Welcome to CS2110!

- We’ll be learning about…
  - Abstract data types and generics and reflection and other cool Java features
  - Reasoning about complex problems, analysis of the algorithms we create to solve them, and implementing those tricky algorithms with elegant, easy to understand, correct code
  - Recursion on graphs and other linked structures
  - Algorithmic complexity
  - (+ a lecture or two on quantum computing)
Is CS2110 right for you?

- CS32110 assumes you know Java
  - For example, you took cs111x at Cornell
  - Or took a high school course and got a 4 or 5 on the CS AP exam

- Don’t take cs1110 just because you are worried that your high school Java experience won’t do

- We recommend against trying to skip directly into cs3110. Doing so requires permission from both Professor Birman and Professor Joachims!
Lectures

- TR 10:10-11am, Olin 155
  - Attendance is mandatory

- ENGRD 2110 or CS 2110?
  - Same course! We call it CS 2110
  - Non-engineers sign up for CS 2110
  - Engineers sign up for ENGRD 2110
Sections

- Like lecture, attendance is mandatory
- Usually review, help on homework
- Sometimes new material
- Section numbers are different for CS and ENGRD
- Each section will be led by a member of the teaching staff
- No permission needed to switch sections
- You may attend more than one section if you wish
## Sections

### Sections Start Next Week!

<table>
<thead>
<tr>
<th>Non-Eng</th>
<th>Eng</th>
<th>Day</th>
<th>Time</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>4943</td>
<td>DIS 201</td>
<td>T</td>
<td>12:20PM - 01:10PM</td>
<td>OLH 245</td>
</tr>
<tr>
<td>4945</td>
<td>DIS 202</td>
<td>T</td>
<td>01:25PM - 02:15PM</td>
<td>OLH 165</td>
</tr>
<tr>
<td>4947</td>
<td>DIS 203</td>
<td>T</td>
<td>02:30PM - 03:20PM</td>
<td>BRD 140</td>
</tr>
<tr>
<td>4949</td>
<td>DIS 204</td>
<td>W</td>
<td>12:20PM - 01:10PM</td>
<td>PHL 219</td>
</tr>
<tr>
<td>4951</td>
<td>DIS 205</td>
<td>W</td>
<td>01:25PM - 02:15PM</td>
<td>HLS 306</td>
</tr>
<tr>
<td>4953</td>
<td>DIS 206</td>
<td>W</td>
<td>02:30PM - 03:20PM</td>
<td>PHL 219</td>
</tr>
<tr>
<td>4955</td>
<td>DIS 207</td>
<td>T</td>
<td>12:20PM - 01:10PM</td>
<td>HLS 401</td>
</tr>
</tbody>
</table>
Resources

- Course web site
  - [http://courses.cs.cornell.edu/cs2110](http://courses.cs.cornell.edu/cs2110)
  - Watch for announcements

- Course discussion forum
  - Currently setting this up on Google groups
  - Good place to ask questions
  - Once we have it running, we do expect you to check *daily* for updates on homeworks!
Academic Excellence Workshops

- Two-hour labs in which students work together in cooperative setting
- One credit S/U course based on attendance
- Time and location TBA
- See the website for more info

www.engineering.cornell.edu/student-services/learning/academic-excellence-workshops/
Resources

- Book: Frank M. Carrano, *Data Structures and Abstractions with Java, 2nd ed.*, Prentice Hall
  - *Note: The 1st edition is seriously obsolete*
  - Sharing the textbook is a fantastic idea. You won’t need a personal copy but you do need access to a copy from time to time
  - Copies of 2nd Edition on reserve in Engr Library
- Additional material on Prentice Hall website

- Great Java resource: the online materials at the Sun JDK web site. Google has it indexed.
Obtaining Java

- See Resources on website
- We recommend that you work with Java 6
- Need Java Development Kit (JDK), not just Java Runtime Environment (JRE)
- Many production releases... latest is usually best
Eclipse IDE

- IDE: Interactive Development Environment
  - Helps you write your code
  - Protects against many common mistakes
  - At runtime, helps with debugging
- Follow “Resources” link to download

“In my country of Kazakhstan everyone is use Eclipse! Excellent for hack American web site and steal credit card.”
Java Help

- CS 2110 assumes basic Java knowledge
  - classes, objects, fields, methods, constructors, static and instance variables, control structures, arrays, strings, exposure to inheritance

- Need a refresher? Consider CS 1130
  - Transition to Object-Oriented Programming.
  - (self-guided tutorial, material on website)
Coursework

- 5 assignments involving both programming and written answers (45%)
- Two prelims (15% each)
- Final exam (20%)
- Course evaluation (1%)
- Occasional quizzes in class (4%)
Assignments

- Except for assignment A1, assignments may be done by teams of two students
  - A1 is already posted on CMS
  - Just the same, we encourage you to do them by yourself and have considered making this the rule
  - Finding a partner: choose your own or contact your TA. Newsgroup may be helpful.
Academic Integrity… Trust but verify!

- We use artificial intelligence tools to check each homework assignment
  - The software is very accurate!
  - It tests your code and also notices similarities between code written by different people

- Sure, you can fool this software
  - … but it’s easier to just do the assignments
  - … and if you try to fool it and screw up, you might fail the assignment or even the whole course.
Sam Loyd’s 8 Puzzle

- Initially scrambled configuration
- Transition (N/S/E/W means tile moves North/South/East/West)
- Sequence of moves
- Sorted configuration

- 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
-------
Simulating the 8-puzzle

- What operations should puzzle objects support?
- How do we represent states?
- How do we specify an initial state?
- What algorithm do we use to solve a given initial configuration?
- How should we present information to the user? (GUI design)
- How to structure the program so it can be understood, maintained, upgraded?
Graphs

- State Transition Diagram in previous slide is an example of a **graph**: a mathematical abstraction
  - **vertices** (or **nodes**): (e.g., the puzzle states)
  - **edges** (or **arcs**): connections between pairs of vertices
  - Vertices and edges may be labeled with some information (name, direction, weight, cost, …)
- Other examples of graphs: roadmaps, airline routes, . . .
  - A common vocabulary for problems
A very different example of a graph

- Garmin GPS unit tracks your bike ride
Actual data is a graph!

- Garmin GPS records a series of locations
  - (time, GPS-coordinates, distance since last record, temperature, etc)
  - The graph is defined by the sequence of points

- Road maps are also graphs and have a similar representation

- Allows Garmin to match my bike ride to a map for display
... graphical displays
... or comparisons

- How did I do today compared to the last time I rode this same route?
Path Problems in Graphs

- Is there a path from node A to node B?
  - If you can solve this problem you can solve the 8-puzzle, or recommend a bike route

- What is the shortest path from A to B?
  - Find fastest way to solve the 8-puzzle
  - Or the Google Maps / Mapquest problem

- Traveling salesman problem

- Hamiltonian cycles
Why take CS 2110?

- You’ll learn to think in a more logical, structured way
- In the modern world, computational thinking pervades almost every subject of inquiry, is reshaping almost every industry, and is even reshaping society
- Mastery of computational thinking will help you become a master of the universe!
- Also: Great job prospects with high salaries…
So let’s go for it!

We hope you have fun, and enjoy programming and “computational thinking” as much as we do.