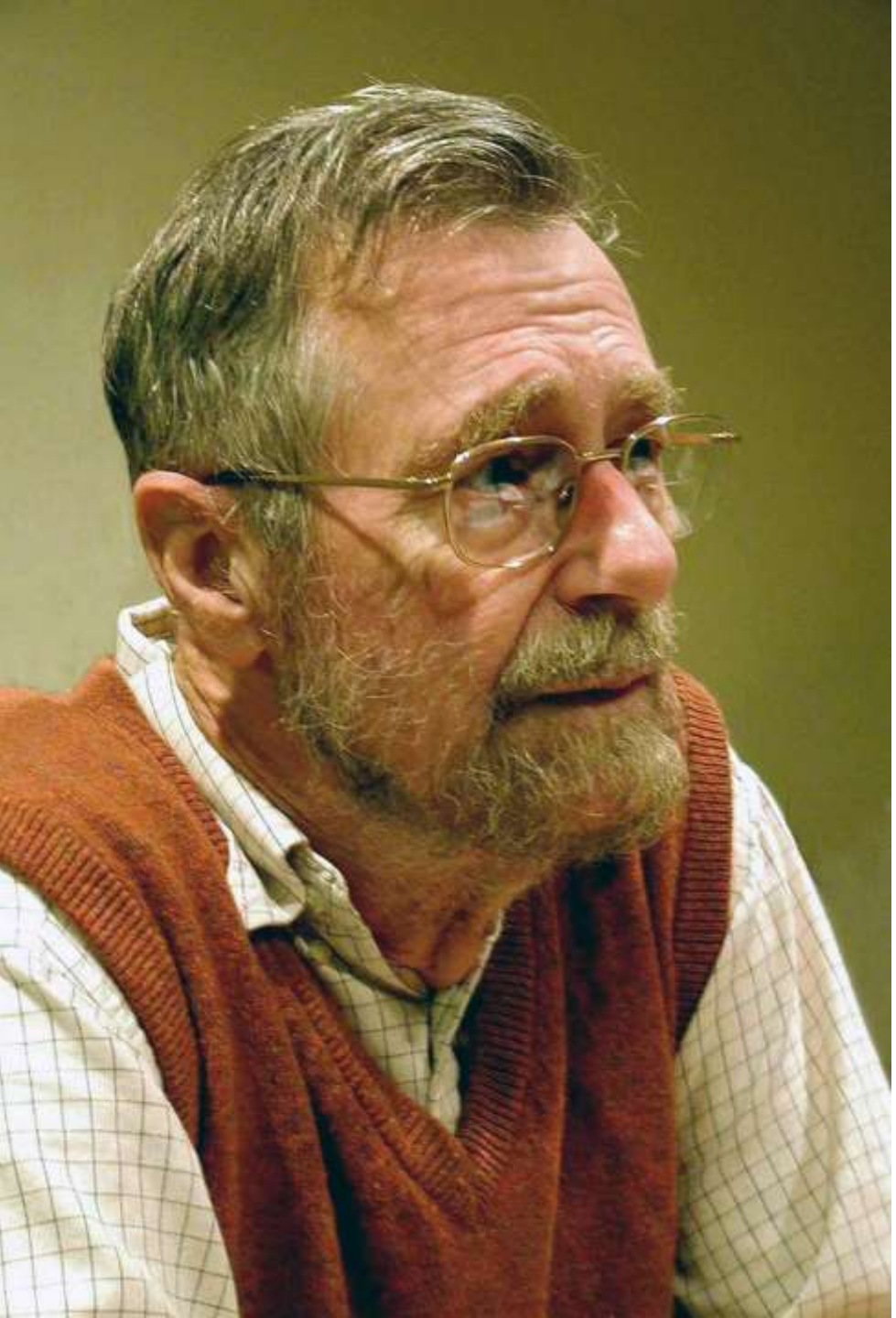
Comparing algorithm performance



Flow charts and assembly (late 1940s)

Compiled languages (1950s)

