CSE 331
Software Design & Implementation

Hal Perkins

Fall 2015

Lecture 1 – Introduction & Overview

(Based on slides by Mike Ernst, Dan Grossman, David Notkin, Hal Perkins)

Welcome!

We have 10 weeks to move well beyond novice *programmer*:

- Larger programs
  - Small programs are easy: "code it up"
  - Complexity changes everything: "design an artifact"
  - Analogy: using hammers and saws vs. making cabinets (but not yet building houses)
- Principled, systematic software: What does "it's right" mean? How do we know "it's right"? What are best practices for "getting it right"?
- Effective use of languages and tools: Java, IDEs, debuggers, JUnit, JavaDoc, git, …
  - Principles are ultimately more important than details
    - You will forever learn details of new tools/versions

Concise to-do list

Before next class:

1. Familiarize yourself with website
   http://courses.cs.washington.edu/courses/cse331/15au/
2. Fill out office hour doodle to help us pick good times
3. Read syllabus and academic-integrity policy
4. Fill in online form after class if still trying to add
   - Instructions to be supplied at end of class
   - Go to any section tomorow if not registered
5. Do Homework 0 (in homework list), due 10AM Friday!
   - (Can put in dropbox even if not registered)

Who: Course staff

- Lecturer:
  - Hal Perkins: Faculty since last century, CSE331 veteran
- TAs:
  - Justin Bare
  - Nicholas Carney
  - Daniel Fang
  - Naruto Iwasaki
  - Geoff Liu
  - Erin Peach
- Office hours will be figured out ASAP

Get to know us!

  - Make sure this *feels like* a 40-person class with 90 students
  - We're here to help you succeed

Acknowledgments

- Course designed/created/evolved/edited by others
  - Michael D. Ernst
  - Dan Grossman
  - David Notkin
  - (me)
  - A couple dozen amazing TAs

- Hoping my own perspective offers benefits

- [Because you are unlikely to care, I won't carefully attribute authorship of course materials]

Staying in touch

- Course email list: **cse331a_au15@u.washington.edu**
  - Students and staff already subscribed
  - You must get announcements sent there
  - Fairly low traffic
- Message Board
  - For appropriate discussions; staff will monitor
  - Help each other out and stay in touch outside of class
- Course staff: **cse331-staff@cs.washington.edu**
  - For things that don't make sense to post on message board

Lecture and section

- Both required

- All materials posted, but they are visual aids
  - Arrive punctually and pay attention (& take notes!)
  - If doing so doesn't save you time, one of us is messing up (!)

- Section will often be more tools and homework-details focused
  - Especially next week: preparing for projects
  - This week's section is different – "lecture 2"
  - Rooms have changed recently – be sure to check

- Other posted handouts related to class material

Homeworks

- Biggest misconception about CSE331 (?)

  "Homework was programming projects that seemed disconnected from lecture"

  - If you think so, you are making them harder!
    - Reconsider
    - Seek out the connections by thinking-before-typing
    - Approaching them as CSE143 homework won't work well
    - Don't keep cutting with a dull blade

- First couple assignments are "more on paper", followed by software development that is increasingly substantial

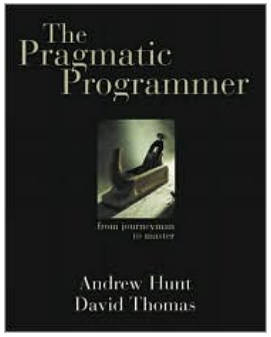
- Four late days for the quarter: save for emergencies
  - Max 2 per homework

Resources – Books

Required:

- *Pragmatic Programmer*, Hunt & Thomas
- *Effective Java* 2nd ed, Bloch

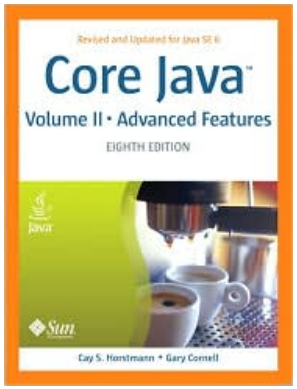
Serious programmers
should study these



Decent "Java book" is a wise thing to have

- *Core Java* Vol I, Horstmann

And use the Java API Docs



Readings (and quizzes)

- These are “real” books about software, approachable in 331
  - Occasionally slight reach: accept the challenge
- Overlap only partial with lectures
- Want to make sure you “do it”
  - Reading and thinking about software design is essential
    - Books seem expensive given your budget, but very cheap as a time-constrained professional
  - Will have some simple online reading quizzes
    - In a few batches; no late days
  - Material is fair-game for exams

Books? In the 21st century?

