LEC 00

CSE 122

Welcome!

BEFORE WE START

Talk to your neighbors:
Introduce yourself to your neighbor!

What is your name? Major? What did you do this summer?

Music: Hunter/Miya's Playlist

Instructor Hunter Schafer / Miya Natsuhara

TAs	Ajay	Gaurav	Melissa
	Andrew	Hilal	Noa
	Anson	Hitesh	Parker
	Anthony	Jake	Poojitha
	Audrey	Jin	Samuel
	Chloe	Joe	Sara
	Colton	Joe	Simon
	Connor	Karen	Sravani
	Elizabeth	Kyler	Tan
	Evelyn	Leon	Vivek

Questions during Class?

Raise hand or send here

sli.do #cse-

Lecture Outline

- **Introductions**
- About this Course
  - Course Components & Tools
  - Policies
  - Making the Most of this Class
- Intro/Review Java

Course Staff

- Instructor: Hunter Schafer
- Instructor: Miya Natsuhara

- Teaching Assistants: 31 Awesome TAs
  - 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 503 students registered for the course!

- Strength in numbers
  - With 503 students, if you’re confused about something, I guarantee someone else is too!
  - Students come from all different backgrounds & majors & interests in future career goals.

- Focus on us trying to help you build community
  - Meet others in the class to form study groups or have people you can work with.

What is this Class?

CSE 121 – Computer Programming I
or **Other Programming Experience**

- Print statements
- Data types (int, String, boolean)
- Methods / Functions
  - Parameters
  - Returns
- Control structures
  - Loops
  - Conditionals
- File I/O
- Arrays
- **Computational Thinking** (language agnostic)

CSE 122 – Computer Programming II

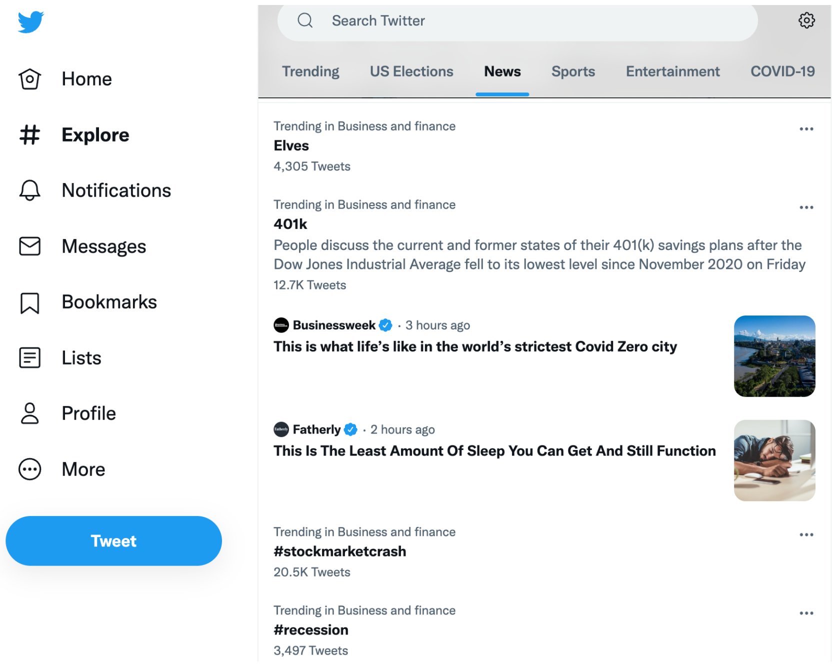
- Decomposing large problems into smaller, manageable, subproblems
- Using data structures
  - List
  - Stacks / Queues
  - Sets
  - Maps
- Object Oriented Programming
  - Interfaces

Prerequisite Knowledge

- Students entering CSE 122 are coming from many of different backgrounds
  - UW: CSE 121 (Soon™) or other intro programming course
  - Community College: Intro Programming Course
  - High School Programming Course (e.g., UWHS, AP CS, IB CS, etc.)
  - Self-taught or other previous experience
- Importantly: CSE 122 is in Java, but we **do not expect prior experience in Java!** Do expect knowing the list of CSE 121 topics in some language.
  - Students who do not have experience in Java will be focusing on practicing the programming skills you know in a new language!
  - You will find the Java Tutorial and Programming Assignment 0 very helpful!
- If you want to know if this class is the right fit for you, take the Allen School Self-Placement Test

Why 122?