< Grace Hopper



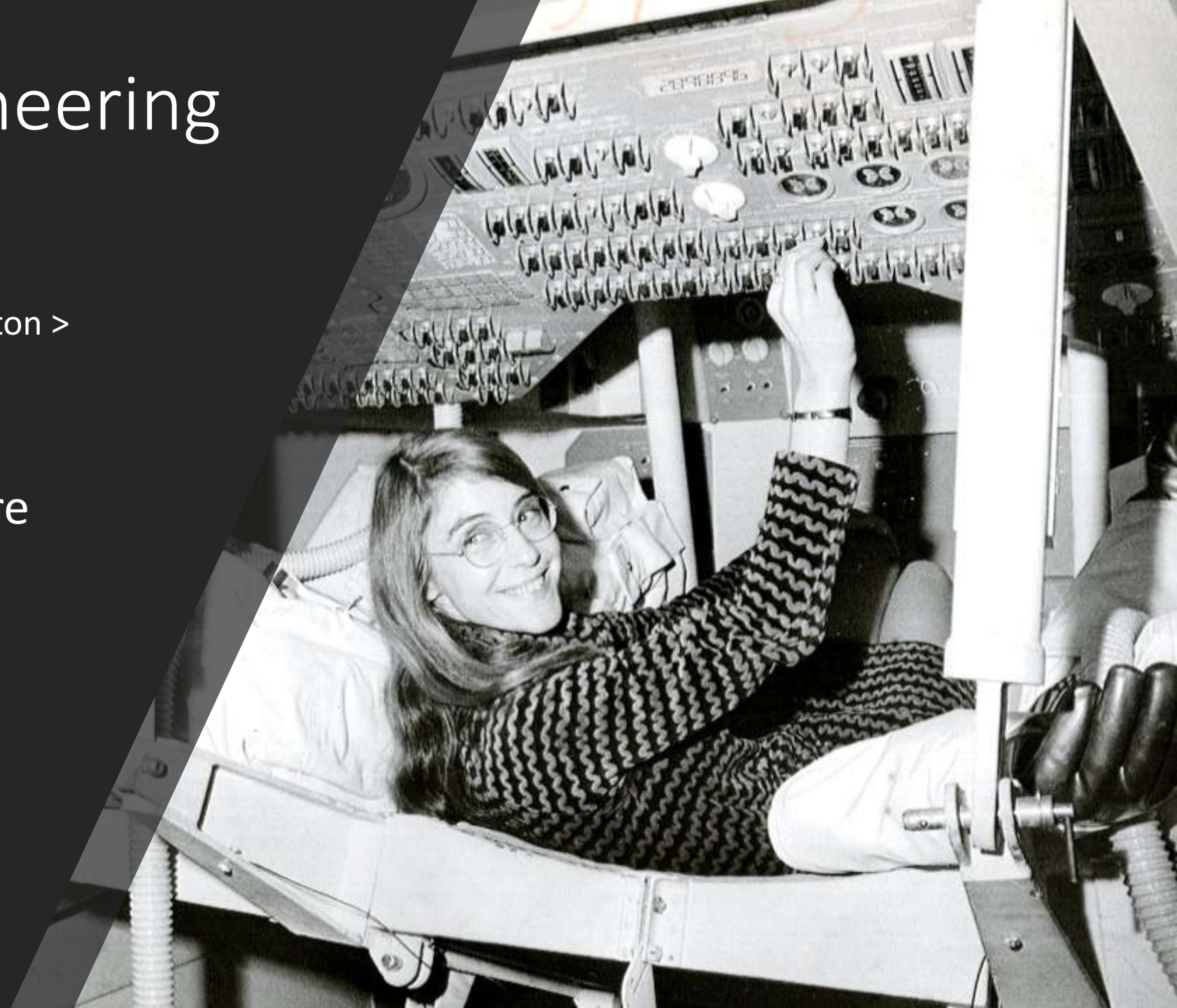
Structured programming (1960s)

- Flow chart design, GOTO leads to “spaghetti code”
- Standardized common patterns
  - Blocks (Python indentation, MATLAB `end`, Java `{}`)
  - Control structures: `if`/`else`, `while`
  - Subroutines: `def`/`function`
- Discourage more general techniques
  - Unnecessary – subroutines, conditionals, and loops can compute anything
  - 1968: Dijkstra's *Go To Statement Considered Harmful*

