

---

# CSE 331

# Software Design & Implementation

Hal Perkins  
Winter 2021  
Course Wrapup

# Today

---

- Reminder: Fill out your course evaluations (!)
- Reminder: Nominate great TAs for the Bandes award
- Final quiz out Sunday, closes Tuesday night, 7pm
  - Remember to set your clocks ahead Sunday morning for daylight time
- Meet with a TA next week to talk about hw9 – sign up now!
- A look back at CSE 331
  - High-level overview of main ideas and goals
  - Connection to homeworks
  - Context
- Also:
  - Thank-yous

# CSE 331

---

What was it all about?

But first....

# Huge thanks to the folks who made it work

---

Course staff: 16 Amazing TAs:

Hritik Aggarwal, Samuel Berensohn, Ramya Challa, Andrew Gies,  
Elijah Greisz, Kellie Gui, Bryan Lim, CJ Lin, Ardi Madadi,  
Kaushal Mangipudi, Gavin Parpart, Helena Stafford, Yihang Wu,  
David Xu, Yifan Xu, and Hoaran Yu

*This course is itself a sophisticated  
(or at least really, really complicated) system  
requiring savvy design and implementation*

And a big thanks to **you** for all you've done!

---

*4 slides from Lecture 1...*

# What is the goal of CSE 331?

---

How to build harder-to-build software

- Move from CSE 143 problems toward what you'll see in upper-level CSE courses and in industry

Specifically, how to write code of

- Higher **quality**
- Increased **complexity**

We will discuss *tools* and *techniques* to help with this

- There are *timeless principles* to both

# What is high quality code?

---

In summary, we want our code to be:

1. Correct
2. Easy to change
3. Easy to understand
4. Easy to scale (modular)

These qualities also allow for increased complexity

# What we will cover in CSE 331

---

- Everything we cover relates to the 4 goals
- We'll use Java but the principles apply in any setting

## **Correctness**

1. Tools
  - Git, IntelliJ, JUnit, Javadoc, ...
  - Java libraries: equality & hashing
  - Adv. Java: generics, assertions, ...
  - debugging
2. Inspection
  - reasoning about code
  - specifications
3. Testing
  - test design
  - coverage

## **Changeability**

- specifications, ADTs
- listeners & callbacks

## **Understandability**

- specifications, ADTs
- Adv. Java: exceptions
- subtypes

## **Modularity**

- module design & design patterns
- event-driven programming, MVC, GUIs

# Back to Goals

---

- CSE 331 will teach you to how to write correct programs
- What does it mean for a program to be **correct**?
  - Specifications
- What are ways to **achieve correctness**?
  - Principled design and development
  - Abstraction and modularity
  - Documentation
- What are ways to **verify correctness**?
  - Testing
  - Reasoning and verification

---

*Some new slides to tie the pieces together...*

# Divide and conquer: Modularity, abstraction, specs

---

No one person can understand all of a realistic system

- **Modularity** permits focusing on just one part
- **Abstraction** enables ignoring detail
- **Specifications** (and **documentation**) formally describe behavior
- **Reasoning** relies on all three to understand/fix errors
  - Or avoid them in the first place
  - **Proving, testing, debugging**: all are intellectually challenging

# How CSE 331 fits together

---

Lectures: ideas	⇒ Assignments: get practice
Specifications	⇒ Design classes
Testing	⇒ Write tests
Subtyping	⇒ Write subclasses
Equality & identity	⇒ Override equals, use collections
Generics	⇒ Write generic classes
Design patterns	⇒ Larger designs; MVC
Reasoning, debugging	⇒ Correctness, testing
Events	⇒ GUIs
Systems integration	⇒ N/A

# What you have learned in CSE 331

---

Compare your skills today to 10 weeks ago

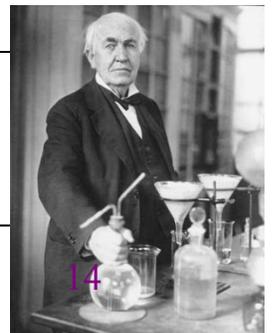
- Theory: abstraction, specification, design
- Practice: implementation, testing
- Theory & practice: correctness

Bottom line aspiration: Much of what we've done would be *easy* for you today

This is a measure of how much you have learned

There is no such thing as a “born” programmer!

Genius is 1% inspiration and 99% perspiration.  
Thomas A. Edison



# What you will learn later

---

- Your next project can be much more ambitious
  - But beware of “second system” effect
- Know your limits
  - Be humble (reality helps you with this)
- You will continue to learn
  - Building interesting systems is never easy
    - Like any worthwhile endeavor
  - Practice is a good teacher
    - Requires thoughtful introspection
    - Don't learn *only* by trial and error!
  - Voraciously consume ideas *and* tools

# What comes next?

---

## Courses

- CSE 403 Software Engineering
  - Focuses more on requirements, software lifecycle, teamwork
- Capstone projects
- Any class that requires software design and implementation

## Research

- In software engineering & programming systems
- In any topic that involves software

## Having an impact on the world

- Jobs (and job interviews)
- Larger programming projects

# Last slide

---

- System building is fun!
  - It's even more fun when you're successful!!
- Pay attention to what matters
  - Take advantage of the techniques and tools you've learned (and will learn!)
- On a personal note:
  - Don't be a stranger: I love to hear how you do in CSE and beyond as alumni
  - Students are amazing; I believe in you! 😊
- Closing thoughts?