

# Computer Science Principles

CSE 120 Winter 2018

## Instructor:

Justin Hsia

## Teaching Assistants:

Anupam Gupta, Cheng Ni,  
Sam Wolfson, Sophie Tian,

Eugene Oh,  
Teagan Horkan

## Ten years ago, Amazon changed Seattle, announcing its move to South Lake Union

“[Dec. 21, 2017] marks 10 years since Amazon committed to building an 11-building campus in South Lake Union. But before the company locked up that space, Amazon's landlord went to the city seeking a deal, sparking debates about the impact of development that have grown louder a decade on.”

- <https://www.seattletimes.com/business/amazon/ten-years-ago-amazon-changed-seattle-announcing-its-move-to-south-lake-union/>

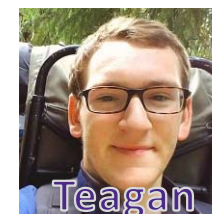




# Who: Course Staff

- ❖ Your Instructor: just call me Justin
  - From California (UC Berkeley and the Bay Area)
  - I like: teaching, the outdoors, board games, and ultimate
  - Excited to be teaching CSP again at UW!

- ❖ 6 TAs:



- Available during lab, in office hours, and on Piazza
  - An invaluable source of information and assistance
- ❖ Get to know us
  - We are here to help you succeed and improve your experience

# Who: You!

- ❖ 61 students registered
  - Undergrads from *many* different majors (or pre-majors)
  - This class is intended for students without significant previous experience with computing/programming
- ❖ Get to know each other and help each other out!
  - Learning is much more fun with friends
  - Working well with others is a valuable life skill
  - Diversity of perspectives expands your horizons
- ❖ Submit Pre-Course Survey so we can find out more

# Why Study Computer Science?

- ❖ Massive impact on our lives and society as a whole
- ❖ Increasingly useful for *all* fields of study and areas of employment
  - Creative Writing – word editing, spell check, need to outcompete robots soon!
  - Massage Therapist – massage robots soon?
  - Dancing – stage lighting, motion capture to study technique
  - Ren Faire Actor – analyze data on ticket sales, popular booths

# Computing in Your Future

- ❖ Computing and its data are inescapable
  - You generate “digital footprints” all the time
- ❖ Computing is a regular part of *every* job
  - Use computers and computational tools
  - Generate and process data
  - Dealing with IT and software people
  - Understanding the computational portion of projects
- ❖ Our goal is to help you make sense of the “Digital Age” that we now all live in

# Computing and Society

## ❖ Raise your hand if:

- You know someone who works from home
- You have “stalked” someone online
- You communicate mostly using images instead of words
- You have taken extra trips outside to catch that Pokémon
- You get the majority of your news from social media
- The majority of the media you own (*e.g.* music, movies, books, art, games) is digital
- You know someone who’s had their identity or credit card number stolen online
- You’ve seen a parent quiet a child by giving them a digital device

# What This Course IS

❖ This course is split into two major themes:

## 1) Computational Thinking

- How can you use computers to solve problems
- Using programming as a tool

## 2) Computational Principles

- The “big ideas of computing” that we think everyone should know
- *e.g.* bits can represent anything and everything, what a computer can and can't compute, how do websites and the Internet work, social implications of computing

# What This Course Is NOT

- ❖ Preparation for CSE142: Computer Programming I
  - This is not just a programming course
    - Introduce you to the concepts, but not expect you to master them
  - But great if you feel motivated to continue afterward!
- ❖ Trivial
  - Supposed to be material you haven't seen before
  - A technical class that asks you to read and write and be creative
- ❖ Boring or back-breaking
  - Assignments intended to be fun, interesting, and reasonable