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



Home
Explore
Notifications
Messages
Bookmarks
Lists
Profile
More
Tweet

Search Twitter

Trending | US Elections | News | Sports | Entertainment | COVID-19

Trending in Business and finance
Elves
4,305 Tweets

Trending in Business and finance
401k
People discuss the current and former states of their 401(k) savings plans after the Dow Jones Industrial Average fell to its lowest level since November 2020 on Friday
12.7K Tweets

Businessweek · 3 hours ago
This is what life's like in the world's strictest Covid Zero city

Fatherly · 2 hours ago
This Is The Least Amount Of Sleep You Can Get And Still Function

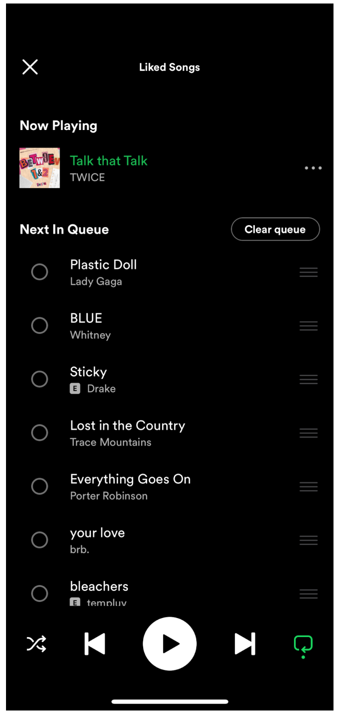
Trending in Business and finance
#stockmarketcrash
20.5K Tweets

Trending in Business and finance
#recession
3,497 Tweets

Source: Twitter 9/23

122 Data Structures

Source: Ethical CS



Liked Songs

Now Playing
Talk that Talk
TWICE

Next In Queue
Clear queue

Plastic Doll	Lady Gaga
BLUE	Whitney
Sticky	Drake
Lost in the Country	Trace Mountains
Everything Goes On	Porter Robinson
your love	brb.
bleachers	templuv

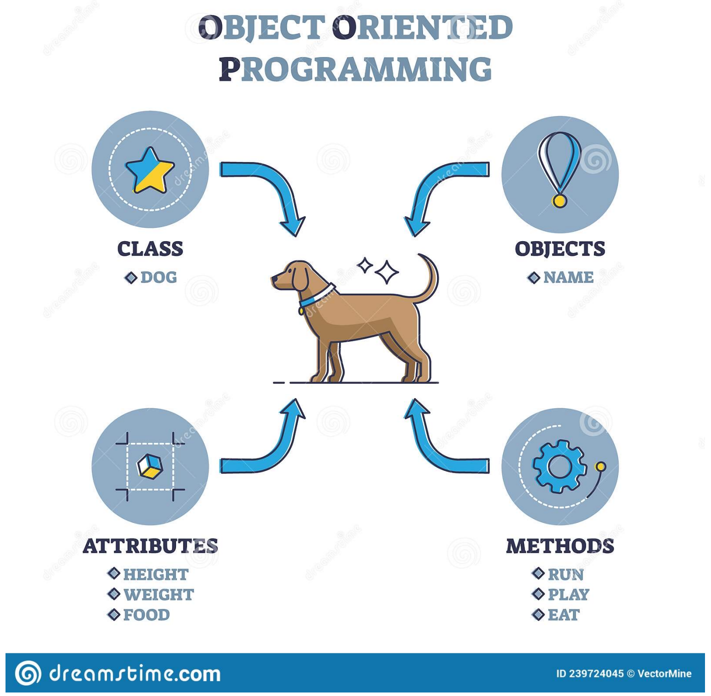
Source: Hunter's Spotify

Why 122?

2. Learn an important structural pattern for representing **objects** in code to make our code more **reusable** and **maintainable** and **easier to understand.**

- Java is designed around this idea of objects. We haven't been leveraging that yet!

- Used in almost every real-world software project.



Lecture Outline

- Introductions
- **About this Course**
  - **Course Components & Tools**
  - Policies
  - Making the Most of this Class
- Intro/Review Java

Course Components

Meetings

LECTURES (x19)

- We're here!
- Introduce concepts, practice ideas, discuss applications.
- Pre-class materials to prepare for class each day. Due **before** class.

