One handout up front!

CSE 331
Software Design & Implementation

Hal Perkins

Winter 2013

Lecture 0 – Course Introduction

Course staff

- Lecturer:
  - Hal Perkins
- TAs:
  - Wing Lam
  - David Mailhot
  - Lindsey Nguyen
  - James Okada
  - Ryan Tsoi

Ask us for help!

Welcome!

- We have 10 weeks to move to a level well above novice programmer:
  - Larger programs
  - Principled, systematic programming: What does it mean to get it right? How do we know when we get there? What are best practices for doing this?
  - Effective use of languages and tools: Java, IDEs, debuggers, JUnit, JavaDoc, svn
    - The principles are ultimately more important than the details
      - (Yeah, right…)

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
  - 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

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

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
- 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

Prerequisites

- Knowing Java is a prerequisite
  - We assume you have mastered 142 and 143

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

- 3 lectures/week + 1 section
  - You are responsible for what happens, even if you skip a day (but contact us if it is an emergency)
- All course materials are on the web (often after class): but **TAKE NOTES!**
- Communications:
  - Discussion board (not Delphic oracle)
    - Post/reply and it'll keep track of your new stuff
  - Mailing list: messages from course staff to everyone (you are subscribed if you are enrolled; you are responsible for messages sent to the list)

Requirements

- Primarily programming assignments but some written problem sets, approximately weekly (55%)
- 1 midterm (15%), 1 final (25%)
- 5% online quizzes, exercises, citizenship, etc.
- Collaboration: individual work unless announced otherwise; *never* look at or show your code to others
- Extra credit: when available, small effect on your grade if you do it – no effect if you don't

- We reserve the right to adjust percentages as the quarter evolves to reflect the workload

Academic Integrity

- Policy on the course web. **Read it!**
- Do your own work – always explain any unconventional action on your part
- I trust you completely
- I have no sympathy for trust violations – nor should you
- Honest work is the most important feature of a university (or engineering, or business). It shows respect for your colleagues *and yourself.*

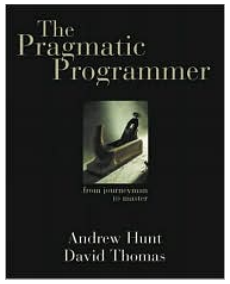
Deadlines

- Turn things in on time!
- But things happen, so …
  - You have 4 late days for the quarter for assignments (not quizzes, exercises)
  - No more than 2 per assignment
  - Counted in 24 hour chunks (5 min = 24 hours late)
  - If group projects, can only use if both partners have late days and both partners are charged
- That's it. No other extensions (but contact instructor if you are hospitalized)
- Advice: Save late days for the end of quarter when you (might) really need them

Resources – Books

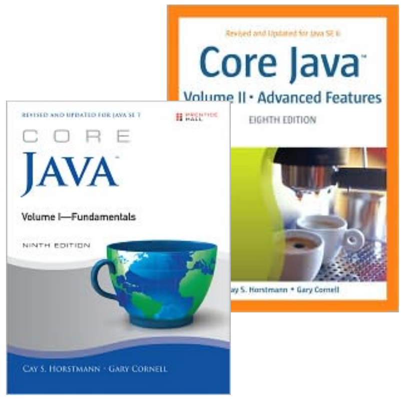
Required (assigned readings, some online quizzes) – every serious programmer should read these

- Pragmatic Programmer, Hunt & Thomas
- Effective Java 2nd ed, Bloch
  - Will be more proactive about quizzes, readings this quarter



Decent "Java book" if you want one

- Core Java Vol I, Horstmann



You have homework!

- Exercise 0, due online by 10 am Wednesday
  - Links went live right before class
- Write (don’t run!) an algorithm to rearrange the elements in an array
  - And argue that your solution is correct!
- No late submissions accepted on exercises or quizzes (late days are only for larger homework / programming assignments)

Work to do!

- If you're still trying to add the course, please sign the info sheet before leaving today
- Fill in the Office Hours Doodle on the web site
  - We're trying to get an idea what would be most useful
- Post an answer to the welcome message on the discussion list (get catalyst to track new postings for you)
- Exercise 0 due by 10 am Wed.
- So let's get going…
  - Before we create masterpieces we need to hone our technique….