

# CS/ENGRD 2110

## FALL 2016

Lecture 1: Overview and intro to types  
<http://courses.cs.cornell.edu/cs2110/2016fa>

2 OO Programming and Data Structures

## Welcome to CS2110!

101 Freshmen  
 243 Sophomores  
 062 Juniors  
 062 Seniors  
 055 Meng/Masters  
 012 Graduate PhD

**Instructors:**  
 David Gries, Mike George

**Recitation leaders:** 20

**Consultants:** 23

535 Total

As of Sunday, 21 August.

3

## Did you receive an email from us Sunday?

If not, there's something wrong with how you forward from your Cornell netid. About 3 people's email bounced.

4


## 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, hash tables, graphs, etc.
- Recursion
- Algorithmic complexity
- Parallelism —threads of execution

5


## Usefulness of 2110



This summer I'm working in particle physics, making simulations of some of the background signal we'd expect to see in our detector for an experiment run in the particle accelerator. What I'm working on a clustering algorithm to put together energy depositions from several quantized points in the detector to learn what the initial particle's energy and position was. After some thought, I decided the best first sweep over this data would be to do a **depth first search** starting about a high energy deposition in the calorimeter. It works great, and my PI was very excited about the results!

6

## Usefulness of 2110



I am working at a startup in Silicon Valley this summer ... that does subscription-based financial management and billing among other things. It has been pretty incredible the amount I've learned from your class that relates to this internship and I have definitely decided to pursue computer science (I was initially engineering physics).

## 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, on the level of CS1110 (Python based) and CS1112 (Matlab based). Prior knowledge of OO not required.
- We assume you do not know Java!

## 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.

### Homework 4. BEFORE EACH LECTURE: download the pdf form of the slides, bring them to class, and look at them during the lecture. We will be projecting not only PPT but also Eclipse and other things; having the PPT slides in paper form or on your laptop/tablet can help you during the lecture.

## 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 (Recitations)

T 12:20 4 sections: 50, 24, 21, 19  
 T 1:25 2 sections: 50, 25  
 T 2:30 2 sections: 47, 12  
 T 3:35 1 section: 49  
 W 12:20 2 sections: 50, 35  
 W 01:25 2 sections: 46, 36  
 W 02:30 2 sections: 25, 25  
 W 07:30 1 section: 12

Some time EARLY, visit StudentCenter and change your section to even out the numbers

- Attendance mandatory
- Sometimes flipped: you watch videos beforehand, come to recitation and do something
- Sometimes review, help on homework, new material
- No permission needed to switch sections, but do register for whichever one you attend

## CS2111

- An “enrichment” course
- Help students who might feel overwhelmed by CS2110
- Gives more explanation of core ideas behind Java, programming, data structures, assignments, etc.
- Taught by Gries and George, 1 credit S/U
- Only for students who also take CS2110
- Only requirement: Attend weekly lecture

I would just like to thank you for taking the time to hold CS2111 this year. You have no idea how the class helped and impacted a lot of us. I would never had "survived" CS2110 without your generous share of your knowledge. I appreciated your time.

## Academic Excellence Workshops

- Two-hour labs: students work together in cooperative setting
- One credit S/U course based on attendance
- Time and location TBA
- Visit Olin 167 and ask about AEWs
- See website for more info:

[www.engineering.cornell.edu/academics/undergraduate/curriculum/courses/workshops/index.cfm](http://www.engineering.cornell.edu/academics/undergraduate/curriculum/courses/workshops/index.cfm)

## Piazza

13

- 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

14

- Book: Frank M. Carrano, *Data Structures and Abstractions with Java*, 3<sup>rd</sup> ed., Prentice Hall
  - 2<sup>nd</sup> edition is okay. E-book not required
  - Share textbook. Need access to it from time to time
  - Copies on reserve in Engr Library
- 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.
- VideoNote: videos of lectures from Fall 2015.  
<http://www.videonote.com/cornell>. Log in with netid

## Obtaining Java and Eclipse

15



- Follow instructions on our [Resources](#) web page
  - Make sure you have Java JDK 1.8, if not download and install. We explain how on the web page.
  - Then download and install the Eclipse IDE
- 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



## DrJava IDE

16



- 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). Download the jar file, not the app!!!

## Coursework

17

- 7–8 programming assignments (37%)
- Two prelims (14% – 16%)
- Final exam (30%)
- Course evaluation (1%)
- Work in recitations (1–3%)

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

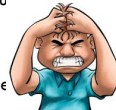
Exams are most important aspect in determining final grade

## Assignments: a real learning experience

18

- 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

One way to do an assignment:  
Wait until the day before it is due.  
Result: Frustration, anger, impatience, long lines in consulting room. No fun.  
**Not a good educational experience**

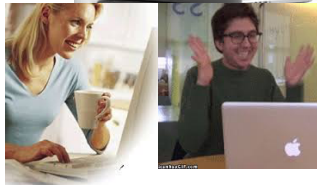


## Assignments: a real learning experience

19

One way to do an assignment:  
Read the handout immediately.  
Work on it every (other) day.  
Ponder. Look things up. Get help in  
consulting room, with no lines, or  
office hours. **Fun, hard work, a  
great learning experience**

Piano lessons:  
Practice Daily?  
Or put off practicing  
until an hour before  
weekly lesson?