SECTIONS (x18)

- Held in person
- More practice, reviews, applications
- TA advice, how to be an effective student
- Preparation for quizzes / exams

Assessments

PROGRAMMING ASSIGNMENTS (x4)

- Structured assignments
- Programming in Java
- Applying & implementing course concepts

CREATIVE PROJECTS (x4)

- More open-ended assignments
- Explore new ideas and applications

QUIZZES (x4)

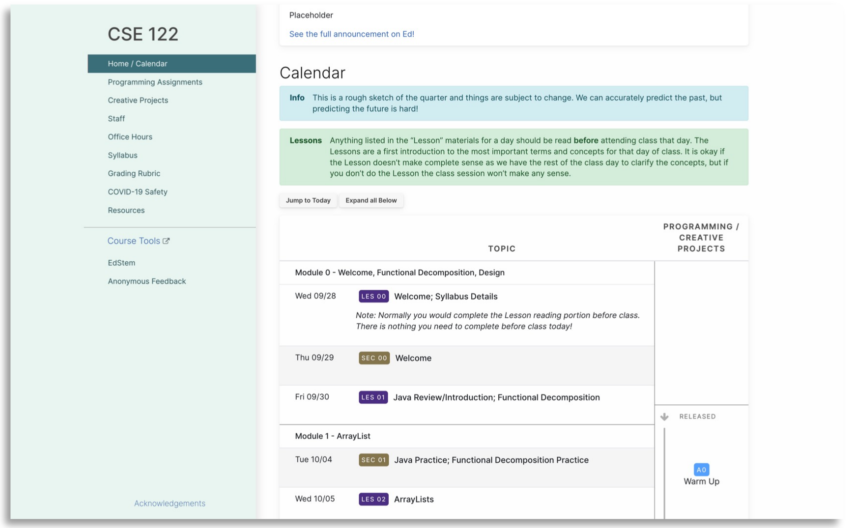
- Taken in quiz section
- 30 minutes on computer
- One retake per quiz

EXAM (x1)

- Culminating exam
- **Tuesday 12/13 @ 12:30 pm**

Course Website

cs.uw.edu/122



Placeholder

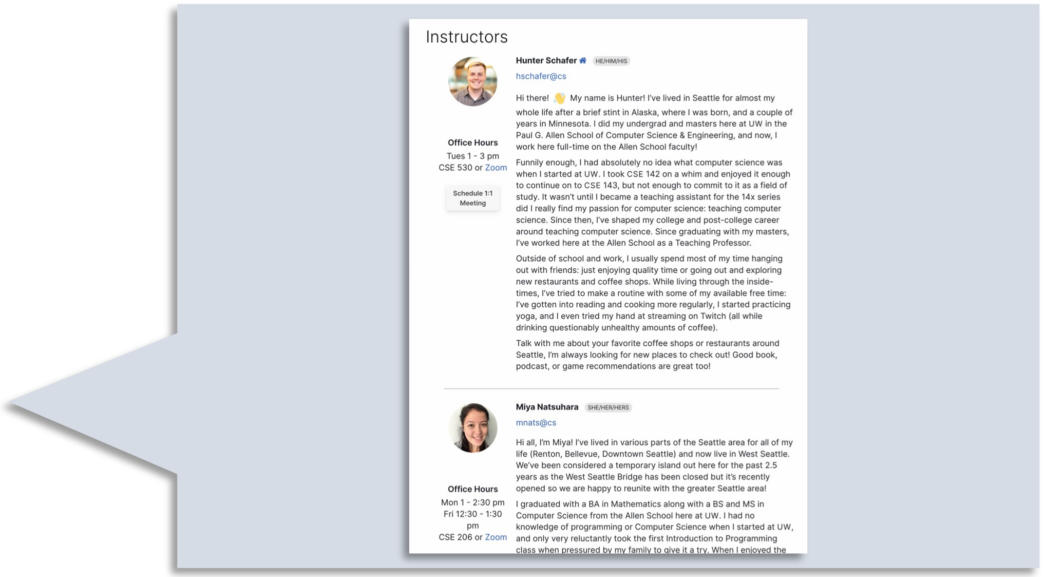
See the full announcement on Ed!

