- Introduction
- Administrivia
- Overview of high level programming languages
- Introduction to Java and Types and Type Systems in general
Course homepage (Summer 2023)

Welcome to the homepage for Cornell's intermediate-level course on computer programming and software design. The majority of materials used in the course will be available on this publicly-accessible website.

See Cornell's class roster for official meeting times and locations. Lectures and discussion sections will be delivered exclusively in person.

Prerequisites

We assume that all students in the class have prior programming experience with a general-purpose procedural language (e.g. Python, Java, ...). Suitable courses offered by Cornell include CS 1110 (this is also offered this summer) and CS 1112. Credit for CS 1110 is also offered to students who scored a 5 on the "Computer Science A" AP exam, passed the CASE exam during orientation, or took an equivalent course at another university. If you are not familiar with recursion or reference semantics (i.e. objects), then your prior experience is likely insufficient, and we recommend taking one of Cornell's introductory offerings.

The language used in this course is Java. Knowledge of Java is not a prerequisite, but we do assume that you will be able to adapt your knowledge of other languages to this setting quickly. CS 2110 focuses on generalizable design principles, algorithms, and data structures, not on the syntax and quirks of a particular language, so be prepared to do additional reading and practice at the outset if the language is new to you.

Related courses

CS 2110 is cross-listed as ENGRD 2110. These are the same exact course (same lecture, same discussion sections); it makes no difference which one you enroll in. The ENGRD label means this course can count towards the engineering distribution requirement for students in the College of Engineering (if taken for a letter grade). Whenever course staff or materials refer to "CS 2110", they also apply to "ENGRD 2110".

An honors course on object-oriented programming and data structures is also usually offered in the fall semester as CS 2112. That course covers topics in more depth, and its assignments entail writing significantly more code.
The language used in this course is Java. Knowledge of Java is not a prerequisite, but we do assume that you will be able to adapt your knowledge of other languages to this setting quickly. CS 2110 focuses on generalizable design principles, algorithms, and data structures, not on the syntax and quirks of a particular language, so be prepared to do additional reading and practice at the outset if the language is new to you.

Related courses

CS 2110 is cross-listed as ENGRD 2110. These are the same exact course (same lecture, same discussion sections); it makes no difference which one you enroll in. The ENGRD label means this course can count towards the engineering distribution requirement for students in the College of Engineering (if taken for a letter grade). Whenever course staff or materials refer to "CS 2110", they also apply to "ENGRD 2110".

An honors course on object-oriented programming and data structures is also usually offered in the fall semester as CS 2112. That course covers topics in more depth, and its assignments entail writing significantly more code.
How do I do this?

......... ??
How do I do this?

Hmm ..... ??
Maybe I should just go get ice cream ....
Solution?

Normal Person View

Search...

Programmer view

Search: Assignment 1

Syllabus

Object-Oriented Programming and Data Structures

Assignments

Instructions for the programming assignments in CS 2110 are available below. Release code for the assignments can be downloaded from the assignment's page in CMSX.

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Release date</th>
<th>Due date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment 1</td>
<td>June 20</td>
<td>June 23</td>
</tr>
</tbody>
</table>
Given a webpage as a string of characters:
Page = “<p> ...</p>
<table class="table">
<thead>
<tr> ... ”
Given a webpage as a string of characters:
Page = “<p> …</p>
<table class="table">
<thead>
<tr> ... ”

That’s easy just do page.contains("Assignment 1")
Solution?

Let’s graduate to a list of pages
Pages = [page1, page2, page3 ....]
Solution!

Let's graduate to a list of pages
Pages = [page1, page2, page3 ...]

Result = filter out pages based on the contains function.
What did we just do?

We reduced an abstract feature into two things:

- **An Algorithm**
  - The procedure to do “something”.

- **Datastructure**
  - What is your algorithm operating over?

Is our Algorithm missing anything?
Hidden Piece of The puzzle: Graphs!

Websites are actually graphs!
What is an algorithm good for?

- **Automate**
  - Can automate a given procedure based on a set of parameters
    - [e.g. I can search for other keywords]

- **Scale**
  - Can automate larger instances of the same problem
    - [e.g. Scaling the course website to a larger website]

- **Generalize**
  - Can abstract the nature of the problem across instances
    - [e.g. Can run the exact same procedure over any kind of graph]
How about now? Can we Scale?
Yes! The entire web is one big graph!
What about this other problem?

What are all the destinations that are reachable from San Francisco?
What if I tell you that you can do all this and more!
What is this class about?

- Who are you?
- Why are you here?
- What will you take away from this class?
Who Are You?

Class Demographic Quiz

CS Major or Intended CS major

Not a CS major but I do programming

I don't need to write code for any reason. But I'm here to learn.

