Welcome to CS2110!

Learning about:

- OO, abstract data types, generics, Java Collections, …
- Reasoning about complex problems, analyzing algorithms we create to solve them, and implementing algorithms with elegant, easy-to-understand, correct code
- Testing; Reasoning about correctness
- Data structures: linked lists, trees, graphs, etc.
- Recursion
- Algorithmic complexity
- Parallelism — threads of execution
Homework!

**Homework 1.** Read article *Why Software is So Bad.*
   Link: Course website -> Lectures notes (Lecture 1)

**Homework 2.** Get Java and Eclipse on your computer

**Homework 3.** Spend some time perusing the course website.
   Look at course information, resources, links, etc.
What’s CS 2110 about?

- Computational tools are “universal” but the key is to master computational thinking.
  - Looking at problems in ways that lead naturally to highly effective, correct, computational solutions
  - There are many ways to do anything, but some are far better than others
- Mastery of computational thinking will help you become a master of the universe!
- Great job prospects with high salaries...
Is CS2110 right for you?

- Knowledge of Java not required
  - Only ~30% of you know Java — others know Matlab, Python …
  - Requirement: comfort with some programming language.
    Prior knowledge of OO not required.
  - We assume you do not know Java!

- Don’t take CS1110 just because you are worried that your high school programming experience won’t do

- Consider taking the honors version: CS2112, taught by Andrew Meyers. Given at same time. Switching between 2110 and 2112 during first 3 weeks is fine.
Lectures

- TR 10:10-11am, Statler auditorium
  - Attendance mandatory

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

- Like lecture, attendance is mandatory
- Sometimes review, help on homework
- Sometimes new material
- Section numbers are different for CS and ENGRD
- Each section led by member of teaching staff
- No permission needed to switch sections, but do register for whichever one you attend
An “enrichment” course

We want to help students who might otherwise feel overwhelmed by CS2110

Gives more explanation of core ideas behind Java, programming, data structures, assignments, etc.

Taught by Gries, 1 credit S/U

Only for students who also take CS2110

Only requirement: Attend weekly lecture
Academic Excellence Workshops

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

www.engineering.cornell.edu/academics/undergraduate/curriculum/courses/workshops/index.cfm
Piazza

- Click link on our web page to register

- Incredible resource for 24 x 7 help with anything

- We keep an eye on it and answer questions. YOU can (and will) too. Visit the Piazza often.
Resources

  - *Note: 2nd edition is okay*
  - Share textbook: fantastic idea. You do need access to it from time to time
  - Copies on reserve in Engr Library

- Additional material on Prentice Hall website
  - “e-Book” not required

- PPT slides (on course website and Piazza) outline all of OO in Java. Has index at beginning

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

- Follow instructions on our Resources web page
  - Make sure you have Java JDK 1.7, if not download and install. We explain how on the web page.
  - Then download and install the Eclipse Juno « IDE » for Java developers from Eclipse IDE for Java Developers

- Test it out: launch Eclipse and click “new>Java Project”
  - This is one of a few ways Java can be used
  - When program runs, output is visible in a little console window
Eclipse IDE

- IDE: Integrated Development Environment
  - Helps you write your code
  - Protects against many common mistakes
  - At runtime, helps with debugging

- Follow Resources link to download and install

“In my country of Kazakhstan everyone is use Eclipse and Java! Java 1.7 is best for hack American web site and steal credit card.”
DrJava IDE

- IDE: Integrated Development Environment
- DrJava is a much simpler IDE, few features
- We use it **only** to demo Java features and programming concepts. Has an “interactions pane”, which allows trying things without requiring a complete Java program.
- DON’T use it for course assignments —use Eclipse
- Free at [www.drjava.org](http://www.drjava.org)
Coursework

- 5–7 assignments involving both programming and written answers (35%)
- Two prelims (15% each)
- Final exam (30%)
- Course evaluation (1%)
- Possible surprise in-class quizzes (4%)

Formula will change as the course progresses and we make changes in assignments, give quizzes, etc.

Exams are most important aspect in determining final grade
Assignments

- Teams of one or two
  - A0 and then A1 will be posted soon on the CMS
  - Finding a partner: choose your own or contact your TA. Piazza can be helpful.

Two kinds of assignment:
- **Vanilla**: specific experience to learn and practice what’s being taught. We give exact instructions for doing it
- **Chocolate**: Open-ended project done in 3 chunks (AI robot butterfly). Parts of the design are left to you. CS 2111 will give more help on it.
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.
Types in Java

References in text and in JavaSummary

- type: A.14 slide 4
- variable: A.13 slide 7
- variable declaration: A.15 slide 7
- Primitive types, A.16, back inside cover slide 5
- Constants, A.17 slide 20
- Assignment, A.18-A.20 slide 8
- Casting, A.21 slide 6
- Expressions: A.22-A.23
- Precedences: A.24, back inside cover
- Unicode character codes, back inside cover
Type: Set of values together with operations on them.

Type integer:
values: ..., −3, −2, −1, 0, 1, 2, 3, ...
operations: +, −, *, /, unary −

God’s integers! Can represent them in many ways — decimal, binary, octal, maybe as strokes |||| (that’s 4)

Do you know how your computer represents them?
Type: Set of values together with operations on them.

Matlab and Python are weakly typed: One variable can contain at different times a number, a string, an array, etc. One isn’t so concerned with types.

Java strongly typed: A variable must be declared before it is used and can contain only values of the type with which it is declared.