Calendar

Info This is a rough sketch of the quarter and things are subject to change. We can accurately predict the past, but predicting the future is hard!

Lessons Anything listed in the "Lesson" materials for a day should be read **before** attending class that day. The Lessons are a first introduction to the most important terms and concepts for that day of class. It is okay if the Lesson doesn't make complete sense as we have the rest of the class day to clarify the concepts, but if you don't do the Lesson the class session won't make any sense.

	TOPIC	PROGRAMMING / CREATIVE PROJECTS
Module 0 - Welcome, Functional Decomposition, Design		
Wed 09/28	LES 00 Welcome; Syllabus Details *Note: Normally you would complete the Lesson reading portion before class. There is nothing you need to complete before class today!*	
Thu 09/29	SEC 00 Welcome	
Fri 09/30	LES 01 Java Review/Introduction; Functional Decomposition	RELEASED
Module 1 - ArrayList		
Tue 10/04	SEC 01 Java Practice; Functional Decomposition Practice	A0 Warm Up
Wed 10/05	LES 02 ArrayLists	



Instructors

Hunter Schafer HE/HIM/HIS

hschafer@cs

Office Hours
Tues 1 - 3 pm
CSE 530 or Zoom

Hi there! 👋 My name is Hunter! I've lived in Seattle for almost my whole life after a brief stint in Alaska, where I was born, and a couple of years in Minnesota. I did my undergrad and masters here at UW in the Paul G. Allen School of Computer Science & Engineering, and now, I work here full-time on the Allen School faculty!

Funnily enough, I had absolutely no idea what computer science was when I started at UW. I took CSE 142 on a whim and enjoyed it enough to continue on to CSE 143, but not enough to commit to it as a field of study. It wasn't until I became a teaching assistant for the 14x series did I really find my passion for computer science: teaching computer science. Since then, I've shaped my college and post-college career around teaching computer science. Since graduating with my masters, I've worked here at the Allen School as a Teaching Professor.

Outside of school and work, I usually spend most of my time hanging out with friends: just enjoying quality time or going out and exploring new restaurants and coffee shops. While living through the inside-times, I've tried to make a routine with some of my available free time: I've gotten into reading and cooking more regularly, I started practicing yoga, and I even tried my hand at streaming on Twitch (all while drinking questionably unhealthy amounts of coffee).

Talk with me about your favorite coffee shops or restaurants around Seattle, I'm always looking for new places to check out! Good book, podcast, or game recommendations are great too!

Miya Natsuhara SHE/HER/HERS

mnats@cs

Office Hours
Mon 1 - 2:30 pm
Fri 12:30 - 1:30 pm
CSE 206 or Zoom

Hi all, I'm Miya! I've lived in various parts of the Seattle area for all of my life (Renton, Bellevue, Downtown Seattle) and now live in West Seattle. We've been considered a temporary island out here for the past 2.5 years as the West Seattle Bridge has been closed but it's recently opened so we are happy to reunite with the greater Seattle area!

I graduated with a BA in Mathematics along with a BS and MS in Computer Science from the Allen School here at UW. I had no knowledge of programming or Computer Science when I started at UW, and only very reluctantly took the first Introduction to Programming class when pressured by my family to give it a try. When I enjoyed the

