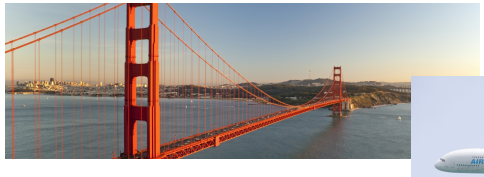
CS 326
Software Methods

Stephen Freund

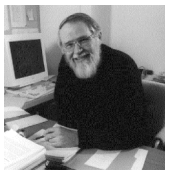
Why Is Programming So Hard?



Software is different from other artifacts

- We build general, reusable mechanisms
- Not much repetition, symmetry, or redundancy
- Large systems have millions of distinct complex parts

"Controlling complexity is the essence of computer programming."



-- Brian Kernighan

(UNIX, AWK, C, …)

Goals

- Primary focus: writing correct programs
  - What does it mean for a program to be correct?
  - How do we determine if a program is correct?
  - How do we build correct programs?
- Will cover both *principles* and tools.
- Tools change, principles are forever...

Outcomes

- Better at:
  - design
  - writing
  - debugging
  - using development tools
  - evaluating quality / behavior
  - *communication*
    - Can you convince yourself and others something is correct via precise, coherent explanations?
- Essential skills regardless of what you do next
- Work hard. Have fun. Build nifty systems.

5

A Problem

"Complete this method so that it returns the index of the max of the first **n** elements of the array **a**."

```
func indexOfMax(a: [Int], n: Int) -> Int {
  …
}
```

6

A Problem

"Complete this method so that it returns the index of the max of the first **n** elements of the array **a**."

```
func indexOfMax(a: [Int], n: Int) -> Int {
  …
}
```

What should we ask about the *specification?*

Given (better) specification, how many possible implementations are there?

7

Prerequisites

- Proficient in Java, eg:
  - Sharing:
    - Distinction between == and equals()
    - Aliasing: multiple references to the same object
  - Object-oriented programming:
    - Inheritance and overriding
    - Objects/values have a run-time type
  - Subtyping
    - Expressions have a compile-time type
    - classes vs. interfaces
- Reasoning and proof techniques
- Basic Unix and OS X skills

8

Course Components

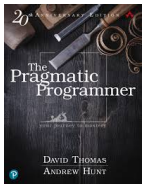
- Lecture
- Reading
- Written Homework
- Labs
- Final Project
- Midterm Exam

- CS 326 Web Page
- Honor Code

9

Resources

- The Pragmatic Programmer
  - Thomas and Hunt (2019)
  - Collection of best practices



- Class notes, additional readings
- Swift Language and API Docs:
  - Swift Language
  - https://developer.apple.com/documentation/

10

Pragmatic Programmer

- Advice from top-notch programmers
- Stuff all serious programmers should know
- Approachable but sometimes challenging
- Only partial overlap with lecture

- Keep up with reading
  - Reading and contemplating design is essential
  - Time investment that pays dividends in the long run

11

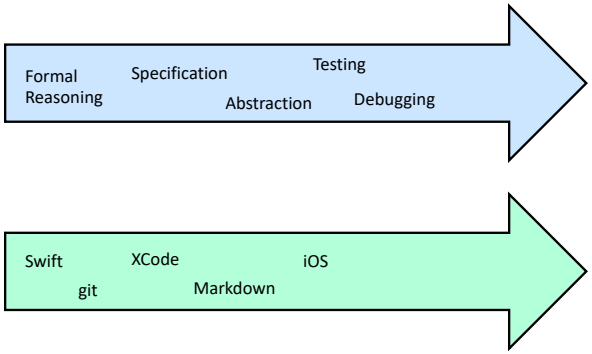
Programming is Hard

- Despite decades of practice, still surprisingly difficult to specify, design, implement, test, and maintain even small, simple programs.

- Assignments will be reasonable if you apply the techniques taught in class…
  ... but likely very difficult to do brute-force
  ... and almost certainly impossible (or at least painful) unless you start early.
- **Think before you type!**

12

Looking Ahead A Few Weeks



13

You Have Lab Today!

- Lab 0
- Set up lab environment
- Git
- Markdown
- Swift Tutorial

14

You Have Homework For Tues.

- HW 1
- Design algorithm to meet a simple specification
- Working up to reasoning about large designs

15

Motivation/Structure of CS 326

- My own experiences
  - 25+ years of building systems (successes/failures)
  - 20 years of advising student projects
  - My research on languages and defect detection
- Hard work, course development, and insights of many others
  - Michael Ernst, Hal Perkins, Dan Grossman, David Notkin, Zach Tatlock, Paul Hegarty, Scott Smith

16