Software engineering (late 1960s)

Margaret Hamilton >

NATO and the “software crisis”



Object-oriented programming and Java

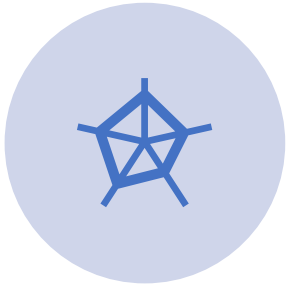
- OOP is the dominant contemporary paradigm – facilitates code reuse
- Java is our *vehicle* for exploring OOP concepts and implementing data structures
- Among most popular languages for more than *two decades*
  - Hits a sweet spot

Course themes

Programming languages and paradigms

Producing correct & maintainable software

Organizing information in memory



Comparing algorithm performance

Software systems are BIG

Grappling with the scale

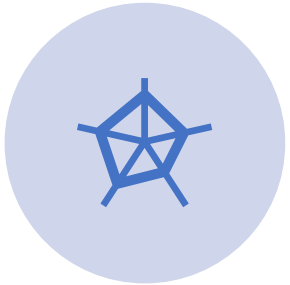
- Who enjoys debugging?
- Who can afford to spend lots of time debugging?
- Can you afford a bug?

Course themes

Programming languages and paradigms

Producing correct & maintainable software

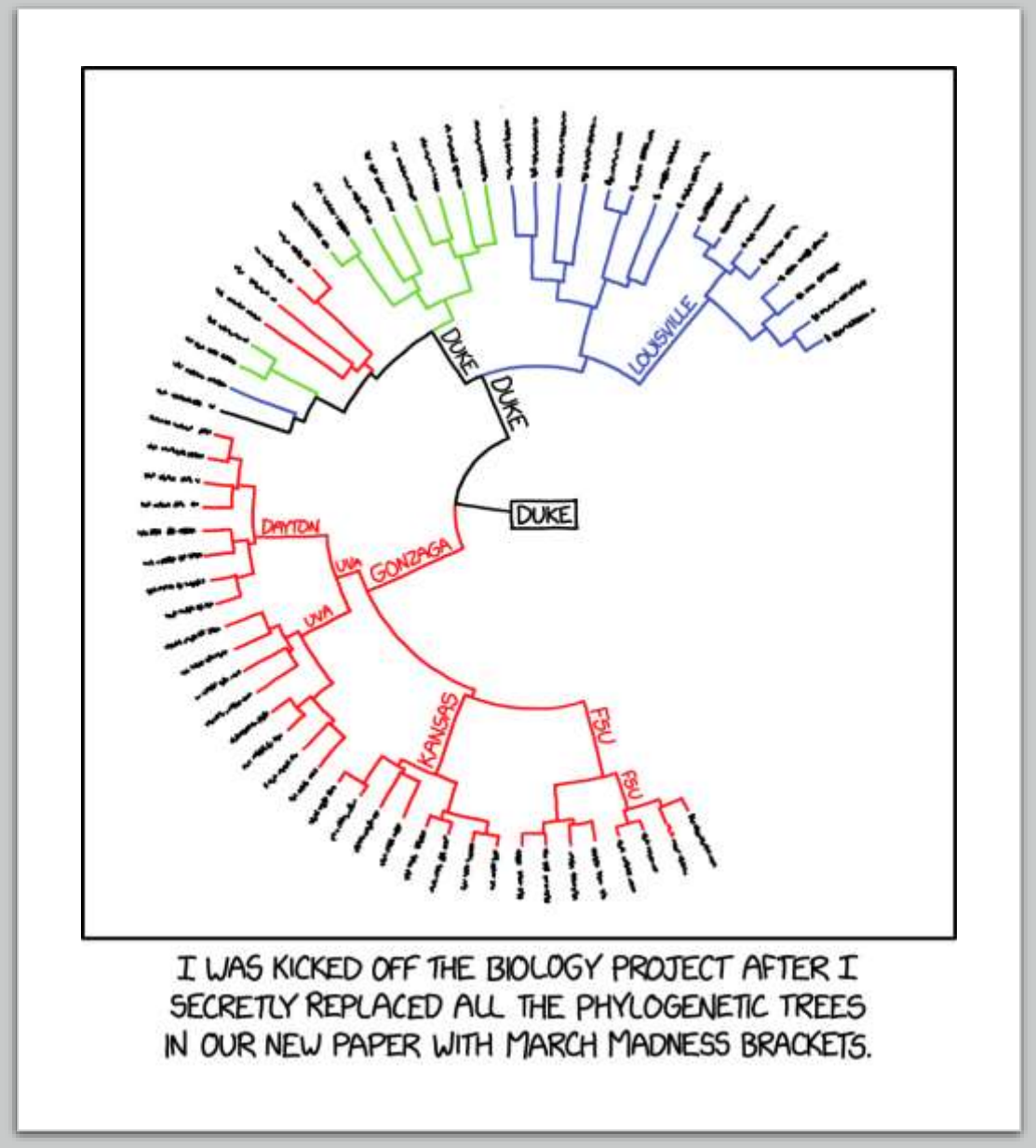
Organizing information in memory



Comparing algorithm performance

The world is richer than lists

- How might we organize data so that we can compute with…
  - Friend networks
  - Road maps
  - Prioritized tasks
  - Evolutionary lineages
  - Possible board game moves
  - Language



