Introduction to CSE 331
Software Design & Implementation

CSE 331
University of Washington

Michael Ernst
Welcome to CSE 331!

CSE 331 will teach you 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, modularity
  - Documentation

- **What are ways to verify correctness?**
  - Testing
  - Reasoning and verification

- **Moving beyond novice programming**
  - Larger programs
    - Small programs are easy; complexity changes everything
  - Effective use of tools: Java, IDEs, debuggers, JUnit, Javadoc, Git
    - Principles >> tools
Managing complexity

• Abstraction and specification
  – Procedural, data, control flow
  – Why they are useful and how to use them

• Writing, understanding, and reasoning about code
  – The examples are in Java, but the issues are more general
  – Object-oriented programming

• Program design and documentation
  – What makes a design good or bad (example: modularity)
  – The process of design and design tools

• Pragmatic considerations
  – Testing
  – Debugging and defensive programming
  – Managing software projects (more in CSE 403)
The goal of system building

• To create a **correctly functioning artifact**!

• All other matters are secondary
  – Many of them are *essential* to producing a correct system

• We insist that you learn to create correct systems
  – This is hard (but fun and rewarding!)
Why is building good software hard?

• Large software systems are enormously complex
  – Millions of “moving parts”
• People expect software to be malleable
  – After all, it’s “only software”
  – Software mitigates the deficiencies of other components
• We are always trying to do new things with software
  – Relevant experience often missing

• Software engineering is about:
  – Managing complexity
  – Managing change
  – Coping with potential defects
    • Customers, developers, environment, software
  – Communication (with people and computers)
Programming is hard

• It is surprisingly difficult to specify, design, implement, test, debug, and maintain even a simple program
• CSE 331 will challenge you
• If you are having trouble, *think* before you act
  – Then, look for help
• The assignments are reasonable if you apply the techniques taught in class
  – ... but hard to do in a brute-force manner
  – There is a method to our madness
CSE 331 is hard (but very rewarding)

- You will learn a lot!
- Be prepared to work and to think
- The staff will help you learn
  - We will work hard as hard as you do
Course staff

• Lecturer:
  – Michael Ernst

• TAs:
  – Alexey Beall
  – Avidant Bhagat
  – Michael Hart
  – Anny Kong
  – Kaushal Mangipudi
  – Jacob Murphy
  – Kaidi Pei
  – Jason Qiu
  – Andrew Tran
  – Joyce Zhou

Ask us for help!
Prerequisites

• Knowing Java is a prerequisite
  – We assume you have mastered 142 and 143
  – ... and you remember it

Examples:
• Sharing:
  – Distinction between == and equals()
  – Aliasing (multiple references to the same object)

• Subtyping
  – Varieties: classes, interfaces
  – Inheritance and overriding

• Object-oriented dispatch:
  – Expressions have a compile-time type
  – Objects/values have a run-time type
Logistics

- Website: https://cs.washington.edu/331
- See the website for all administrative details
- Read the handouts and required texts
- Take notes
- First assignment will be posted today
- You get 4 late days throughout the quarter
  - No other extensions (contact the instructor if you are hospitalized)
Academic Integrity

• Honest work is required of an engineer
• Collaboration policy on the course web. **Read it!**
  – Discussion is permitted
  – Carrying materials from discussion is not permitted
  – Everything you turn in must be your own work
    • Cite your sources, explain any unconventional action
  – You may not view others’ work
  – If you have a question, ask

• I trust you completely
• I have no sympathy for trust violations – nor should you