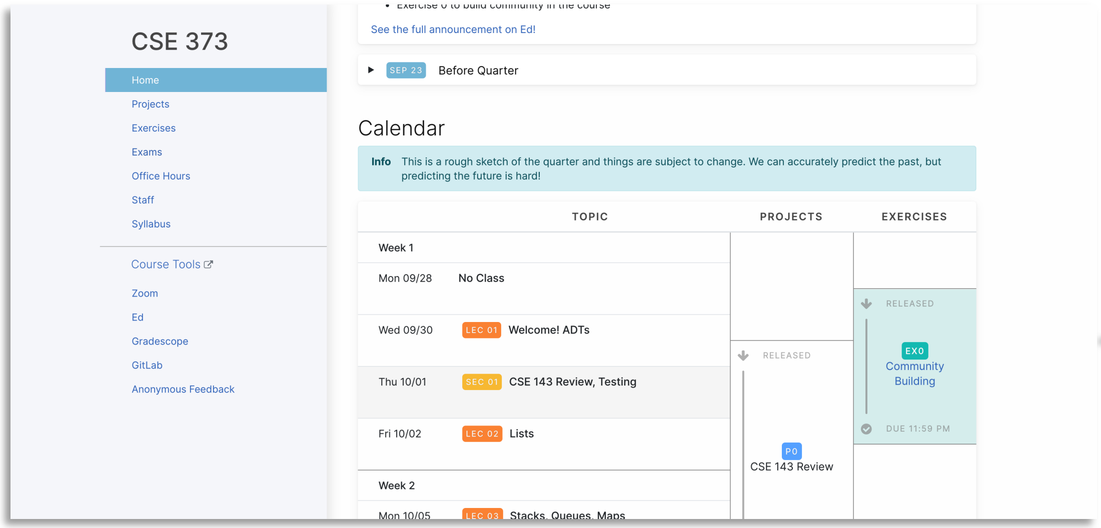
Get to know the staff

Contains most course info – check frequently!

- Announcements, Calendar, Lecture Slides, Office Hours schedule, Staff Bios, Important Links

Course Website

cs.uw.edu/122



CSE 373

- Home
- Projects
- Exercises
- Exams
- Office Hours
- Staff
- Syllabus

Course Tools

- Zoom
- Ed
- Gradescope
- GitLab
- Anonymous Feedback

• Exercise 0 to build community in the course

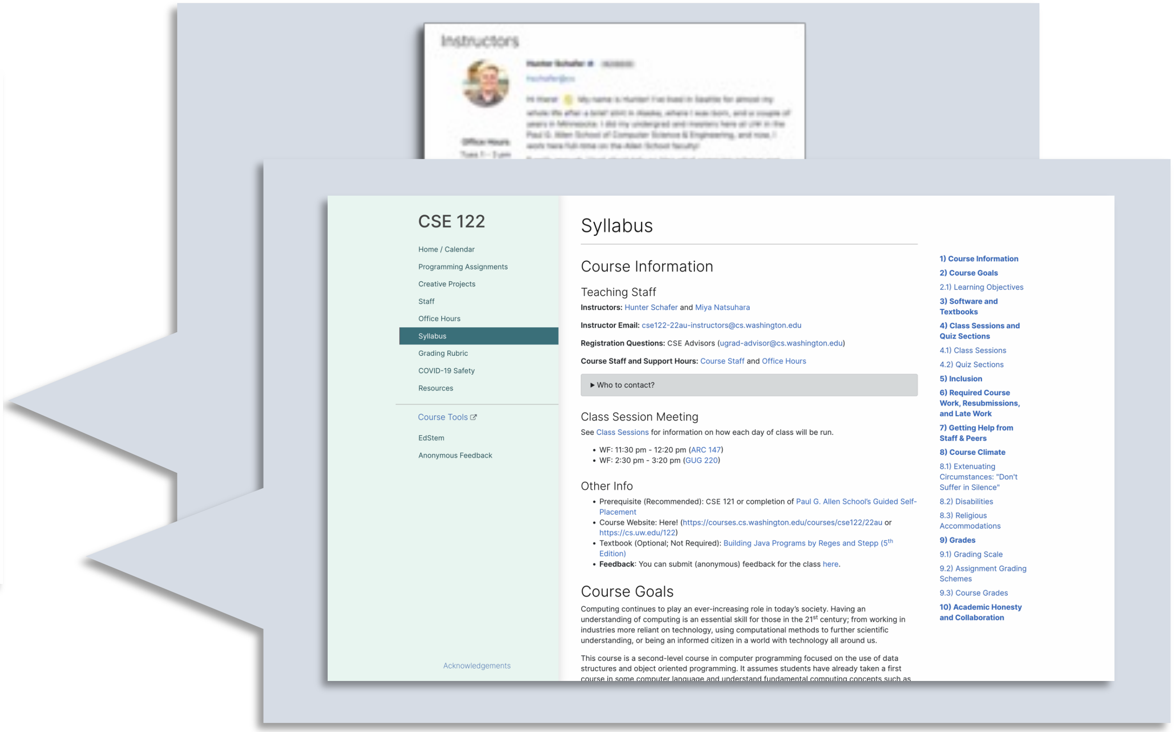
See the full announcement on Ed!