Course themes

Programming languages and paradigms

Producing correct & maintainable software

Organizing information in memory

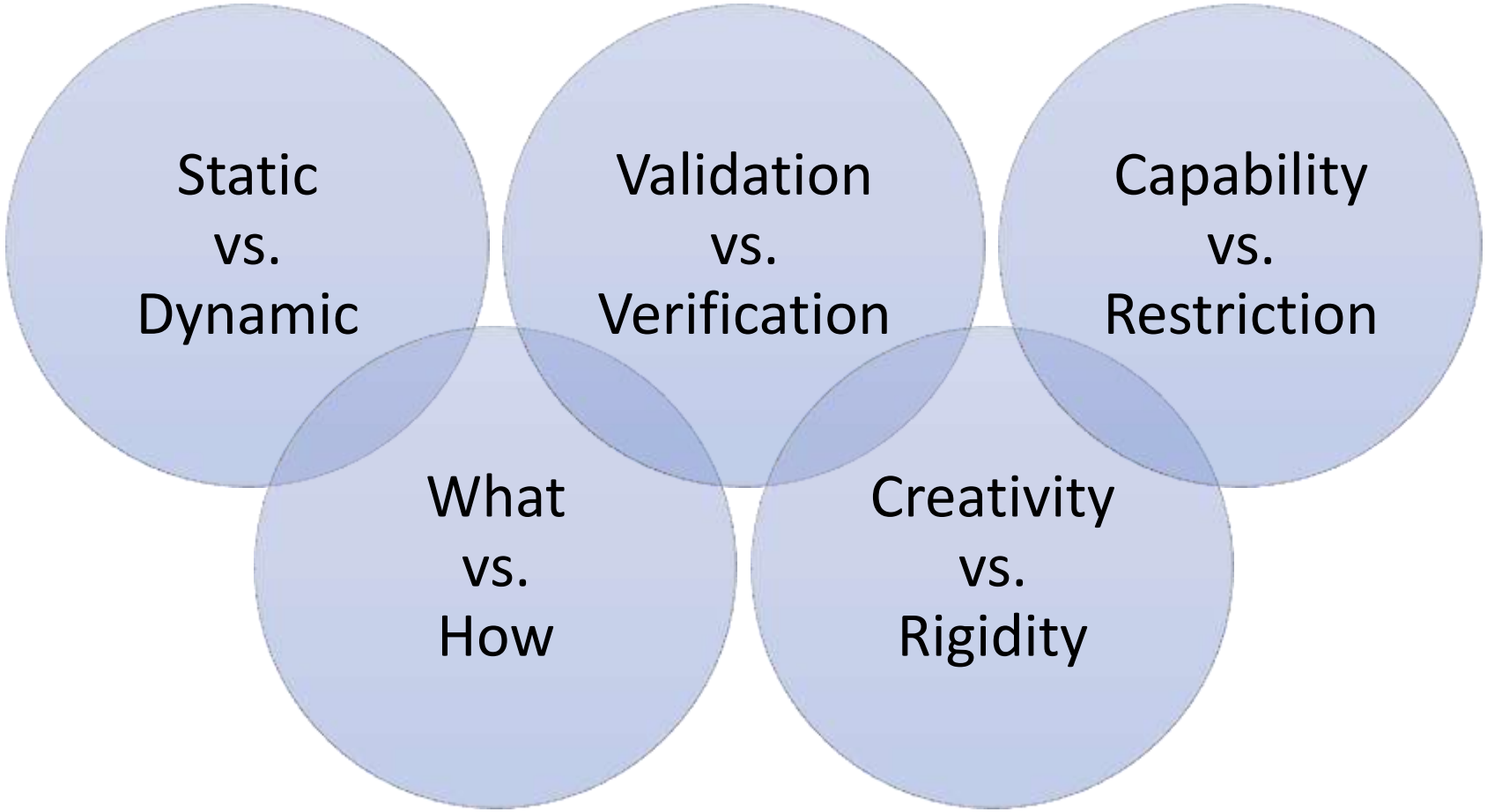
Comparing algorithm performance

Poll: algorithm efficiency

You have a ring of 50 chemistry flashcards, sorted alphabetically. Your study partner gives you a jumbled pile of 50 new cards. Which is the more efficient way to incorporate them into your alphabetized ring?

A. Find the appropriate place for each new card, inserting them one at a time
B. Add all the new cards to the end, then sort the entire ring
C. Both require about equal work

Recurring dualities



A taste of Java

Survey

What programming language are you most comfortable with?

A. Java
B. Python
C. MATLAB
D. C/C++
E. Other

Starting point (Python)

```
def hoursToSeconds(hours):