## Academic Integrity... Trust but verify!

20



- 98% of you are honest and don't try to cheat
- We use artificial intelligence tools to check each homework assignment, so catch the other 2%
  - The software is accurate!
  - It tests your code and 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

21

### 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.

22

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?

## The integers as the basis

23

Leopold Kronecker (1823-1891), Prussian  
mathematician,  
Argued that arithmetic and analysis should be  
founded on the whole numbers (integers):  
*Die ganzen Zahlen hat der liebe Gott gemacht,  
alles andere ist Menschenwerk.*  
The beloved God made the whole numbers,  
everything else is the work of man.

He insisted on the constructibility of math  
objects. Real numbers —do they really exist?  
You can't compute most of them because they  
have an infinite number of digits.

God's integers!



## Type: Set of values

together with operations on them.

24

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.

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

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

Corresponding Java  
**int** x;  
x= 100;  
x= "Hello";

Illegal assignment:  
"Hello" is not an **int**

Declaration of x:  
x can contain only  
values of type **int**

## Weakly typed versus strongly typed

25

### 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

## Type: Set of values together with operations on them.

26

Java Type `int`:

values:  $-2^{31} \dots 2^{31}-1$

operations: `+`, `-`, `*`, `/`, `%`, unary `-`

$b \% c$  : remainder when  $b$  is divided by  $c$ .  
 $67 \% 60 = 7$

Java designers decided on this Principle: primitive operations on type `int` should yield an `int`.

## Most-used 'primitive' types

Inside back cover, A-6..7

27

**int**: values:  $-2^{31} \dots 2^{31}-1$

operations: `+`, `-`, `*`, `/`, `%`, unary `-`

$b \% c$  : remainder when  $b$  is divided by  $c$ .  
 $67 \% 60 = 7$

**double**: values like :  $-22.51E6$ ,  $24.9$

operations: `+`, `-`, `*`, `/`, `%`, unary `-`

Write values in "scientific notation"

**char**: values like : `'V'` `'$'` `'\n'`

operations: none

Use single quotes for type char.  
`'\n'` is new-line char

**boolean**: values: `true` `false`

operations: `!` (not), `&&` (and), `||` (or)

Can't use integers as booleans!

## About 'primitive' type `int`

Inside back cover, A-6..7

28

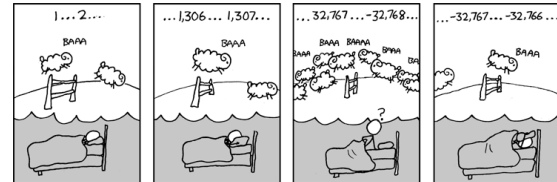
**int**: values:  $-2^{31} \dots 2^{31}-1$ , i.e.

operations: `+`, `-`, `*`, `/`, `%`, unary `-`

Java Principle: A basic operation of type `int` must produce an `int`

`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

Inside back cover, A-6..7

29

**Integer types:** **byte** 1 byte **short** 2 bytes **int** 4 bytes **long** 8 bytes

usual operators

**Real types:** **float** 4 bytes **double** 8 bytes  $-22.51E6$   $24.9$

usual operators

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

Page A-9, inside back cover

30

`(int) 3.2` casts **double** value 3.2 to an **int**

any number type

any number expression

narrow  $\xrightarrow{\text{may be automatic cast}}$  wider

**byte** **short** **int** **long** **float** **double**

$\xleftarrow{\text{must be explicit cast, may truncate}}$

`(int)` is a unary prefix operator, just like `-`

`- 3` evaluates to `3`  
`-(int) 3.2` evaluates to `-3`

## Char is a number type!

Page A-9, inside back cover

31

**char** is a number type: (int) 'V'      (char) 86

Unicode repr. in decimal: 86

'V'

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

'\u0041' is 'A'      '\u0950' is 'ॐ'      —Om, the sound of the universe  
 '\u0042' is 'B'      '\u5927' is '大'      —大衛 is (I think) a transliteration  
 '\u0056' is 'V'      '\u885b' is '衛'      of David into Chinese (Da Wei)  
 '\u0024' is '\$'

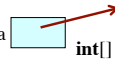
See [www.unicode.org](http://www.unicode.org)

## Basic variable declaration

Page A-6

32

**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

33

Much like in other languages — need ';' at end:

&lt;variable&gt; = &lt;expression&gt;;

**int** x;  
 x = 10;  
 ... other code  
 x = x + 1;

Have to declare x before assigning to it.

**int** x = 10;  
 ... other code  
 x = x + 1;

Can combine declaration with an initializing assignment. Shorthand for a declaration followed by an assignment.

## Assignment statement type restriction

34

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

35

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

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

/\*\* return sum of a and b \*/      Specification: in comment before function  
**public static double** sum(**double** a, **double** b) {

**return** a + b;  
 }

return type      Declarations of parameters a and b