Introduction to CSE 331
Software Design & Implementation

Autumn 2010

ASAP: Answer initial questionnaire on Catalyst
https://catalysttools.washington.edu/webq/survey/notkin/107233
Course staff

• Instructor: David Notkin (notkin@cs, CSE 542)
• Office hours
  – Monday 1-2PM; Thursday 12:30-1:30PM
  – and by appointment
  – and when my door is open
• Teaching assistants
  – Matthew Kay (mjskay@cs)
  – Nicholas Brekhus (gu13@cs)
• TA office hours TBA, held in the CSE 002 lab
The Tao (道) of CSE331

Programming (in Java)
- Control (loops, conditionals, methods, parameter passing, recursion, etc.)
- Variables
- Abstract data types (ADTs):
  - Stacks, linked lists, ...
  - Interfaces, inheritance and encapsulation
- Basics of complexity and performance tradeoffs
- Using off-the-shelf components from Java Collections

Designing and implementing more realistic software: managing complexity
- Abstraction and specification
- Writing, understanding and reasoning about code
- Program design and documentation: process and tools
- What makes a design good or bad?
- Pragmatic considerations
  - Testing
  - Debugging and defensive programming
- Software management issues
Prerequisites

• We assume you have “mastered” 142 and 143 – your Java programming may need some brushing up, of course!

• Examples of material we will use extensively
  – Sharing
    • Distinction between == and equals() – that is, are two objects the same object, or do two objects have equal values (the definition of which can be overridden)?
    • Aliasing – that is, 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
Learning styles: just two dimensions

Excerpted from
http://serc.carleton.edu/NAGTWorkshops/earlycareer/teaching/learningstyles.html

• “Active” learners tend to retain and understand information best by doing something active with it – discussing or applying it or explaining it to others. Reflective learners prefer to think about it quietly first.”

• “Sequential” learners tend to gain understanding in linear steps, with each step following logically from the previous one. Global learners tend to learn in large jumps, absorbing material almost randomly without seeing connections, and then suddenly ‘getting it.’”

• Each of you has different learning styles (as do the course staff); with hard work, you’ll all get “there”

I found “How to help someone use a computer” interesting: you are at varying points, in varying dimensions, with respect to your knowledge – learn stuff, work hard, but don’t beat yourself up! And definitely don’t beat others up!
Two goals of software system building
(Barry Boehm)

- Building the right system.
- Building the system right.

In CSE331, the second goal is the focus – that is, we focus only on creating a correctly functioning artifact.

- Even this one goal is challenging: to specify, design, implement, test, debug and maintain even a simple program can be surprisingly complicated.
- And learning how to do it can be fun and rewarding.
Why is building good software hard?

• Large software systems are enormously complex
  – Millions of “moving parts” with very little duplication

• Varied stakeholders
  – Customers (known and unknown), developers, managers, testers, marketers, ...

• People expect software to be malleable
  – After all, it’s “only software” – it may be hard to change, but it’s easier to change than anything else!
  – Try changing hardware, for example!

• We are always trying to do new things with software
  – Always pushes past our comfort zones

• ...

UW CSE331 Autumn 2010
Software engineering is …

“The multi-person construction of multi-version software.”
–David Parnas

• Must manage product complexity
• Must manage teams of people
• Must manage change over time
• Must cope with potential defects
Some observations

• If you are having trouble, *think* before you type – programming by permutation rarely succeeds

• If you still have trouble, ask for help
  – Do not stay stuck on something for a long time without asking for help – doing this will likely make the course far more time-consuming than it should be

• We strive to create assignments that are reasonable if you apply the techniques taught in lecture (but they may be hard to do in a brute-force manner)
Logistics

• Website  [http://www.cs.washington.edu/331](http://www.cs.washington.edu/331)
  — See it, read it, find our email and office hours, read the handouts, check out the texts, and more. Lots more. Indeed, a huge amount. Keep at it!

• Problem Set 0 is due on Friday @ 8PM
  — Thursday’s section will be in CSE 002 with the staff helping you with the initial setup for the course – these are the first couple of parts on PS0
  — The assignment write-up is long and possibly intimidating – the work is far more manageable than the write-up, so get started soon!
Collaboration policy

• Discussion is permitted ... indeed, encouraged!
• Representing someone else’s work as your own is not permitted
• Please apply the Gilligan’s Island (well, now it should be House or Glee, perhaps) rule
  – if you watch mindless TV for 30-60 minutes after a discussion with classmates,
  – and you can still reproduce the materials from memory (no notes, no email, etc.),
  – then you can consider it your work/knowledge
• Familiarize yourselves with the CSE, COE and UW policies on academic honesty – we rely on them heavily.
• If you have a question about what is allowed, ask me!