- Why not just use Google, Stack Overflow, Reddit, Quora, …?
- Web-search good for:
  - Quick reference (What is the name of the function that does …? What are its parameters?)
  - Links to a good reference
- (can be) Bad for
  - Why does it work this way?
  - What is the intended use?
  - How does my issue fit into the bigger picture?
- Beware:
  - Random code blobs cut-and-paste into your code (why does it work? what does it do?)
  - This inscrutable incantation solved my problem on an unstated version for no known reason

Exams

- Midterm: Monday, November 9, in class
- Final: Monday Dec. 14, 8:30-10:20 AM (Sorry! Not my fault!)
- All the concepts, different format than homework
  - Will post old exams from various instructors later

Academic Integrity

- Read the course policy carefully
  - Clearly explains how you can and cannot get/provide help on homework and projects
- Always explain any unconventional action
- I have promoted and enforced academic integrity since I first started teaching (many years ago)
  - Great trust with little sympathy for violations
  - Honest work is the most important feature of a university (or engineering or business). Anything less disrespects your colleagues (including me) and yourself.

Questions?

Anything I forgot about course mechanics before we discuss, you know, software?

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

Main topic: Managing complexity

- Abstraction and specification
  - Procedural, data, and control flow abstractions
  - Why they are useful and how to use them
- Writing, understanding, and reasoning about code
  - Will use Java, but the issues apply in all languages
  - Some focus on object-oriented programming
- Program design and documentation
  - What makes a design good or bad (example: modularity)
  - Design processes and tools
- Pragmatic considerations
  - Testing
  - Debugging and defensive programming
  - [more in CSE403: Managing software projects]

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!)

Related skill: *communication*

- Can you convince yourself and others something is correct via precise, coherent explanations?

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”
- 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

Programming is hard

- It is surprisingly difficult to specify, design, implement, test, debug, and maintain even a simple program
- CSE331 will challenge you
- If you are having trouble, *think* before you act
  - Then, look for help
- We strive to create assignments that are reasonable if you apply the techniques taught in class…
  … but likely hard to do in a brute-force manner
    … and almost certainly impossible to finish if you put them off until a few days before they're due

Prerequisites

- Knowing Java is a prerequisite
  - We assume you have mastered CSE142 and CSE143

Examples:

- Sharing:
  - Distinction between `==` and `equals()`
  - Aliasing: multiple references to the same object
- Object-oriented dispatch:
  - Inheritance and overriding
  - Objects/values have a run-time type
- Subtyping
  - Expressions have a compile-time type
  - Subtyping via `extends` (classes) and `implements` (interfaces)

You have homework!

- Homework 0, due online by 10 AM Friday
  - Write (don't run!) an algorithm to rearrange (swap) the elements in an array
  - in O(*n*) time (and preferably in a single pass)
  - And argue (prove) in concise, convincing English that your solution is correct!

- Purpose:
  - Great practice
  - Surprisingly difficult
  - So we can build up to reasoning about large designs, not just 5-10 line programs

CSE331 is hard!

- You will learn a lot!
- Be prepared to work and to think
- The staff will help you learn
  - And will be working hard, too

- So let's get going…
  - Before we create masterpieces we need to hone our ability to reason very precisely about code…

A Problem

"Complete this method such that it returns the index of the max of the first **`n`** elements of the array **`arr`**."

```
int index_of_max(int[] arr, int n) {
    ...
}
```

A Problem

"Complete this method such that it returns the index of the max of the first **`n`** elements of the array **`arr`**."

```
int index_of_max(int[] arr, int n) {
    ...
}
```

What questions do you have about the *specification*?

Given a (better) specification, is there 1 *implementation*?

Moral

- You can all write the code

- More interesting in CSE331:
  - What if n is 0?
  - What if n is less than 0?
  - What if n is greater than array length
  - What if there are “ties”?
  - Ways to indicate errors: exceptions, return value, …
  - Weaker versus stronger specifications?
  - Hard to write English specifications (n vs. n-1)

Concise to-do list

Before next class:

1. Familiarize yourself with website
   http://courses.cs.washington.edu/courses/cse331/15sp/

2. Read syllabus and academic-integrity policy

3. Fill in online form after class if still trying to add
   – Instructions to be supplied next

4. Do Homework 0 (in homework list), due 10AM Friday!

Reminder: to-do list

Before next class:

1. Familiarize yourself with website
   http://courses.cs.washington.edu/courses/cse331/15sp/
2. Fill out office hour doodle to help us pick good times
3. Read syllabus and academic-integrity policy
4. Fill in online form after class if still trying to add
   - Instructions to be supplied next
   - Go to any section tomorow if not registered
5. Do Homework 0 (in homework list), due 10AM Friday!
   - (Can put in dropbox even if not registered)