Valid Python sequence:
- \( x = 100; \)
- \( x = 'Hello World'; \)
- \( x = (1, 2, 3, 4, 5); \)

Corresponding Java
- \( \text{int } x; \)
- \( x = 100; \)
- \( x = "Hello"; \)

Declaration of \( x \): \( x \) can contain only values of type \text{int}

Illegal assignment: “Hello” is not an \( \text{int} \)
Weakly typed versus strongly typed

**Weakly typed:**
Shorter programs, generally.
Programmer has more freedom, language is more liberal in applying operations to values.

**Strongly typed:**
Programmer has to be more disciplined. Declarations provide a place for comments about variables.
More errors caught at compile-time (e.g. it’s a syntax error to assign a string to an `int` variable).

Note: weak and strong typing not well defined; literature has several definitions
Most-used ‘primitive’ types

**int**: values: $-2^{31}$ .. $2^{31}-1$
operations: $+$, $-$, $\ast$, $/$, $\%$, unary $-$

**double**: values like: $-22.51E6$, $24.9$
operations: $+$, $-$, $\ast$, $/$, $\%$, unary $-$

**char**: values like: '\V', '$', '\n'
operations: none

**boolean**: values: true, false
operations: ! (not), && (and), || (or)

$b \% c$ : remainder when $b$ is divided by $c$.

$67 \% 60 = 7$

Write values in “scientific notation”

Use single quotes for type char.

'\n' is new-line char

Can’t use integers as booleans!
About ‘primitive’ type int

**int**: values: $-2^{31}$ .. $2^{31}-1$, i.e.

operations: +, −, *, /, %, unary −

**Integer.MAX_VALUE**: name for max int value: $2^{31}-1$: 2147483647

**Integer.MAX_VALUE + 1** is $-2^{31}$: -2147483648 **WRAP-AROUND**
## Primitive number types

<table>
<thead>
<tr>
<th>Integer types:</th>
<th>byte</th>
<th>short</th>
<th>int</th>
<th>long</th>
<th>usual operators</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 byte</td>
<td>2 bytes</td>
<td>4 bytes</td>
<td>8 bytes</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Real types:</th>
<th>float</th>
<th>double</th>
<th>usual operators</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4 bytes</td>
<td>8 bytes</td>
<td>–22.51E6</td>
</tr>
</tbody>
</table>

Use these to save space.

Have an array of 1,000,000 integers in range 0..7?
Use a **byte** array rather than an **int** array.

Don’t worry about this in next 7-8 weeks. Use **int** and **double**.
Casting among types

\[(\text{int}) \ 3.2\] casts \textbf{double} value 3.2 to an \textbf{int}

any number type \hspace{2cm} any number expression

narrow \hspace{2cm} may be automatic cast \hspace{2cm} wider

\begin{align*}
\text{byte} & \quad \text{short} & \quad \text{int} & \quad \text{long} & \quad \text{float} & \quad \text{double} \\
\text{must be explicit cast, may truncate} & \quad & \quad & \quad & \quad & \quad
\end{align*}

\[(\text{int})\text{ is a unary prefix operator, just like } -\]

\[ - - 3 \quad \text{evaluates to } 3 \]

\[ - (\text{int}) \ 3.2 \quad \text{evaluates to } -3 \]
Char is a number type!

\texttt{char} is a number type: \( (\texttt{int}) 'V' \) \( \overset{}{\text{(char) 86}} \)

Unicode repr. in decimal: 86

Unicode: 16-bit char repr. Encodes chars in just about all languages. In java, use hexadecimal (base 16) char literals:

'\u0041' is 'A'
'\u0042' is 'B'
'\u0056' is 'V'
'\u0024' is '$'

'\u0950' is 'ॐ' — Om, the sound of the universe
'\u0950' is 'ॐ'
'\u5927' is '大' — 大衛 is (I think) a transliteration of David into Chinese (Da Wei)
'\u885b' is '衛'

See \url{www.unicode.org}
**Declaration**: gives name of variable, type of value it can contain

- `int x;`  
  Declaration of `x`, can contain an **int** value

- `double area;`  
  Declaration of `area`, can contain a **double** value

- `int[] a;`  
  Declaration of `a`, can contain a pointer to an **int** array. We explain arrays much later

- `x 5 int`  
- `area 20.1 double`  
- `a int[]`
Assignment statement

Much like in other languages — need ‘;’ at end:

<variable> = <expression> ;

```java
int x;
x = 10;
... other code
x = x + 1;
```

Have to declare `x` before assigning to it.

```java
int x = 10;
... other code
x = x + 1;
```

Can combine declaration with an initializing assignment. Shorthand for a declaration followed by an assignment.
Every expression has a type, which depends on its operators and the types of its operands in a natural way.

**Rule:** In `x = e;` type of `e` has to be same as or narrower than type of `x`. Reason: To avoid possibly losing info without the programmer realizing it.

- `double y = 5 + 1;`  
  The value of 5+1 is automatically cast from type `int` to type `double`.

- `int x = 75.5 + 1;`  
  Illegal: The exp value is of type `double`.

- `int x = (int) (75.5 + 1);`  
  You can cast to `int` explicitly. 76 will be stored in `x`.
A function in Matlab, Python, and Java

function s = sum(a, b) 
    % Return sum of a and b 
    s= a + b;

def sum(a, b):
    """ return sum of a and b""
    return a + b

/** return sum of a and b */

public static double sum(double a, double b) {
    return a + b;
}

Specifications:
- In comment before function
- Declaration of parameters a and b
- Return type