

LEC 00

CSE 122

Welcome!



Questions during Class?

Raise hand or send here

sli.do #cse122



BEFORE WE START

Talk to your neighbors:
Introduce yourself to your neighbor!

*What is your name? Year? What did
you do over summer break?*

Instructors: Ivory Wang & Colin Lim

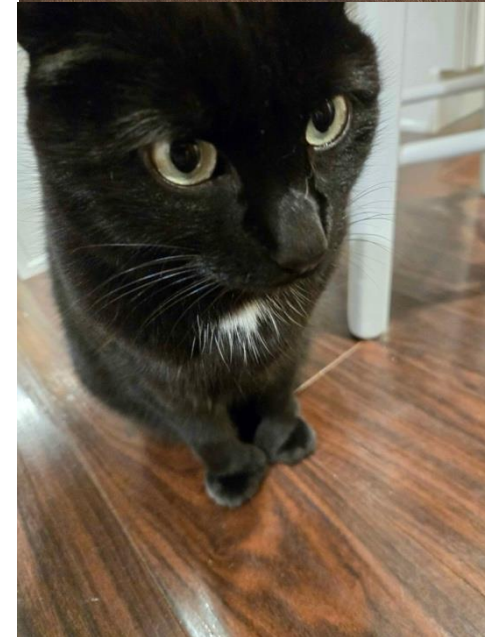
TAs: Dani
Rohan
Wesley
Connor
Yang

Lecture Outline (1/6)

- **Introductions** 
- About this Course
 - Course Components & Tools
 - Grading
 - Policies
- Intro/Review Java
- In-Class Activities
- Functional Decomposition
- Code Quality

Course Staff

- Instructors: Ivory Wang, Colin Lim
- Call us: Ivy/Ivory & Colin
- Teaching Assistants: 5 Fantastic TAs!
 - Available in section, office hours, and discussion board
 - 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 35 students registered for the course!
- Strength in numbers
 - With 35 students, if you're confused about something, we guarantee someone else is too! Ask questions in Slido or in class 
 - 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.

CSE 12x Behavioral Expectations

- Be professional towards:
 - Fellow Students
 - TAs
 - Instructor
- Ask questions, help others, be a good 12X citizen!
- Treat everyone as you wish to be treated.
- You are in college—you have college-level responsibilities.

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
- Arrays & 2D arrays
- **Computational Thinking**
(language agnostic)

CSE 122 – Computer Programming II

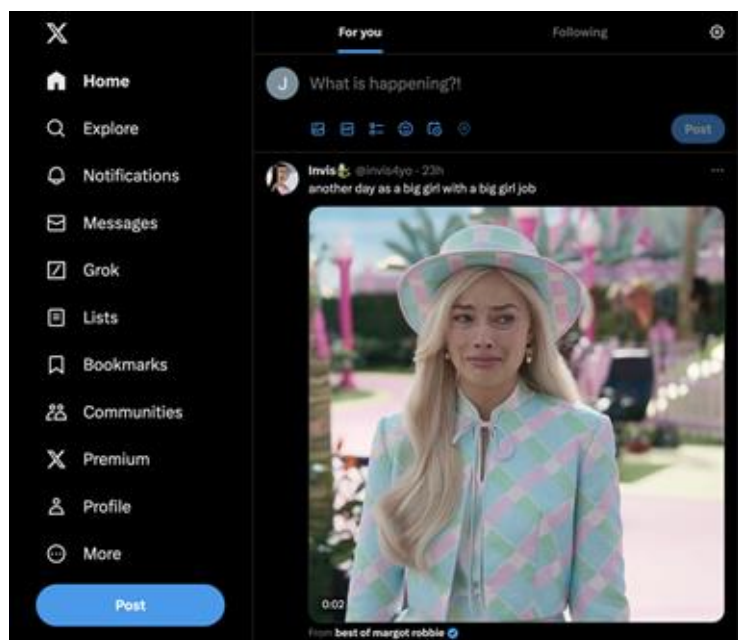
- Decomposing large problems into smaller, manageable subproblems
- File I/O
- Using data structures
 - List
 - Stacks / Queues
 - Sets
 - Maps
- Object Oriented Programming
 - Interfaces

Prerequisite Knowledge

- Students entering CSE 122 are coming from many of different backgrounds
 - UW: CSE 121 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 Creative Project 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? (1/2)

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



122 Data Structures



Why 122? (2/2)

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 (2/6)

- Introductions
- **About this Course** 
 - **Course Components & Tools**
 - **Grading**
 - **Policies**
- Intro/Review Java
- In-Class Activities
- Functional Decomposition
- Code Quality