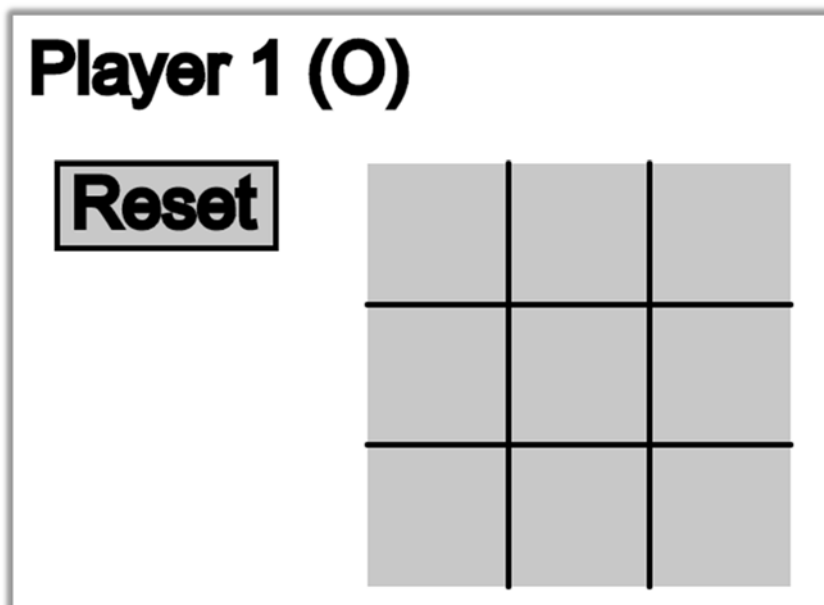
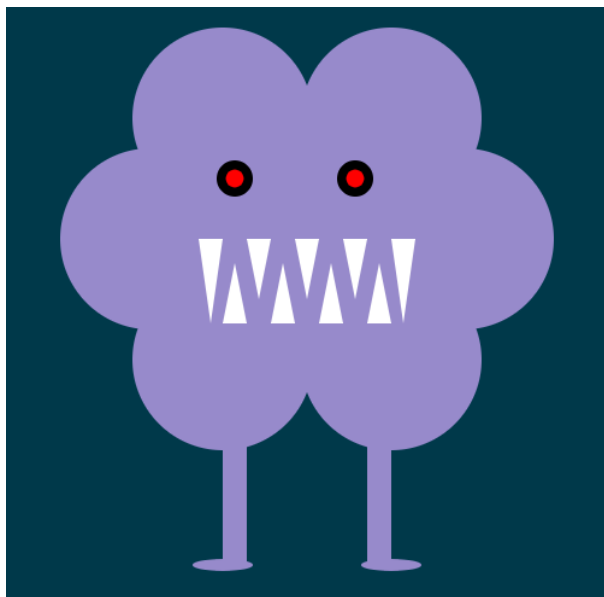


# About Programming

- ❖ **programming  $\neq$  computational thinking**
  - *Computational thinking* is knowing how to break down and solve a problem in a way that a computer can do it
  - *Programming* is the tool you use to execute your solution
  - We use programming in this course as a way of teaching computational thinking
- ❖ Can be learned, just like any other skill
  - It's not black magic; there's no such thing as a "coding gene"
  - Yes, at first it may be challenging and mind-bending – just like learning your first non-native language
  - My hope is that you will think differently after this course

# Programming in CSE120

- ❖ Use a language called **Processing**
  - Text-based language that is good for visuals and interaction
  - We will use Java syntax
  - At the end of the day, the language you use doesn't matter as long as you develop computational thinking skills



# Big Ideas of Computing

- ❖ Exposure to a broad range of topics in computer science
  - Not going to dive into the details
  - These are the motivations & the applications for programming (the tool)
  - Focus on what to be aware of to navigate the digital world
- ❖ **Goal: become “literate” in computing**
  - As new innovations arise, can you read about it, understand its consequences, and form your own opinion?
  - This course will ask you to *read*, *discuss*, and *write* about computing innovations

# Lecture Outline

- ❖ Course Introduction
- ❖ **Course Policies**
  - <http://courses.cs.washington.edu/courses/cse120/18wi/syllabus/#policies>
- ❖ Abstraction