    """Return number of seconds in `hours` hours."""
    minutes = 60*hours
    seconds = 60*minutes
    return seconds
```

Poll

What will `hoursToSeconds('1')` do?

A. Syntax error
B. Runtime error
C. Return '3600'
D. Return something else

Let’s write some Java!

- Rules:
  1. All code must be written inside of a function (“method”) – no scripts
  2. All methods must be defined inside of a “class”

Ask questions

If something in lecture doesn’t make sense, please raise your hand

Types

- A set of allowed values
- A list of supported operations

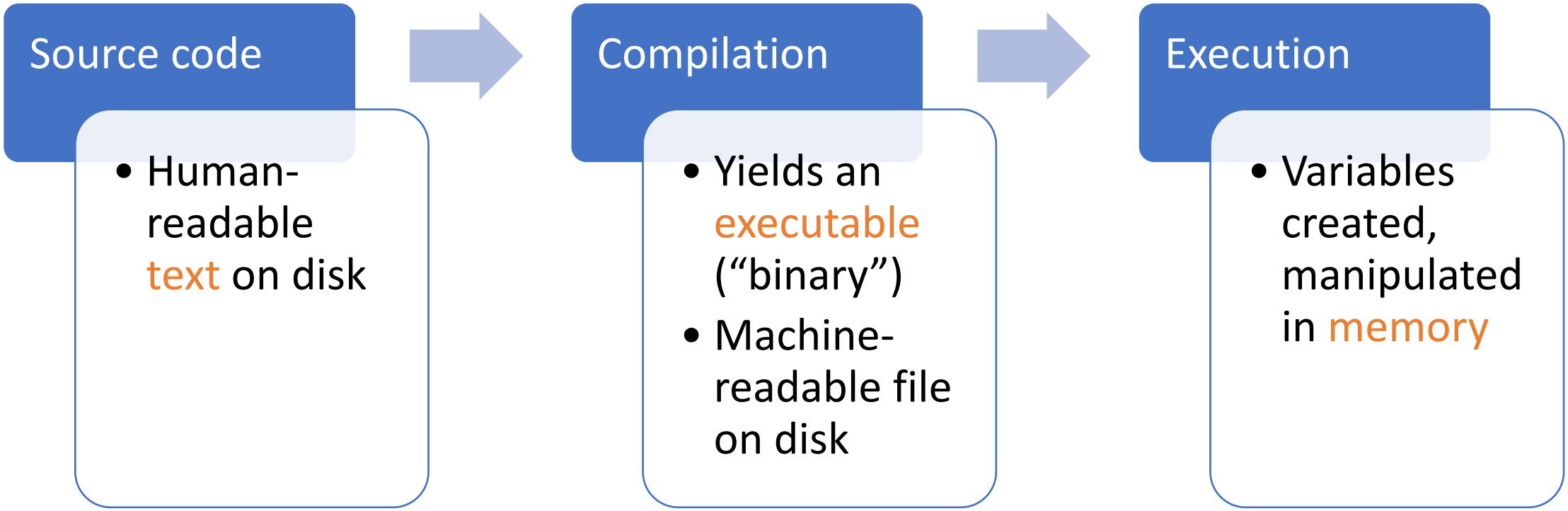
- Java's *primitive* types:
  - Integers (`int`)
  - Floating-point numbers (`double`)
  - Booleans (`boolean`)
- Some values can be converted to another type
  - Sometimes implicit (*widening*): `1.8*20 + 32`
  - Sometimes explicit (*cast*): `(double)3/5`