Course Components

Meetings

LECTURES

(x14)

- Introduce concepts, practice ideas, discuss applications.
- Pre-class materials to prepare for class each day. Due **before** class.
- Recorded 🎥

SECTIONS

(x13)

- More practice, reviews, applications
- TA advice, how to be an effective student
- Preparation for quizzes / exams
- Incentives to attend this quarter!
- 🚨 Not Recorded! 🚨

Assessments

PROGRAMMING ASSIGNMENTS

(x4)

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

CREATIVE PROJECTS

(x3)

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

QUIZZES

(x3)

- Taken in quiz section
- 55 minutes on paper

EXAM

(x1)

- Culminating exam
- **August 19 & 21, 2026 (both days required!)**

Resubmissions

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.
 - An assignment can be resubmitted in the 3 cycles after feedback has been published
 - *Tip: Resubmit as early as possible*

See [syllabus](#) for more details

Section Credit

- Students receive "section credit" for a quiz section by attending and engaging in the quiz section's class and activities.
- Section credit is logged and tracked by TAs on behalf of students.
- Students will be able track and verify their participation credits [on Gradescope](#).
- Section 0 will be a freebie!

Section Credit

- Students can earn **one extra resubmission** at the end of the quarter by participating in 8+ sections throughout 26su!
- Missing a quiz section will not count against you, but attending section can help you earn that extra resubmission to use at the end of the quarter.

Graded Course Components

- Each mark is graded on the scale:
 - **E**(xcellent)
 - **S**(atisfactory)
 - **N**(ot yet)
- Your grade will consist of the following categories:

Category	#	Marks per	Total Marks
Programming Assignments	4	4 (Behavior, Concepts, Quality, Testing/Reflection)	16
Creative Projects	3	4 (Behavior, Concepts, Quality, Testing/Reflection)	12
Quizzes	3	6 (3 questions)	18
Exam	1	12 (6 questions)	12

Course Grades

In assigning course grades, we'll use a bucket system:

- Marks earned place in an initial bucket, additional S+ marks improve grade.
- Must meet all requirements of a bucket for initial placement.
- These are minimum GPA guarantees – grade can always be higher than minimum promise.

Minimum Grade	Assignments	Quiz/Exam Problems
3.5	22 E AND 2 additional S+ AND no Us	20 E AND 2 additional S+ AND no Us
3.0	18 E AND 4 additional S+ AND no Us	16 E AND 4 additional S+ AND no Us
2.5	14 E AND 6 additional S+ AND no Us	12 E AND 6 additional S+ AND no Us
2.0	18 S+	16 S+
1.5	11 S+	9 S+

S+ indicates S or E

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). 35 of you >>> 7 of us!
 - For serious personal circumstances, you can email us directly. It never hurts to email us, but if it's a common logistic question, we may politely ask you to post on the discussion board instead.

Collaboration & Resources

- 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.
 - **Submitted work must be done completely without LLM/AI tool help.**
 - Be aware of and avoid use of Forbidden Features in submitted work
 - Always cite the help you receive on graded work
- **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 - Ed Lessons

Arrays Review

Pre-Class Materials 2: ArrayLists

- Arrays Review ✓
- ArrayList Basics
- ArrayList Methods
- Syntax: Arrays vs. ArrayList
- ArrayLists (Video Walkthrough)
- ArrayList Review
- Builds Arraylist Programming Review
- Count Unique

Arrays Review

On the left hand side, you'll see there's a lesson titled **ArrayLists (Video Walkthrough)**. The video and the reading both have the same information! You're not required to go through both the video and the reading, as the video just walks through the reading to help contextualize it!

Previously in CSE 121, we had learned about **arrays** – a data structure than can hold multiple values of the same type!

As mentioned previously, we like to think of arrays as **cubbies** – or a group of variables that are stored together in one data structure. Remember that arrays have the following (with an accompanying diagram below):

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

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 the opening and 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, and follows prescribed naming guidelines.
3. **array construction code** – the remaining code to construct a new array follows the template: `new type[ int length ]`; where the type should match the type listed on the left hand side of the line of code.

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

name: arr (int[]) 0 1 2 3

Course Website (1/2)

cs.uw.edu/122

CSE 122

[Home / Calendar](#)

[Syllabus](#)

[Assignments](#)

[Resubmissions](#)

[Exam](#)

[Staff](#)

[Office Hours](#)

[Grading Rubrics](#)

[Resources](#)

[Course Tools](#) [↗](#)

[EdStem](#)

[Anonymous Feedback](#)

[Code Quality Guide](#)

[Commenting Guide](#)

[Acknowledgements](#)

Introduction to Computer Programming II Summer 2026

Welcome to CSE 122: Introduction to Computer Programming II 🎉

► What is this class? What will I learn?