# Communication

- ❖ Website: <http://cs.uw.edu/120>
  - Calendar, schedule, policies, labs, links, assignments, etc.
  - Grade book and assignment submissions via Canvas
- ❖ Discussion: <http://piazzza.com/washington/winter2018/cse120>
  - Ask and answer questions – staff will monitor and contribute
  - *ALL* questions on course material should go here
- ❖ Office Hours: spread throughout the week
  - Can also email to make individual appointments
- ❖ Anonymous feedback form

# Weekly Schedule


- ❖ Lectures are Mon, Wed, Fri (3 hr)
  - Friday lectures will generally be reserved for “Big Ideas”
- ❖ Weekly reading is due before lab on Thursday
  - All readings online, complete “reading check” to prep
- ❖ Labs on Tue, Thu (3 hr)
  - Worksheets and work time with help from TAs
  - 10-15 minutes at start of Thu lab will be spent discussing the weekly reading
- ❖ Can be a demanding schedule, but should be fun!

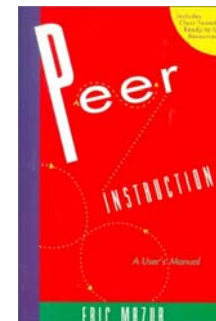
Monday	Tuesday	Wednesday	Thursday	Friday
Lecture	Lab	Lecture	Lab (Reading)	Lecture

# Course Components and Grading

- ❖ **Programming Assignments (40%)**
  - Includes a website portfolio of your work
- ❖ **Final Project (20%)**
  - Of your own choosing!
- ❖ **Written Assignments (15%)**
  - Includes reading checks, field trip report, mini research report
- ❖ **Exams: Midterm (10%) and Final (10%)**
  - Double-check your understanding of concepts
- ❖ **EPA: Effort, Participation, and Altruism (5%)**
  - Encourage class-wide learning

# Peer Instruction

- ❖ Increase real-time learning in lecture, test your understanding, increase student interactions
  - Lots of research supports its effectiveness
- ❖ Multiple choice question at end of lecture “segment”
  - 1 minute to decide on your own
  - 2 minutes in pairs to reach consensus
  - Learn through discussion
- ❖ Vote using  **Poll Everywhere**
  - Use website (<https://www.polleverywhere.com>) or app
  - Linked to your UWNetID





# Some topics that we will touch on

- ❖ Which of the following seems the most interesting to you? (vote at <http://PollEv.com/justinh>)
  - A. How do I program simple game mechanics?
  - B. How does a computer work?
  - C. How has the Internet changed the way we interact with each other?
  - D. What is copyright and how does it apply to digital works
  - E. What is artificial intelligence?
  - F. What will I do if/when my job gets taken over by robots (*i.e.* automated)?

# Hooked on Gadgets

- ❖ Gadgets reduce focus and learning
  - Bursts of info (*e.g.* emails, IMs, etc.) are *addictive*
  - Heavy multitaskers have more trouble focusing and shutting out irrelevant information
    - <http://www.npr.org/2016/04/17/474525392/attention-students-put-your-laptops-away>
  - This applies to all aspects of life, not just lecture
- ❖ NO audio allowed (mute phones & computers)
- ❖ Non-disruptive use okay
  - Stick to side and back seats
  - Stop/move if asked by fellow student

# To-Do List

- ❖ Explore website thoroughly: <http://cs.uw.edu/120>
  - Read through the full course policies!!!
- ❖ Check that you are registered on Piazza, Canvas, and Poll Everywhere
- ❖ **Pre-Course (Introduction) Survey** due tomorrow (1/4)
  - More assignments will be introduced in lab tomorrow

# Lecture Outline

- ❖ Course Introduction

- ❖ Course Policies

- ❖ **Abstraction**

  - abstract art

    - ↳ not depict anything that's real

  - abstract description

    - ↳ "levels" of concepts/details

  - paper abstract

    - ↳ summary, main points

# Complexity and Abstraction

- ❖ Programming is straightforward, as long as your programs are small
  - *Complexity* is our enemy
  - *Abstraction* is the key to conquering complexity
- ❖ **Abstraction** allows us to build general-purpose artifacts
  - **Detail Removal:** Hide unnecessary details from users and designers
  - **Generalization:** Avoid unnecessary repetitive work
- ❖ Learning to reason using the most appropriate abstraction is a key goal of computational thinking

