LEC 01

Welcome!

Use the Zoom chat:
Introduce yourself! What are you most excited about in this class? What’s your least favorite vegetable?

Instructor: Aaron Johnston
TAs: Timothy Akintilo, Brian Chan, Joyce Elauria, Eric Fan, Siddharth Vaidyanathan, Farrell Fileas, Leona Kazi, Keanu Vestil, Howard Xiao
Lecture Outline

• **Introductions**

• About this Course
  - Course Components & Tools
  - Policies
  - Making the Most of this Class

• Abstract Data Types
Course Staff

• Instructor: Aaron Johnston
  - Grad student from UW CSE, previously taught CSE 333 (Systems Programming) and CSE 390B (Academic Skill-Building)

• Teaching Assistants:
  - Available in section, office hours, discussion board, and 1:1 meetings
  - Invaluable source of information & help in this course

• We’re excited to get to know you!
  - Our goal is to help you succeed
Students

• Currently 205 students registered for the course
  - Over double the size of last year’s summer 373 offering (!)

• If you’re waiting to register, unfortunately there are no overloads available, and the course staff does not have add codes
  - Reach out to ugrad-advisor@cs.washington.edu with any registration questions

• Strength in numbers
  - With 205 students, if you’re confused about something, I guarantee someone else is too!
  - Students come from all different backgrounds & majors
What is this Class?

• CSE 143 – Object Oriented Programming
  - Classes and Interfaces
  - Methods, variables and conditionals
  - Loops and recursion
  - Linked lists and binary trees
  - Sorting and Searching
  - O(n) analysis
  - Generics

• CSE 373 – Data Structures and Algorithms
  - Design decisions
  - Design analysis
  - Implementations of data structures
  - Debugging and testing
  - Abstract Data Types
  - Code Modeling
  - Complexity Analysis
  - Software Engineering Practices
Why 373?

1. Build a strong foundation of data structures and algorithms that will let you tackle the biggest problems in computing

Why 373?

2. Pick up the vocabulary, skills, and practice needed to make **design decisions**. Learn to **evaluate** the tools in your CS toolbox

- Differences between technical implementations
- Evaluation can mean many different things!
Lecture Outline

• Introductions

• About this Course
  - Course Components & Tools
  - Policies
  - Making the Most of this Class

• Abstract Data Types
## Course Components

### LECTURES (x26)
- Held live via Zoom
- Recordings available after
- In-lecture activities
- Introduction to course concepts

### SECTIONS (x9)
- Held live via Zoom
- Review videos available after
- Practice problems, reviews, TA advice
- Preparation for exams

### PROJECTS (x5)
- Partner recommended
- Programming in Java
- Applying & implementing course concepts
- More practical

### EXERCISES (x4)
- Individual
- Written problems, focusing on the “why”
- More conceptual

### EXAMS (x2)
- Available over a multi-day window, complete whenever works for you
- More details as we get closer
Using Zoom

• Two important ways to interact in lecture:

• Open Participants Pane
  - Use the feedback buttons for quick cues

• Open Chat Pane
  - Type your questions in the chat
    - Other students, TAs, or I can answer!
  - Please conduct yourself as you would in a classroom – be respectful!

• Sign in early to chat with other students, warm up for the day
Using PollEverywhere

• Sometimes I’ll ask for more involved feedback or we’ll pause to do an active learning activity
• Go to pollev.com/uwcse373 to register and participate
• Let’s practice: which puppy is cutest?
Course Website

Undergraduate Course on Pattern Recognition

Contains most course info – check frequently!
- Announcements, Calendar, Lecture Slides, Assignment Specs, Office Hours schedule, Staff Bios, Important Links
Course Website

Contains most course info – check frequently!
- Announcements, Calendar, Lecture Slides, Assignment Specs, Office Hours schedule, Staff Bios, Important Links

Please familiarize yourself with the course syllabus this week!
Other Course Tools

**Piazza**
- Discussion Board & Announcements
- Please ask AND answer!
- Anonymous option
- Opt out of Piazza Network

**Discord**
- Community: meet other students, form study groups
- Most Office Hours held here
- More details to come

**Gitlab**
- Everyone gets a git repo
- We’ll distribute starter code, you’ll push your work
- More details to come

**Gradescope**
- Submit all your assignments
- Get feedback

**Canvas**
- Only used for Zoom recordings and gradebook
Lecture Outline

• Introductions

• About this Course
  - Course Components & Tools
  - Policies
  - Making the Most of this Class

• Abstract Data Types
Grading Breakdown

• Your grade will consist of the following weighted categories:

<table>
<thead>
<tr>
<th>Category</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming Projects</td>
<td>45%</td>
</tr>
<tr>
<td>Individual Exercises</td>
<td>25%</td>
</tr>
<tr>
<td>Exam I</td>
<td>15%</td>
</tr>
<tr>
<td>Exam II</td>
<td>15%</td>
</tr>
</tbody>
</table>

• Instead of curving the class as usual, we’ll use a bucket system:
  - These are minimum GPA guarantees – may adjust upward

<table>
<thead>
<tr>
<th>Percentage</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>95%</td>
<td>4.0</td>
</tr>
<tr>
<td>90%</td>
<td>3.5</td>
</tr>
<tr>
<td>80%</td>
<td>3.0</td>
</tr>
<tr>
<td>60%</td>
<td>2.0</td>
</tr>
<tr>
<td>50%</td>
<td>0.7</td>
</tr>
</tbody>
</table>
Assignment Policies

Collaboration & Academic Integrity

• These concepts are hard: we strongly encourage discussion + collaboration!
  - Don’t attempt to gain credit for something you didn’t do
  - In general, share ideas and work together, but don’t copy work. Never show someone else your code or solution write up.
  - Always cite the help you receive
  - Full collaboration with your partner on projects!