► Prior Experience and Expectations

Registration Please **do not** email the course staff or instructors regarding registration for the course. The do not have access to add codes. Please email ugrad-adviser@cs.washington.edu for

This Week (at a glance)

Monday (06/22)

- Nothing!

Tuesday (06/23)

- Nothing!

Wednesday (06/24)

Instructor



Colin Lim HE/HIM/HIS

colinlim@cs

Hi everyone! I'm Colin a recent Computer Science graduate. In my free time I like playing games, cooking, hiking, and visiting cat cafes! If you have any pet pictures you wanna share please don't hesitate to hunt me down and show them to me - I'll greatly appreciate them :)). Looking forward to meeting you all!

Office Hours
TBD



Ivory Wang

iywang@cs

Hello everyone! My name is Ivy and I recently graduated from UW after three fun years of computer science, math, and statistics. Outside of school and work, I like to haunt the IMA dance studios, pet my cats (either in real life or in Stardew Valley), and introspect to one of my many music playlists. Looking forward to a super educational & super interesting quarter with you all!

Office Hours
TBD

Get to know the course staff

Contains most course info – check frequently!

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

Course Website (2/2)

cs.uw.edu/122

CSE 122

- Home / Calendar
- Syllabus
- Assignments
- Resubmissions
- Exam
- Staff
- Office Hours
- Grading Rubrics
- Resources

Course Tools [↗](#)

- EdStem
- Anonymous Feedback
- Code Quality Guide
- Commenting Guide

[Acknowledgements](#)

Introduction to Computer Programming II

Summer 2026

Welcome to CSE 122: Introduction to Computer Programming II 🎉

- ▶ What is this class? What will I learn?
- ▶ Prior Experience and Expectations

Registration Please **do not** email the course staff or instructors regarding registration for the course. The do not have access to add codes. Please email ugrad-adviser@cs.washington.edu for

This Week (at a glance)

- Monday (06/22)**
 - Nothing!
- Tuesday (06/23)**
 - Nothing!
- Wednesday (06/24)**

Instructor



Colin Lim HE/HIM/HIS
colinlim@cs

Syllabus

Course Information

Teaching Staff

Instructor: Ivory Wang and Colin Lim

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

Registration Questions: CSE Advisers (ugrad-adviser@cs.washington.edu)

Course Staff and Support Hours: [Course Staff](#) and [Office Hours](#)

▼ Who to contact?

Please familiarize yourself with the course syllabus this week!

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

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

My Digital Hand

- Queueing in office hours



VSCode (Optional)

- Develop offline
- Visual debugger



Canvas

- Lecture recordings



Sli.do

- In-class activities
(ungraded)
- No account needed

Lecture Outline (3/6)

- Introductions
- About this Course
 - Course Components & Tools
 - Grading
 - Policies
- **Intro/Review Java** ◀
- In-Class Activities
- Functional Decomposition
- Code Quality

Review Java Syntax

[Java Tutorial](#) reviews all the relevant programming features you should be 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
- Arrays & 2D Arrays

Lecture Outline (4/6)

- Introductions
- About this Course
 - Course Components & Tools
 - Grading
 - Policies
- Intro/Review Java
- **In-Class Activities** 
- Functional Decomposition
- Code Quality



Practice: Think

[sli.do](#)

#cse122

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](#) [#cse122](#)

What is the output of this Java program?

```
public class Demo {  
    public static void main(String[] args) {  
        int[] nums = {1, 4, 4, 8, 13};  
  
        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 = 10
- C) Total Diff = 9
- D) Exception!



Practice: Pair



sli.do #cse122

What is the output of this Java program?

```
public class Demo {  
    public static void main(String[] args) {  
        int[] nums = {1, 4, 4, 8, 13};  
  
        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 = 10
- C) Total Diff = 9
- D) Exception!