# Abstraction: Detail Removal

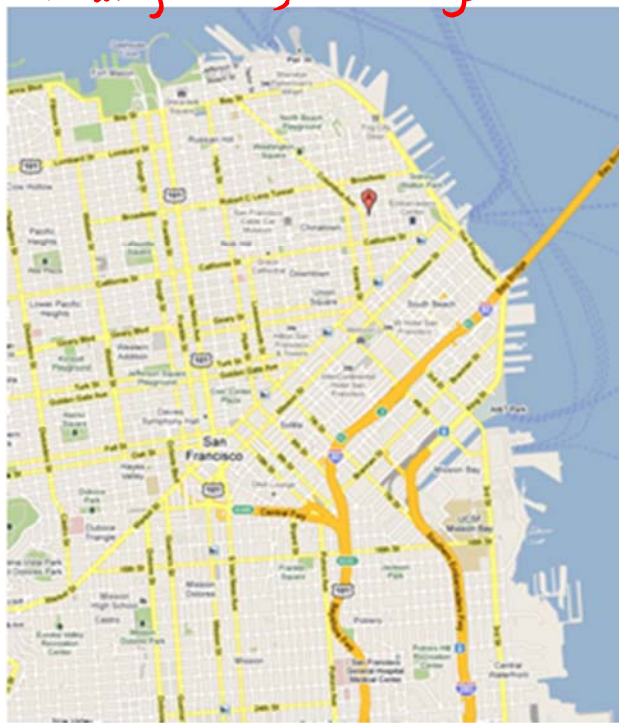
- ❖ “The act or process of leaving out of consideration one or more properties of a complex object so as to attend to others.”

missing person's features



Henri Matisse “Naked Blue IV”

missing trees, buildings



missing unnecessary small streets



Maps for directions

# Abstraction: Detail Removal

## ❖ Detail removal example:

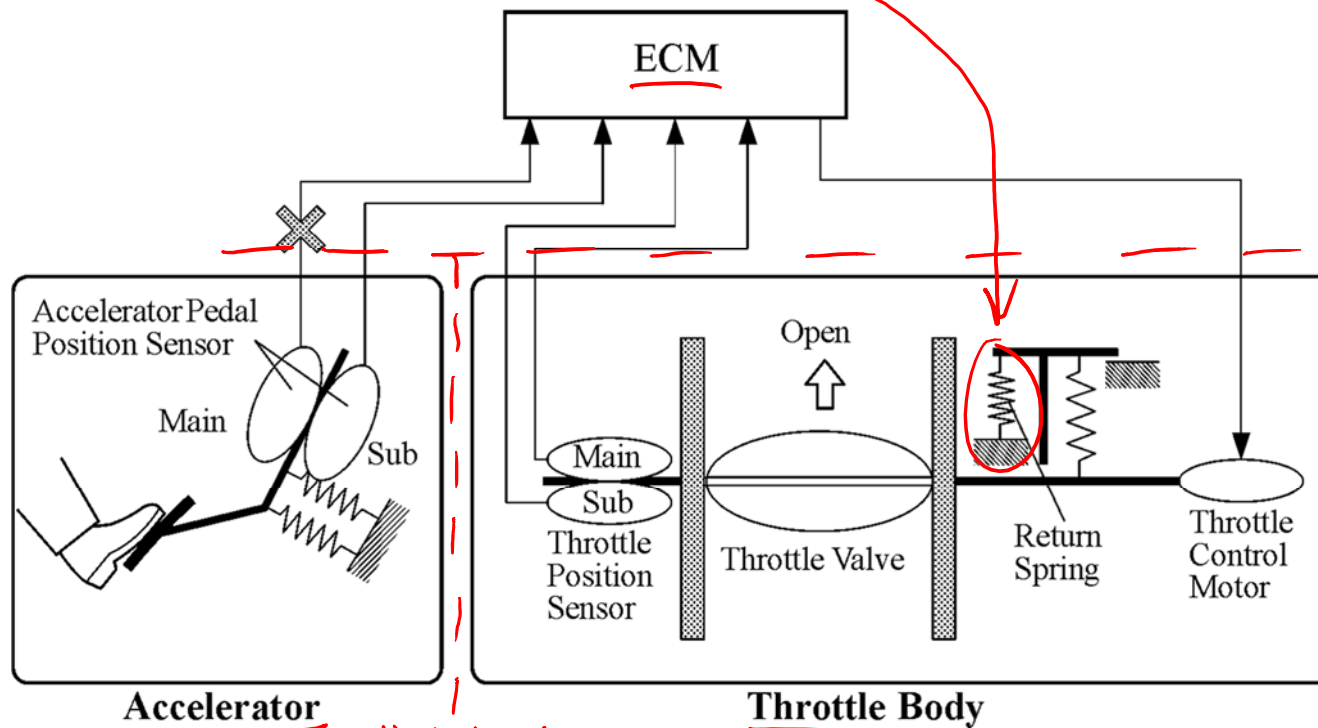
- Modern user interface: Right pedal is “accelerate”, left is “decelerate”
- Even as underlying technology has changed, this abstraction has not!
  - Computer controlled fuel injection
  - Anti-lock brakes (ABS)