Static types

- Dynamic typing (Python): values know what their type is at runtime
  - Danger: might discover that a requested operation is invalid during the middle of running a program

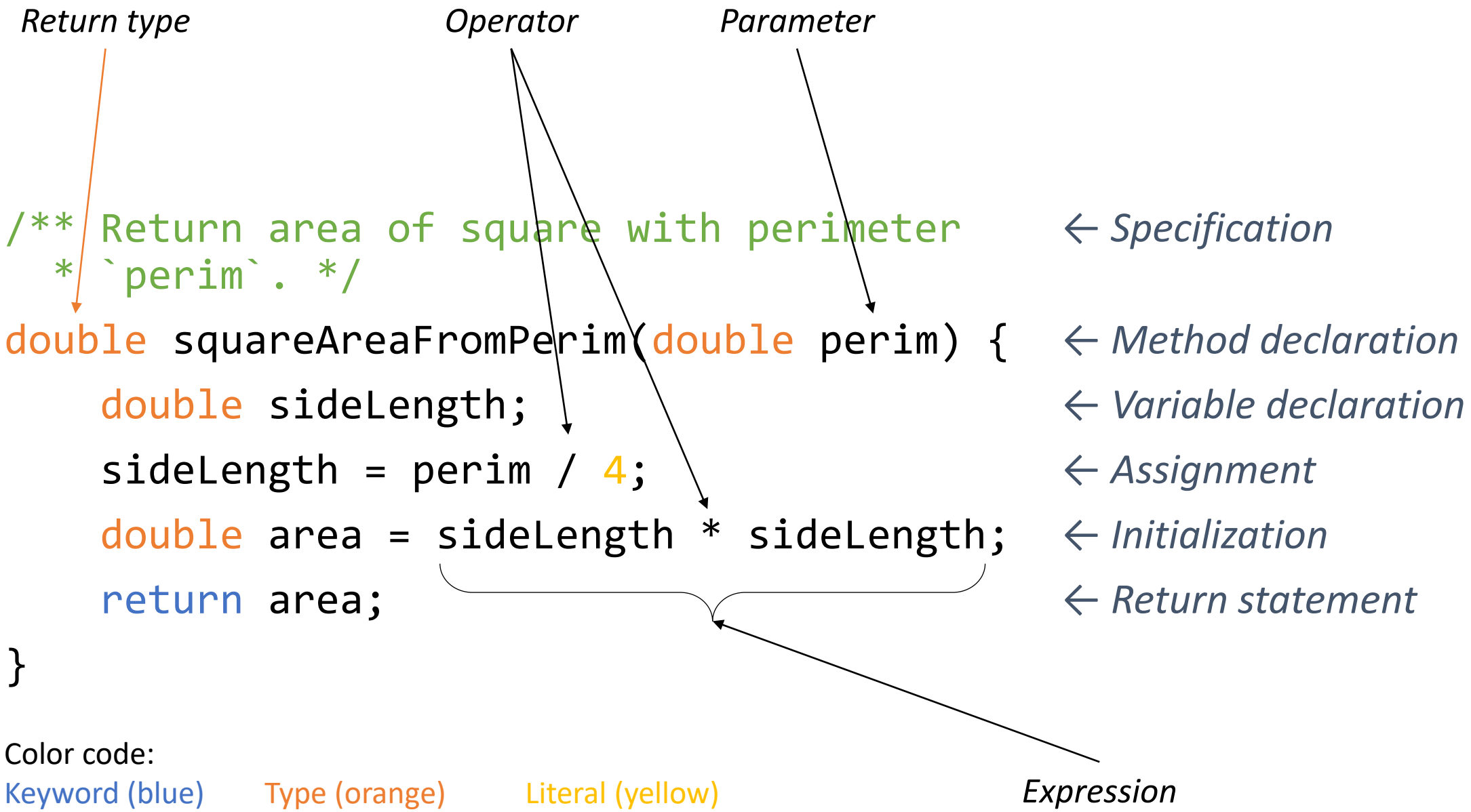
- Static typing (Java): expressions in source code have a type known at compile-time
  - Variables restrict what types of values can be stored in them
  - Invalid operations flagged before program ever runs

