CSCI 261
Computer Science II
About Me

- David Chiu (pronounced "chew")
  - At Puget Sound since 2014

Teaching at Puget Sound

- CS 161 Introduction to Computer Science
- CS 261 Computer Science II
- CS 440 Capstone in Computer Science
- CS 455 Principles of Database Systems
- CS 475 Operating Systems

Dreese Labs,
Ohio State University
Reaching Me

- In order of preference:
  - Office: Thompson Hall 390 B
  - Slack: univpugetsound.slack.com
    - Signup with your pugetsound email
  - Email: dchiu@pugetsound.edu
    - Will not respond to emails addressed to variations of "Hey"

- Office Hours: Open-Door Policy
  - When door is cracked/open: Available
  - Best: Email me and make an appointment
Course Requirements

- **Textbook:** “K&W”

- **Assumptions**
  - You have a firm grasp of:
    - Classes/objects, object interaction, loops, arrays and ArrayLists, user input, static methods including main()
  - Great if you’ve had exposure to:
    - Simple recursive algorithms, file-input handling, exceptions
    - Basic sorting algorithms, running-time analysis
Course Goals

- What will be taught?
  - Advanced topics in OOP design
    - Inheritance, polymorphism, interfaces
    - Exception handling
  - Elementary complexity analysis
    - Big-O notation
  - Data structures and their operations
    - Lists, queues, stacks, maps
    - Binary trees, BSTs
  - Recursion
    - Recursive algorithms and data structures
    - Classical sorting algorithms
Course Organization

- Important! Bookmark these links.

- Course Web Page
  - [http://cs.pugetsound.edu/~dchiu/cs261](http://cs.pugetsound.edu/~dchiu/cs261)
  - Weekly schedule, labs, homework assignments, code examples, etc.

- Assignment and Lab Submission page:
  - [https://canvas.pugetsound.edu](https://canvas.pugetsound.edu)
Grading

Breakdown

• 10% Weekly Lab Assignments
• 38% Programming Assignments
• 15% Midterm Exam I
• 15% Midterm Exam II
• 20% Final Exam
• 2% Discretionary
Grading: Labs

• Meet weekly (Tuesday) for lab
  • 10 = complete, 5 = some effort, 0 = no attendance/effort/submission

• Policies
  • Work on the lab (not on your writing assignment)
    - Unless you’re done with lab
  • Paired-programming is **required**
    - Expected to collaborate and switch roles
    - Due by the end of lab
  • Leaving early
    - Lab must be completed and graded before you go
Grading: Homework Assignments

- **Homework Assignments**
  - Difficulty increases quadratically
    - Letter Counter
    - Tic Tac Toe
    - Singly Linked List
    - Bank Simulation
    - Expression Trees
    - Huffman Encoding (Compression)

- **With exception to few, most assignments are team-based (3 students max)**
  - Randomly selected each time
  - "Why? I like working by myself."
    - Too bad. Prepares for real-world
Homework Assignments (Cont.)

- Teamwork - What works:
  - Exchange contacts immediately upon learning who's in your group
  - Carve out a regular meeting time and meet physically
  - Write your code together *uno animo*
  - Make sure everyone is on the same page at all times

- What doesn't work:
  - Split the assignment into tasks for each student
  - Each writes code separately and sync up later (good luck!)
    - Lack of regular check-ins and communication
I have no preference on which Java IDE you choose, but here are the popular ones:

• I still use BlueJ for lectures, and IntelliJ or eclipse in my free time

Not recommended:

• Text editors like Sublime, Atom, vim, emacs, notepad, textpad, ...
  - Need to compile "by hand" on the command-line; no "niceties" like autocomplete
  - No it does not make you look more like a hacker
Grading: Exams

- Two midterms and final will be based on:
  - Labs, homework assignments, lectures, readings
  - Comprehensive, but weighted predominantly on new materials
  - Allowed: Calculator and full page of notes (front & back okay)

- Study guide available on webpage and review the day before

- Midterm exams will take place during lab days (Tuesday)
  - Midterm exams are 50 minutes long
Classroom Participation

- Class should be interactive
  - Ask questions when you don't understand something
    - If questions are irrelevant to lecture, I'll be happy to answer it after class
  - I will pose lots questions each lecture
  - I will do lots of small-group work, when time permits

- Be careful how you present yourself
  - For many here, CS is completely new.
  - Try not to act condescending to others if you have a knack for this.
Things That Make Me Grumpy

- **Class Disruption**
  - Put phone on silent
  - No laptops, unless:
    - Lab day (Tues)
    - Permitted for via SAA

