
CSE 331

Software Design & Implementation

Topic: Course Wrap-up

 **Discussion:** What was the most valuable thing you learned this summer?

Reminders

- Please fill out course evals ASAP
 - Very valuable to me as a new instructor!

Upcoming Deadlines

- Submit *all* work by today at 11pm
- Check Canvas assignment grades on Monday

Last Time...

- End-of-quarter timeline
 - Lectures
 - Final Grades
- Short discussion
 - Accessibility

Today's Agenda

- Project Demos
- Course Wrap-up

Grading Timeline

- All work needs to be submitted to us by **Friday** at 11pm
- Course staff will finish grading HW9 and regrades by **Sunday evening**
- Grades will be posted on Canvas on late **Sunday night**
 - You should check that these are consistent with what you expected!
- Soham will calculate final GPA based on these grades on **Monday**
 - This is when I will account for extra credit and special circumstances
- Grades due to the university on **Tuesday at noon**

Project Demos

CSE 331

What was it all about?

But first....

Huge thanks to the folks who made it work

Course staff: 5 Amazing TAs:

Sparsh Binjrajka, Katherine Murphy,
Reshabh Sharma, Justin Tysdal, and David Xu.

*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 are the goals of CSE 331?

Learn the skills to be able to contribute to a modern software project

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

Specifically, how to write code of

- higher **quality**
- increased **complexity**

We will discuss *tools* and *techniques* to help with this and the *concepts* and *ideas* behind them

- there are *timeless principles* to both
- widely used across the industry

What is high quality?

Code is high quality when it is

1. **Correct**
Everything else is of secondary importance
2. Easy to **change**
Most work is making changes to existing systems
3. Easy to **understand**
Needed for 1 & 2 above

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

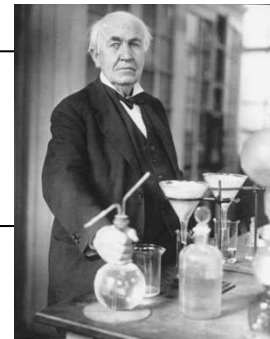
Compare your skills today to 10 weeks ago

- 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?

Before next class...

- Wrap-up any regrades for [HW1-8](#)
 - Won't accept late work after the last day of class
- Fill out [course evaluations](#) for me and the TAs
 - Also remember to nominate your TA
- Complete the Google Form about [additional work](#) on Ed
 - Tell me what other contributions you made to the course
- Check [Canvas grades](#) on Monday