▸ SEP 23 Before Quarter

Calendar

Info This is a rough sketch of the quarter and things are subject to change. We can accurately predict the past, but predicting the future is hard!

	TOPIC	PROJECTS	EXERCISES
Week 1			
Mon 09/28	No Class		
Wed 09/30	LEC 01 Welcome! ADTs		RELEASED EX0 Community Building DUE 11:59 PM
Thu 10/01	SEC 01 CSE 143 Review, Testing	RELEASED P0 CSE 143 Review	
Fri 10/02	LEC 02 Lists		
Week 2			
Mon 10/05	LEC 03 Stacks, Queues, Maps		



Instructors

CSE 122

- Home / Calendar
- Programming Assignments
- Creative Projects
- Staff
- Office Hours
- Syllabus
- Grading Rubric
- COVID-19 Safety
- Resources

Course Tools

- EdStem
- Anonymous Feedback

Acknowledgements

Syllabus

Course Information

Teaching Staff

Instructors: Hunter Schafer and Miya Natsuhara

Instructor Email: cse122-22au-instructors@cs.washington.edu

Registration Questions: CSE Advisors (ugrad-advisor@cs.washington.edu)

Course Staff and Support Hours: Course Staff and Office Hours

▸ Who to contact?

Class Session Meeting

See Class Sessions for information on how each day of class will be run.

- WF: 11:30 pm - 12:20 pm (ARC 147)
- WF: 2:30 pm - 3:20 pm (GUG 220)

Other Info

- Prerequisite (Recommended): CSE 121 or completion of Paul G. Allen School's Guided Self-Placement
- Course Website: Here! (https://courses.cs.washington.edu/courses/cse122/22au or https://cs.uw.edu/122)
- Textbook (Optional; Not Required): Building Java Programs by Reges and Stepp (5th Edition)
- **Feedback**: You can submit (anonymous) feedback for the class here.

Course Goals

Computing continues to play an ever-increasing role in today's society. Having an understanding of computing is an essential skill for those in the 21st century; from working in industries more reliant on technology, using computational methods to further scientific understanding, or being an informed citizen in a world with technology all around us.

This course is a second-level course in computer programming focused on the use of data structures and object oriented programming. It assumes students have already taken a first course in some computer language and understand fundamental computing concepts such as

Contains most course info – check frequently!

- Announcements, Calendar, Lecture Slides, Office Hours schedule, Staff Bios, Important Links

Please familiarize yourself with the course syllabus this week!

Other Course Tools

Ed
- Community & Information
  - Discussion Board
    (please ask & answer!; anonymous option)
  - Chat
  - Announcements
- Pre-Class Materials / Section Handouts
- Assignments
  - Online IDE
  - Submit assignments
  - View Feedback

My Digital Hand
- Queueing in office hours

IntelliJ
- Develop offline
- Visual debugger

Canvas
- Gradebook
- Lecture recordings

Sli.do
- In-class activities (ungraded)
- No account needed

Lecture Outline

- Introductions
- **About this Course**
  - Course Components & Tools
  - **Policies**
  - Making the Most of this Class
- Intro/Review Java

Resubmissions / Retakes

Learning is a challenging process that takes time, it doesn't always happen on your first try.

- Each week, one previous Programming Assignment or Creative Project can be resubmitted
  - Must be accompanied by write up explaining changes
  - Grade on resubmission replaces original grade.
- To stay caught-up with the course, each assignment should only be resubmitted at most once over the quarter.
  - If you find an unforeseen circumstance that requires you to use more than one resub for a particular assignment, you need to discuss with your TA a plan to stay caught-up in order before we can accommodate extra resubs.
- Each quiz can be retaken at most once

See syllabus for more details

Collaboration

- These concepts are challenging: 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.
  - For any ungraded work (e.g., pre-class materials) there is no concern about academic misconduct! You should be collaborating on those without reservation.
  - On graded assignments you should still collaborate, but the code you write should be of your own creation.
  - Always cite the help you receive on graded work
- Withdrawal Policy
- **Read full policy in Syllabus**