Compilation

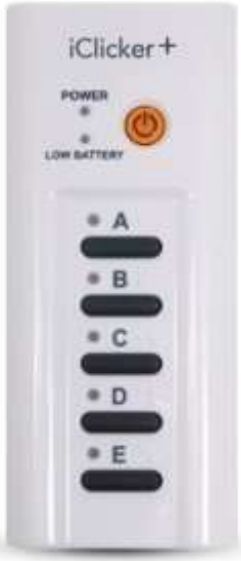


Requirements of static typing

1. Variables must be *declared* with a type before use
2. Functions (“methods”) must specify their *return type*


Return type
Operator
Parameter
/** Return area of square with perimeter
* `perim`. */
double squareAreaFromPerim(double perim) {
double sideLength;
sideLength = perim / 4;
double area = sideLength * sideLength;
return area;
}
← Specification
← Method declaration
← Variable declaration
← Assignment
← Initialization
← Return statement
Color code:
Keyword (blue)
Type (orange)
Literal (yellow)
Expression

Poll: type of expression



What is the **static type** of the highlighted *expression*?

A. `int`

B. `double`

C. `boolean`

D. `String`

E. It depends on the values

```
int x;
double y;
boolean z;
... // assign values

println((x < 2*y) && z);
```

Why static typing?

Bigger, better software

- Catches bugs sooner, before damage
  - Especially ones triggered by “unlucky” values
- Documents requirements / assumptions
- Helps tooling

Downsides?

- More code to write (in Java)
- More work to relax assumptions

Reading

- Website: Transition to Java
- Textbook: Supplement 1

If you have questions about the reading, ask on Ed!

Stretch!