• Read full policy in Syllabus

Lateness

• You get 7 “free” late days for the quarter – submit 24 hours late with no penalty
  - Use on projects or exercises
• After that, -5% each day late
• No assignment can be submitted >72 hours late
  - Except with instructor permission
Textbook

• Data Structures and Algorithm Analysis in Java by Mark Allen Weiss

• Completely optional
  - Nothing assigned out of the textbook
  - No readings

• Advice: only purchase if you learn best with a textbook, otherwise not recommended
Lecture Outline

• Introductions

• About this Course
  - Course Components & Tools
  - Policies
    - Making the Most of this Class

• Abstract Data Types
Getting Help

• Discussion Board
  - Feel free to make a public or private post on Piazza
  - We encourage you to answer other peoples’ questions! A great way to learn

• Office Hours
  - TAs can help you face to face in office hours, and look at your code
  - Discord gives great flexibility – feel free to join your peers in the OH chat and listen in to see who else has a similar problem

• Section
  - Work through related problems, get to know your TA who is here to support you

• Your Peers
  - We encourage you to form study groups! Discord or Piazza are great places to do that

• Email the Staff List
  - You can always email cse373-staff@cs.washington.edu if you don’t know how to get help – we’ll work to get you the support you need
Help Us Improve!

• We’re still learning how to do this online 😊
  - Thank you in advance for your patience and understanding
  - We really value your feedback!
  - Let us know what’s working and what isn’t working for you
  - Something that went well in another course? Tell us about it!

• Email the course staff at cse373-staff@cs.uw.edu

• Submit feedback via the Anonymous Feedback Tool (linked under “Course Tools” on the website)
Metacognition

• **Metacognition**: asking questions about your solution process.

• Examples:
  - **While debugging**: explain to yourself why you’re making this change to your program.
  - **Before running your program**: make an explicit prediction of what you expect to see.
  - **When coding**: be aware when you’re not making progress, so you can take a break or try a different strategy.
  - **When designing**:
    - Explain the tradeoffs with using a different data structure or algorithm.
    - If one or more requirements change, how would the solution change as a result?
    - Reflect on how you ruled out alternative ideas along the way to a solution.
  - **When studying**: what is the relationship of this topic to other ideas in the course?
The World Around 373

• Our goal is to give you a great 373 experience
  - But CSE 373 does not exist in a vacuum – there’s a lot going on in the world right now that can impact your education

• We’ve designed course policies for maximum flexibility: plenty of late days, take-home exams, no participation
  - But we cannot cover every individual situation

• Please reach out if you need accommodations of any kind to deal with these unfamiliar situations
Lecture Outline

• Introductions

• About this Course
  - Course Components & Tools
  - Policies
  - Making the Most of this Class

• Abstract Data Types
Data Structures & Algorithms

• Data Structure:
  - A way of organizing, storing, accessing, and updating data
  - Examples from CSE 14x: Arrays, Linked Lists, Stacks, Queues, Trees

• Algorithm:
  - A series of precise instructions to produce a specific outcome
  - Examples from CSE 14x: Binary Search, Merge Sort, Recursive Backtracking
**Review  Interface vs. Implementation**

- In Java, an **interface** is a data type that specifies what to do but not how to do it.
  - **List**: an ordered sequence of elements.

- A **subtype** implements all methods required by the interface.
  - **ArrayList**: Resizable array implementation of the List interface.
  - **LinkedList**: Doubly-linked implementation of the List interface.
Review Client vs. Object

Client Classes

- A class that is executable, in Java:
  contains a Main method

```java
public static void main(String[] args)
```

Object Classes

- A coded structure that contains data and behavior
- Start with the data you want to hold, organize the things you want to enable users to do with that data
ADTs: Abstract Data Types

• Java interfaces represent the concept of abstract data types.

• An **abstract data type** is a data type that does not specify any one implementation.

• **Data structures** implement ADTs.
  • **Resizable array** can implement List, Stack, Queue, Deque, PQ, etc.
  • **Linked nodes** can implement List, Stack, Queue, Deque, PQ, etc.

**List ADT**

*A collection storing an ordered sequence of elements.*

• Each element is accessible by a zero-based index.
• A list has a size defined as the number of elements in the list.
• Elements can be added to the front, back, or any index in the list.
• Optionally, elements can be removed.
Where we’re Headed: ADTs we’ll look at

- List
- Set
- Map
- Stack
- Queue
- Priority Queue
- Graph
- Disjoint Set
Learning to Bake in a CSE Class

• Think of what you’ll learn this quarter as a cookbook
  - ADTs are the chapters/category: Soups, Salads, Cookies, Cakes, etc
    - High-level descriptions of a category of functionality
    - You don’t serve a soup when guests expect a cookie!
  - Data structures are the recipes: chocolate chip cookies, snickerdoodles, etc
    - Step-by-step, concrete descriptions of an item with specific characteristics
    - Understand your tradeoffs before replacing carrot cake with a wedding cake

• When you go out into the world …
  - Figure out which category is required
  - Choose the specific recipe that best fit the situation