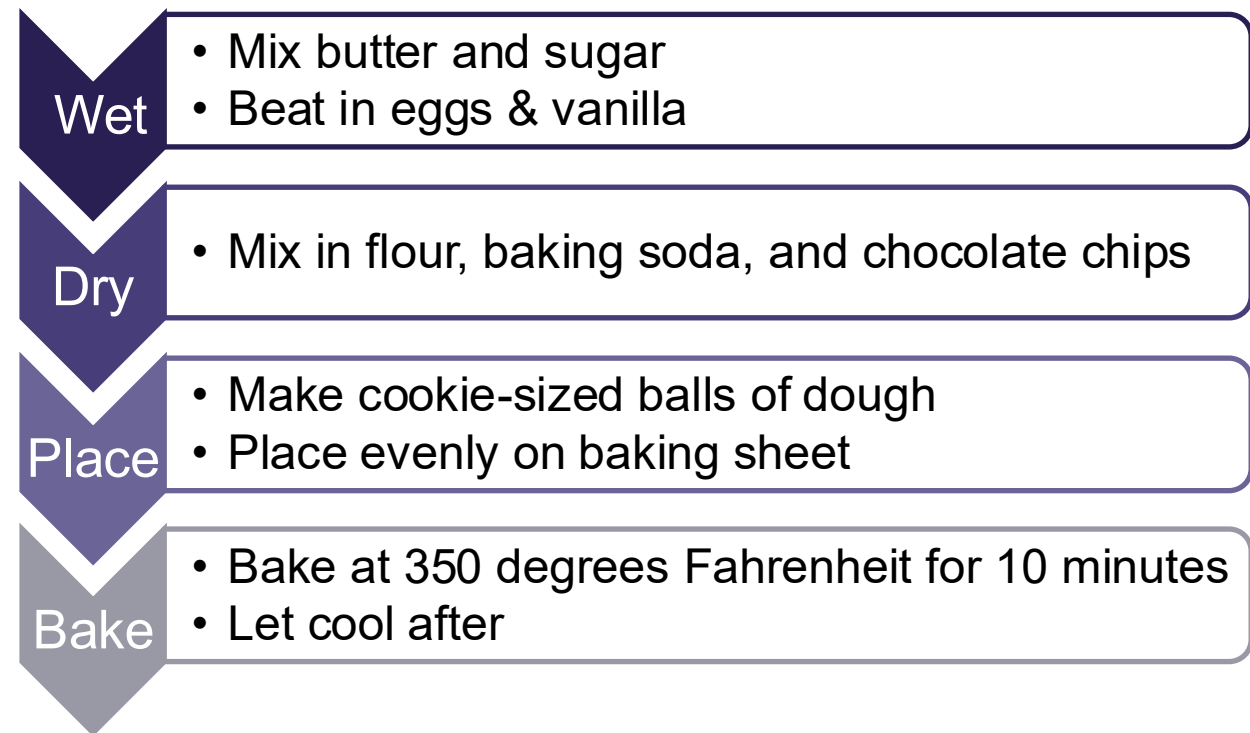
Lecture Outline (5/6)

- Introductions
- About this Course
 - Course Components & Tools
 - Grading
 - Policies
- Intro/Review Java
- In-Class Activities
- **Functional Decomposition** ◀
- Code Quality

Functional Decomposition

Functional decomposition is the process of breaking down a complex problem or system into parts that are easier to *conceive, understand, program, and maintain*.

“Bake the cookies” →



Functional Decomposition

In our code, functional decomposition often means breaking a task into smaller methods (also called functions).

Example: Bingo

- Set up populated game card
- Roll and call a new number
- Mark your card if the number called is present
- Check card for bingo

Avoid Trivial Methods

Introduce methods to decompose a complex problem, not just for the sake of adding a method.

Bad example:

```
public static void printMessage(String message) {  
    System.out.println(message);  
    System.out.println();  
}
```

Good Example:

```
public static void sToQ(Stack<String> s, Queue<String> q) {  
    while (!s.isEmpty()) {  
        q.add(s.pop());  
    }  
}
```

Rule of thumb: A method should do at least two steps

- Ask yourself: Does adding this method make my code easier to understand?

Lecture Outline (6/6)

- Introductions
- About this Course
 - Course Components & Tools
 - Grading
 - Policies
- Intro/Review Java
- In-Class Activities
- Functional Decomposition
- **Code Quality** ◀

Code Quality

“Programs are meant to be read by humans and only incidentally for computers to execute.” – Abelson & Sussman, SICP

Code is about communication. Writing code with good **code quality** is important to communicate effectively.

Different organizations have different *standards* for code quality.

- Doesn't mean any one standard is wrong! (e.g., APA, MLA, Chicago, IEEE, ...)
- Consistency is very helpful within a group project
- See our [Code Quality Guide](#) for the standards we will all use in CSE 122

CSE 122 Code Quality

Examples relevant for this week

- Naming conventions
- Descriptive variable names
- Indentation
- Long lines
- Spacing
- Good method decomposition
- Writing documentation

“Homework” for Next Time

- First assignment will be released Friday, but there are some things to do in the meantime.
- TODO this week
 - [Fill out the introductory survey](#)
 - Go meet your TA and classmates in Thursday’s quiz section
 - 🌟 Complete the pre-class material for Friday (see calendar)
 - [Check over syllabus details](#)