You Made It!

Thank your TAs

Learning Objectives

or, "What will I learn in this class?"

1. **Functionality/Behavior:** Write functionally correct and efficient Java programs and systems of medium to large length and complexity that meet a provided specification and/or solve a specified problem
2. **Comprehension:** Trace and predict the behavior of programs and systems
3. **Data Abstraction:** Select and apply appropriate abstract data types to manage program state
4. **Data Structures:** Design, implement, and modify data structures to efficiently and effectively provide a defined set of operations
5. **Functional Abstraction:** Document, maintain, and utilize appropriate abstractions between the implementer and client of a library
6. **Decomposition:** Solve problems by breaking them into subproblems and recombining the solutions using techniques such as methods, inheritance, and recursion
7. **Code Quality:** Define programs that are well-written, readable, maintainable, and conform to established standards

Road Map

CS Concepts

- Client/Implementer
- Efficiency
- Recursion
- Regular Expressions
- Grammars
- Searching / Sorting
- Backtracking
- Hashing
- Huffman Compression

Data Structures

- Lists
- Stacks
- Queues
- Sets
- Maps
- Priority Queues

Java Language

- Exceptions
- Interfaces
- References
- Comparable
- Generics
- Inheritance / Polymorphism
- Abstract Classes

Java Collections

- `Arrays`
- `ArrayList` 🛠
- `LinkedList` 🛠
- `Stack`
- `TreeSet` / `TreeMap` 🛠
- `HashSet` / `HashMap` 🛠
- `PriorityQueue`

Comparison to CSE 142 (or similar)

CSE 142

- Control structures
- Simple (primitive) data
- Client view
- Java as focus
- *How do I do this?*

CSE 143

- Data structures
- Complex data
- Implementer view
- Java as example
- *What can I do with this?*

Underlying Skills

or "What did I learn in this class without realizing it?"

- **Abstraction**
  - Leverage existing components without understanding details
  - Create components that can be used as black boxes
- **Problem solving**
  - Decomposing a large problem into smaller ones
- **Design tradeoffs**
  - Algorithm analysis - scalability and growth
  - Keeping code easy to read for maintainability
- **Recursive thinking**
  - Reason about problems in terms of self-similarity
  - Write very short code to achieve complex behaviors

FLYING PIGEON

Digression: My New Hobby

Amigurumi: Japanese art of creating crocheted or knitted stuffed toys

Applications of CS

or "What can I do with what I learned?"

- Detect and prevent toxicity online
- Digitize basketball players
- Help DHH people identify sounds
- Figure out how to best distribute relief funds
- Recognize disinformation online
- Make movies
- Improve digital collaboration
- Fix Olympic badminton
- And so much more!

Future Courses

or "What can I do next?"

Majors

Course	Overview
CSE 311	Mathematical foundations
CSE 351	Low-level computer organization/abstraction
CSE 331	Software design/implementation
CSE 341	Programming languages (!)
CSE 340	Interaction programming

All offered Spring 2021

Non-majors

Course	Overview
CSE 154	Intro. to web programming (several languages)
CSE 163	Intermediate programming, data analysis (Python)
CSE 180	Introduction to data science (Python)
CSE 373	Data structures and algorithms (non-majors)
CSE 374	Low-level programming and tools (C/C++)
CSE 416	Intro. to Machine Learning

See: https://www.cs.washington.edu/academics/ugrad/current-students and https://www.cs.washington.edu/academics/ugrad/nonmajor-options/nonmajor-courses

Frequently Asked Questions

- How can I get better at programming?
  - Practice!
- How can I learn to X?
  - Search online, read books, look at examples
- What should I work on next?
  - Anything you can think of! (Here are some ideas)
  - Beware: it’s hard to tell what’s easy and what’s hard.
- Should I learn another language? Which one?
  - That depends– what do you want to do?
- What’s the best programming language?
  - 😬 (take CSE 341)

Thank you!!!

Ask Me (Almost) Anything!