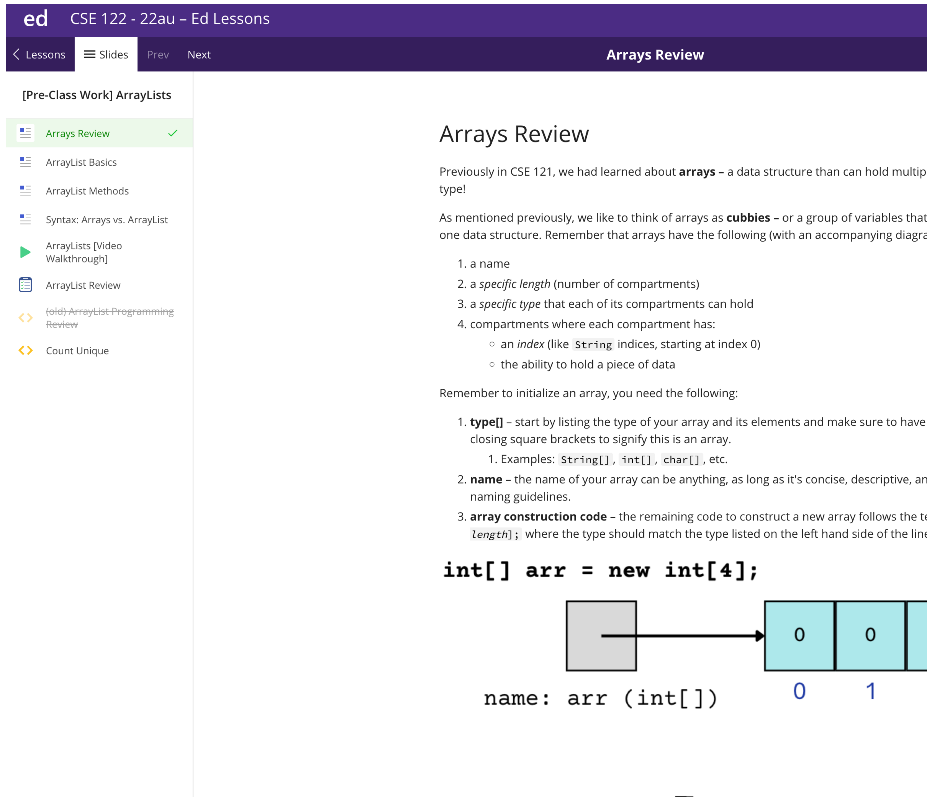
Textbook

Pre-class Materials

- All required readings are available free on Ed!
- Should be finished before class (not graded)

Optional Textbook

- [Building Java Programs by Reges and Stepp (5th Edition)]
- Not required but does add another perspective. Will reference relevant chapters.
- Advice: only purchase if you learn best with a textbook, otherwise not recommended.


ed
CSE 122 - 22au – Ed Lessons
Lessons
Slides
Prev
Next
Arrays Review
[Pre-Class Work] ArrayLists
Arrays Review ✓
ArrayList Basics
ArrayList Methods
Syntax: Arrays vs. ArrayList
ArrayLists [Video Walkthrough]
ArrayList Review
~~(old) ArrayList Programming Review~~
Count Unique

Arrays Review

Previously in CSE 121, we had learned about **arrays –** a data structure than can hold multip
type!

As mentioned previously, we like to think of arrays as **cubbies –** or a group of variables that
one data structure. Remember that arrays have the following (with an accompanying diagra

1. a name
2. a *specific length* (number of compartments)
3. a *specific type* that each of its compartments can hold
4. compartments where each compartment has:
   - an *index* (like `String` indices, starting at index 0)
   - the ability to hold a piece of data

Remember to initialize an array, you need the following:

1. **type[]** – start by listing the type of your array and its elements and make sure to have closing square brackets to signify this is an array.
   1. Examples: `String[]`, `int[]`, `char[]`, etc.
2. **name** – the name of your array can be anything, as long as it's concise, descriptive, an
   naming guidelines.
3. **array construction code** – the remaining code to construct a new array follows the te
   `length];` where the type should match the type listed on the left hand side of the line

```
int[] arr = new int[4];
```

Lecture Outline

- Introductions
- **About this Course**
  - Course Components & Tools
  - Policies
  - **Making the Most of this Class**
- Intro/Review Java

How Learning Works

- Learning requires **active participation** in the process. It's not as simple as sitting and listening to someone talk at you.
  - Requires **deliberate practice** in **learning by doing**
  - Benefits from **collaborative learning**
- Hybrid classroom model
  - Asks you to do some preparation before class in the form of readings and practice problems.
    - Should take ~30 minutes a day
  - Class will start with brief recap, then pick up where the reading and practice problems leave off.
  - Attendance isn't graded, but showing up and trying is the first step in succeeding in the class!
- Pre-class materials are ungraded, but
  - It's okay if you find them challenging! That means you are learning!