- **Cheating: Zero-Tolerance**
  - I compare current and past assignments
  - Rules of thumb on collaboration:
    - **Can do:** Look on another student's screen to help *them*
    - **Can't do:** Look on another student's screen to help *yourself*
Note-Taking for This Class

- Slides are always available on the course web page
  - After the lecture is over

- I do lots of code writing
  - In my Java editor
    - No need to copy! I'll provide the code we write in BlueJ in class
    - Just try to follow along, and understand the thinking behind the code
  - On the board (these are usually impromptu)
    - Write these down! You won't see these again

- I do lots of board-work: drawing/traces/illustrations
  - Copy these down
CS tutors are available in TH 420:
  • Tutors are upper-level CS majors with outstanding grades
  • Everyday except Friday and Saturday
    • 4-6pm

No appointments, just walk-in
  • Two tutors

Go here to see schedule:
  • http://www.pugetsound.edu/academics/departments-and-programs/
    undergraduate/math-and-computer-science/tutoring/
Most Important Thing: Backup

- Back up your code
  - Unlike CS 161, in this class:
    Lost code does not get you homework extension

- Easiest way:
  - Get a free Dropbox or Google Drive account
    - Set it to sync automatically
  - When your Java IDE asks where to place project folders
    - Always put them in the dropbox or drive folder
  - *I can help you set up this environment in lab today*
If you have a physical, psychological, medical, or learning disability, contact:

• Peggy Perno
• Office of Student Accessibility and Accommodation (SAA)
• 105 Howarth Hall
• 253.879.3395

Communications with SAA is confidential
Outline

- Course Syllabus
- Our First Lab
  - Post-Mortem and Motivation
- Inheritance
  - Mechanism
  - Protected Access
  - `super()`, `this()`
- Conclusion
Our First Lab: Around the Farm

- The first lab
  - Work alone on this lab, as it is review.
  - Use any Java editor you wish

- Find the first lab on the course page
  - http://cs.pugetsound.edu/~dchiu/cs261
Lab 1 Post-Mortem

- The **Class Diagram** of your project looks something like this:

```
Client class

Farm
- ArrayList<Cow> cows
- ArrayList<Bird> birds
  + addCow(Cow c)
  + addBird(Bird b)
  + exciteAll()
  + summary()

Cow
- String name
- String quote
- double gallonsMilked
  + display()
  + getName()
  + speak()
  + milk()

Bird
- String name
- String quote
- int eggs
  + display()
  + getName()
  + speak()
  + layEgg()

Farm <<uses>>
Bird <<uses>>

UML (Unified Modeling Language) diagram notation:
- denotes private access
+ denotes public access
Dotted line denotes the "uses" or "has-a" relation
```
The *Class Diagram* of your project probably looks something like this:

- Though you coded it properly, you should've noticed several annoying problems. *Let's walk through them.*
  - Let’s bring up Lab 1’s solution
  - [Code demo]
Problem 1: Code Duplication

- **Problem 1: Code Duplication**

  - In the **Cow** and **Bird** classes:
    - Some instance variables (fields) are **redundant** *(circle them on handout)*
    - Some methods are **redundant** *(circle them on handout)*

  - In the **Farm** class:
    - Are there redundancies here?
    - *(circle these on handout too)*
Problem 1: Code Duplication (Answers)

Problem 1: Code Duplication

• In the **Cow** and **Bird** classes:
  - Some instance variables (fields) are **redundant**:
    - *name, quote*
  - Some methods are **redundant**:
    - getName(), speak(),
    - display() — sort of

• In the **Farm** class:
  - Need an ArrayList for Cows, an ArrayList for Birds
  - Need an "add" method for each type of animal: **addCow()**, **addBird()**
  - **summary()** and **exciteAll()** must loop through all cows and birds separately
Problem 2: Code Maintenance

- **Problem 2: Hard to maintain (update) code**
  - Say we now want to track the weight of all animals
    - No easy way: need to add a field and setter/getter methods to every animal class
    - Then need to update `display()` in Cow and Bird to include weight

![Class Diagram]

**Legend:**
- Added code
- Code needs updated
Administrivia 9/3

- Reading:
  - Chap 1 of K&W, or/and
  - Chap 10 of the BlueJ book if you still have it

- Homework 1 is posted (GET STARTED - people struggle with it)
  - Due Fri, 9/13
  - Do this one alone; can be completed based on CSCI 161 knowledge

- Lab 1 solution posted early for today's lecture
  - BetterFarm code is posted (We’ll write it this week)