CS 2112 / ENGRD 2112
Object-Oriented Design and Data Structures — Honors

Fall 2015

Andrew Myers
Cornell University

Lecture 1: Introduction
Course staff

- Instructor: Prof. Andrew Myers
- Grad TAs: Yizhou Zhang, Clara Thomas
- Undergrad course staff:
  - Laura Herrle
  - James Senter
  - Grant Mulitz-Schimel
  - Agnes Csaki
  - Sumner Hearth
  - Matthew Gharrity
  - Matt Li
  - Zander Bolgar
  - Alex Renda
What it’s about

An introduction to computer science and software engineering

• **Programming language features**
  – data abstraction, subtyping, generic programming
  – concurrency and threads

  *Using Java. Not a course on Java!*

• **Object-oriented design — organizing large programs**
  – specifications
  – design patterns
  – frameworks and event-driven programming

• **Data structures and algorithms**
  – recursive algorithms and data structures
  – algorithm analysis and designing for efficiency
  – asymptotic complexity, induction
  – arrays, lists, stacks, queues, trees, graphs, hash tables
Web site

• Your source for information about this course: http://courses.cs.cornell.edu/cs2112
  – Lecture notes: you are expected to read
    • mostly not slides
    • may not include everything covered in lecture
    • may include extra not covered in lecture
    • often updated after the lecture
  – Assignments
    • may be updated after initial release
  – Pointers to resources
Communicating with staff

• Best way: use Piazza (http://piazza.com/class#fall2015/cs2112) to post questions
  – Answering other questions (well) has good karma.
  – Watch out for violating academic integrity

• Course announcements posted using Piazza (or emailed to all students)

• Consultants—hours online, location TBA
  – “Front line” for answering questions – consulting hours start today
CMS

• Assignments will be submitted to CMS.
• Grades, solutions will be posted on CMS (http://cms.csuglab.cs.cornell.edu)
• Regrade requests should be posted to CMS soon after receiving grade (remind us if necessary...)
Meetings

• **Lectures**: TTh 10:10-11, Gates G01

• **Discussion sections** (attend 1 per week)
  – T 12:20-1:10 (Upson 205)
  – W 1:25-2:15 (Upson 205)

• **Labs** (attend 1 per week)
  – Monday 7:30-8:20 (Upson 205)
  – Wednesday 7:30-8:20 (Olin 245)

• 4 hours per week — attendance required
Room caps

• If the lecture, the lab, or the recitation is full...
  – don’t panic, you can still take the course
  – let us know, including your constraints
  – we will work out a solution
Assignments

- 7 assignments
  - mostly programming but some written problems
  - total to 45% of total score
- First 3 assignments done solo; final project done with a partner.
- Assignment late penalties:
  - 1 day late: 10%
  - 2 days late: 20%
  - 3 days late: 40%
  - weekends count as 1 day
Exams and more

• One evening prelim: Oct 1
  – 15% of total score

• Final exam: sometime during Dec 9–7, location and time TBD
  – 35% of total score

• 5% of score:
  – participation (in-class, Piazza, course evals)
  – possible in-class quizzes
Labs

• Will do programming exercises, solve problems, learn about tools
• Bring a laptop if you have one (or share if you don’t—you will work in pairs anyway)
• First lab: Wednesday in Olin 245 (set up Eclipse and do some programming) — attend if you can even if you are normally in the Monday lab
Textbook

  - Should be available at campus store.
  - On reserve in library
  - Recommended, not required
  - Also the 2110 textbook
  - Not heavily used—can share with a friend or two.
Academic integrity

- **You must not** represent someone else’s work as your own or facilitate someone else’s fraud.
  - Copying code, answers = never okay
  - Letting others copy you? Also a violation.
  - You must be able to explain your answers fully.
  - Discussions with others are definitely fine if they could have happened in a lightless room.
- **We will use highly effective tools for detecting plagiarism.**
- **Report** any discussions about assignments and any use of external code.
- **Our goal:** spend time on course content.
CS 2112 or ENGRD 2112?

Does not matter
CS/ENGRD 2110 or CS/ENGRD 2112?

• 2112 is an ‘honors’ version of 2110.
  – aimed at CS majors: smaller
  – harder and more interesting assignments including a final project spanning 4 assignments
  – more material
    • e.g., more algorithms and their analysis (theory)
    • e.g., more about design, design patterns (practice)
  – more credits (4 vs. 3)
2110 vs. 2112

• Warning: you will be challenged here
What it’s about

An introduction to computer science and software engineering

- **Programming language features**
  - data abstraction, subtyping, generic programming
  - concurrency and threads
  
  *You will learn Java (better) but it’s not a course about Java!*

- **Object-oriented design — organizing large programs**
  - specifications
  - design patterns
  - frameworks and event-driven programming

- **Data structures and algorithms**
  - recursive algorithms and data structures
  - algorithm analysis and designing for efficiency
  - asymptotic complexity, induction
  - arrays, lists, stacks, queues, trees, graphs, hash tables
Sam Loyd’s 8 Puzzle

Goal: Given an initial configuration of tiles, find a sequence of moves that will lead to the sorted configuration.

A particular configuration is called a state of the puzzle.
State Transition Diagram of 8-Puzzle

State Transition Diagram: picture of adjacent states.
A state Y is adjacent to state X if Y can be reached from X in one move.
State Transition Diagram for a 2x2 Puzzle

Sorted State

Solutions for this state:
- SWN
- WSENWSENW
- SWEWN
- ........
Graphs

- State transition diagram in previous slide is an example of a graph: a mathematical abstraction
  - nodes (or vertices): the puzzle states
  - edges (or arcs): the transitions, possibly labeled
- Graphs are all around us: airline routes, roadmaps, org charts, pipelines, ...
Graph algorithms

• Large toolbox of efficient algorithms for graphs help us solve problems:
  – searching for best nodes/shortest paths
  – finding maximum flow through graph
  – minimum spanning trees
  – ... 

• And known hardness results (e.g., finding Hamiltonian cycles) tell you what you can’t solve.
Software design choices

- What operations should puzzle objects have?
- How do we represent states? The initial state?
- How do we present information to the user and support interaction?
- How do we break the coding up into parts that can be coded independently?
- How to structure code so it can be maintained, upgraded?
Why you need CS 2112

• Data structures and algorithms to solve problems efficiently and effectively.

• Design techniques to produce code that works quickly and keeps working.

• Computer science:
  – algorithms, data structures, programming languages, design principles, knowledge of what is possible and feasible.

• Good programmers have more fun!
  – 10X more productive
  – better able to adapt, grow, see opportunities, change the world.
Next steps

• Get your Piazza account set up ASAP
• Keep an eye on the 2112 website.
• Download the first programming assignment, released today, due in one week
• Make sure you have Eclipse downloaded and working. See consultants for help.
• Come to lab on Wednesday for help getting started.
• Have fun!