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?

Getting Help

- Discussion Board
  - Feel free to make a public or private post on Ed
  - We encourage you to answer other peoples' questions! A great way to learn
- Introductory Programming Lab (Office Hours)
  - TAs can help you face to face in office hours, and look at your code
  - You can go to the IPL with **any** course questions, not just assignments
- 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 Ed are great places to do that
- Email
  - We prefer that all content and logistic questions go on the Ed discussion board (even if you make them private). 503 of you >>> 33 of us!
  - For serious personal circumstances, you can email Hunter/Miya directly. It never hurts to email us, but if it's a common logistic question, we will politely tell you to post on the discussion board.

    cse122-22au-instructors@cs.washington.edu

Help Us Improve!

- This is a brand-new course! We are always looking for feedback on how to improve the class for you and for future students! Thank you in advance for your patience and understanding as we develop everything. ☺
  - 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!

- Post on the discussion board (can be public/private).
  - Note: Anonymous here is anonymous to other students, not to the staff.

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

The World Around CSE 122

- Our goal is to give you a great CSE 122 experience
  - But CSE 122 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: ability to resubmit assignments and retake quizzes
  - 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
- **Intro/Review Java**

Hello World

- Java Specifics
  - Every program needs a `class`
  - Runnable programs need a `main` method (*signature* must exactly match)
  - `System.out.println` to print
  - `"Hello world"` is a `String`

```java
public class HelloDemo {
    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```

- Running on Ed
  - **Run** runs your program
  - **Mark** submits and runs autograder
    - Submit as many times as you like
    - “Shotgun submission” = Unhelpful habit
  - **Solution** shows solution (if applicable)

Review Java Syntax

Java Tutorial reviews all the relevant programming features you should familiar with (even if you don't know them in Java).

- Printing and comments
- Variables, types, expressions
- Conditionals (if/else if/ else)
- Loops (for and while)
- Strings
- Methods
- File I/O
- Arrays

Practice: Think

sli.do #cse-122

In-Class Activities

- **Goal**: Get you actively participating in your learning
- Typical Activity
  - Question is posed
  - **Think** (1 min): Think about the question on your own
  - **Pair** (2 min): Talk with your neighbor to discuss question
    - If you arrive at different conclusions, discuss your logic and figure out why you differ!
    - If you arrived at the same conclusion, discuss why the other answers might be wrong!
  - **Share** (1 min): We discuss the conclusions as a class
- During each of the **Think** and **Pair** stages, you will respond to the question via a sli.do poll
  - Not worth any points, just here to help you learn!

Practice: Think

sli.do #cse-122

What is the output of this Java program?

```java
public class Demo {
  public static void main(String[] args) {
    int[] nums = {2, 3, 5, 9, 14};

    int totalDiff = 0;
    for (int i = 1; i <= nums.length; i++) {
      totalDiff += (nums[i] - nums[i - 1]);
    }
    System.out.println("Total Diff = ” + totalDiff);
  }
}
```

A) Total Diff = 12
B) Total Diff = 11
C) Total Diff = 7
D) Error

Practice: Pair

sli.do #cse-122

What is the output of this Java program?

```
public class Demo {
  public static void main(String[] args) {
    int[] nums = {2, 3, 5, 9, 14};

    int totalDiff = 0;
    for (int i = 1; i <= nums.length; i++) {
      totalDiff += (nums[i] – nums[i – 1]);
    }
    System.out.println("Total Diff = ” + totalDiff);
  }
}
```

A) Total Diff = 12

B) Total Diff = 11

C) Total Diff = 7

D) Error

“Homework” for Next Time

- First assignment will be released Friday, but there are some things to do in the mean time.

- TODO this week
  - Fill out the introductory survey
  - Post an introduction video on your sections Ed thread! 😊
  - Go meet your TA and classmates in Thursday’s quiz section
  - ⭐ Complete the pre-class material for Friday (see calendar)
  - Check over syllabus details