# Abstraction: Detail Removal

## ❖ Detail removal example:

- Hide unnecessary details from other designers
  - e.g. Engine Control Module (ECM) designer doesn't care about the return spring inside the Throttle!



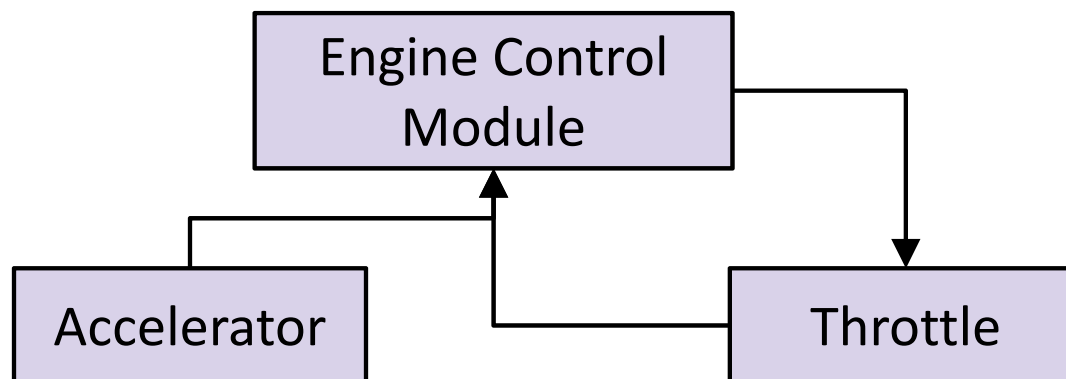
"abstraction barriers": don't need to know details of other "modules"



# Abstraction: Detail Removal

## ❖ Detail removal example:

- Hide unnecessary details from other designers
  - *e.g.* Engine Control Module (ECM) designer doesn't care about the return spring inside the Throttle!
- Nice to be able to think of a system as a hierarchy of well defined “chunks” with precise functionality
  - In CS, we say that we have a **separation of concerns**



# Abstraction: Generalization

- ❖ “The process of formulating general concepts by abstracting common properties of instances.”
  
- ❖ Extensible shower rods
- ❖ Adjustable hats and belts
- ❖ Single recipe for <fruit> cheesecake
- ❖ Feeding animals on a farm
  - To feed <animal>, put <animal> food in <animal> dish

# Audience Responses

## ❖ Other examples of **detail removal**:

- Operating a camera – just push a button
- Sciences – abstracted structures/diagrams
- Simple name for complex phenomena

## ❖ Other examples of **generalization**:

- Apply formula repeatedly (Excel)

# Summary

- ❖ Abstraction is one of the most important challenges in computer science
  - How do you identify the right abstraction you need (block to build) to solve your problem?
  
- ❖ Think about computers:
  - How many of you actually know how a computer works?
  - How many of you can use a computer?
    - Thanks to abstraction!!!