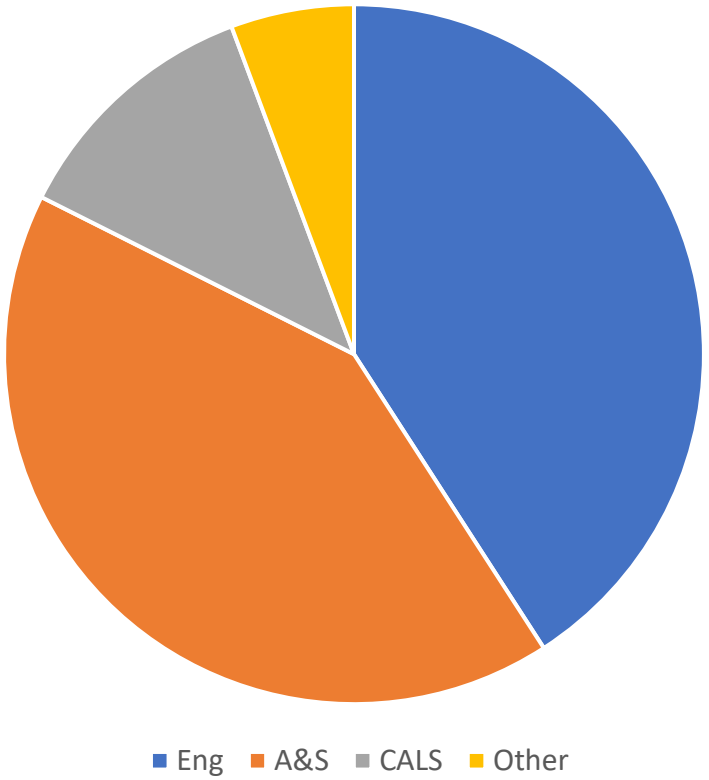
Take 60s to collect your thoughts

Curran Muhlberger

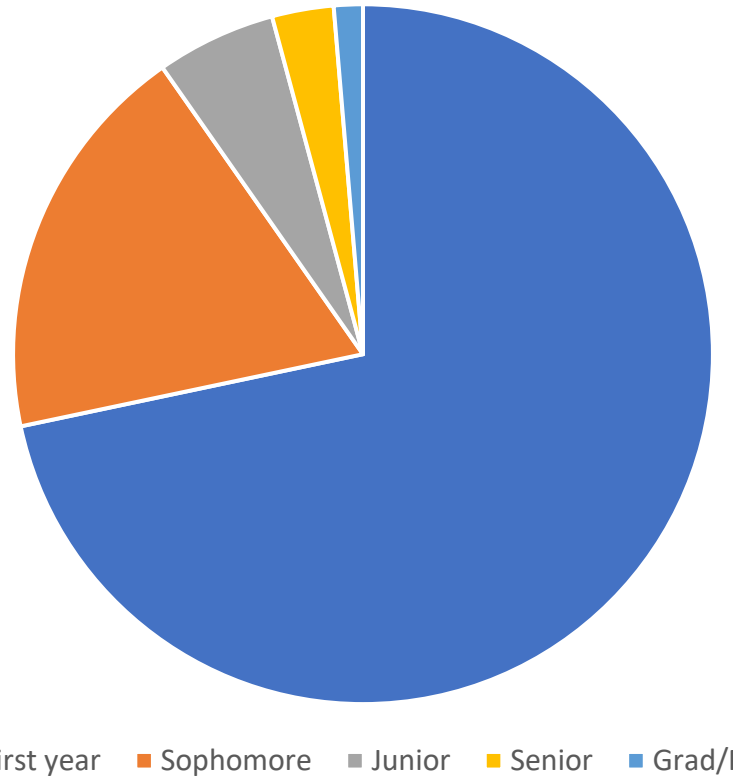
- PhD (Physics), Cornell University
- BSs (Physics, Math, Astronomy), University of Maryland
- Software engineer at SpaceX
- Regularly teach CS 1112, CS 2110
- Interests: DIY, Space!

Who are you?

College



Year



Why are you here?

Discuss

Find out why your new 2110 friends are here!

Go deeper than just “required class.” :)

You Belong Here

Even if sometimes you feel like this

Course Staff

- 64 staff
  - 45 consultants, 10 ugrad TAs, 7 grad TAs
  - 1 course coordinator (Ms. Corey Torres)
  - 1 professor
- About 10:1 ratio

- Will have 160+ person-hours of office/consulting hours each week
- Consulting starts tonight!
  - Rhodes 405, QueueMeIn

Office hours

The syllabus

- What work will you submit?
  - Weekly Canvas quizzes
  - Weekly discussion activities
  - Programming assignments (6x)
  - Exams (2 prelims, 1 final)

- Everything else?
  - On the course website!
  - https://www.cs.cornell.edu/courses/cs2110/2024sp/
  - (also linked from Canvas)

Next up

- **Discussion on Tue/Wed:** IDE setup, debugging
- **Thursday lecture:** variables, reference types (objects)

- **Tasks**
  - Syllabus quiz: released today
  - A1: released today
  - Q1: to be released Thursday