I have no idea what I'm doing
Things this class is not about

- Can I take away something that’s useful in real life here?
Things this class is not about

- Can I take away something that’s useful in real life here?
  - You probably already know
  - If not, I hope I’ve made my case

- 6 weeks to becoming a certified Java developer?
Things this class is not about

- Can I take away something that’s useful in real life here?
  - The proof is in the cake.
  - But the dumb approach is that everyone else seems to be doing it (hedge funds, scientists, geneticists)

- 6 weeks to becoming a certified java developer
  - You will learn and use java as a tool
  - But not a java course!
Administrivia
Most of the course content should be accessible from the course website
  ○ Please read the syllabus carefully today!
● CMSx for assignments
● Ed for discussions [Please make use of this heavily!]
● Gradescope for exam feedback
## Assessment

### Basis of grade determination

Student performance will be assessed using the following elements, weighted approximately as indicated to yield an overall performance score:

<table>
<thead>
<tr>
<th>Component</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participation</td>
<td>5%</td>
</tr>
<tr>
<td>Lecture (PollEverywhere) (lowest 4 dropped)</td>
<td>2.5%</td>
</tr>
<tr>
<td>Discussion (Cooperative exercises on Wednesdays) (Lowest dropped)</td>
<td>2%</td>
</tr>
<tr>
<td>Course evaluation</td>
<td>0.5%</td>
</tr>
<tr>
<td>Bi-Weekly tests (first one is on the second week)</td>
<td>13% each</td>
</tr>
<tr>
<td>Assignments</td>
<td>31%</td>
</tr>
<tr>
<td>Assignment 1</td>
<td>4%</td>
</tr>
<tr>
<td>Assignment 2</td>
<td>5%</td>
</tr>
<tr>
<td>Assignment 3</td>
<td>7%</td>
</tr>
<tr>
<td>Assignment 4</td>
<td>7%</td>
</tr>
<tr>
<td>Assignment 5</td>
<td>8%</td>
</tr>
<tr>
<td>Final</td>
<td>25%</td>
</tr>
</tbody>
</table>
Questions for me?
Computer science is essentially the study of *data*

*Data structures* store and organize data

How do we access and manipulate data efficiently?
The way to think about this class

- The problems I want to solve are in the algorithms and my information is in the data.

- Java is just a tool
  
  A good artist befriends his tools
Course Themes

- Programming languages and paradigms
- Producing correct & maintainable software
- Organizing information in memory
- Comparing algorithm performance
Some personal advice

- I need to understand every little detail on the screen immediately
  - Bad Idea!!!

- I will get everything right away
  - Most probably not, even if you think you do, you could be wrong
    - Always test yourself with example test cases!!!
      - Unlike other courses, you can actually run your code with a single button, so don’t be shy.
What Programming Languages do we know?
Learning The Java Jargon
Java is ...

- High Level
- Compiled
- Statically Typed
- Object Oriented
Java is High Level

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

0111001010100101010011

01110010101001010100101001010011
Course Themes

- Programming languages and paradigms
- Producing correct & maintainable software
- Organizing information in memory
- Comparing algorithm performance
The Old Days

Flow charts and assembly (late 1940s)
Structured programming (1960s)

- Flow chart design, GOTO leads to “spaghetti code”
- Standardized common patterns
  - Blocks (Python indentation, MATLAB `end`, Java `{}`)
  - Control structures: `if/else, while`
  - Subroutines: `def/function`
- Discourage more general techniques
  - Unnecessary – subroutines, conditionals, and loops can compute anything
- 1968: Dijkstra's *Go To Statement Considered Harmful*
Software Systems are Big!

https://www.informationisbeautiful.net/visualizations/million-lines-of-code/
Course Themes

- Programming languages and paradigms
- Producing correct & maintainable software
- Organizing information in memory
- Comparing algorithm performance
Java is also High Level
Java is Compiled

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```
Java is Compiled

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

The Compiler!

0111001010100101010011

0111001010100101010011
Your Environment

Java code goes in here

● IntelliJ is your IDE (Integrated development Environment)
  ○ It compiles/helps maintain and can also run your Java programs
  ○ You need to set this up yourself on your computer

Java code + Magic = executable that runs on your computer

An executable runs here
Static vs Runtime

- Static is the things that happen when the compiler is generating the executable
- Runtime is what happens when the program (aka the executable) is actually running
Java is Statically Typed

- For starters, what's a type?
What does this “code” contain?

- **Data:**
  
  x, y

- **Instructions (on the Data)**

  What should be printed here?

```python
x = "hello";
y = "zee";
print(x + y);
```
What should be printed?

"hello"

"zee"

"hellozee"

""

None of the above (This doesn't run)
What about now?

```plaintext
x = 11;
y = 10;
print(x + y);
```
What should be printed now?

10
11
21
0

None of the above [Something is wrong]
What is assumed in these snippets of code?

We’re assuming how the blobs of data behave.
Java is Statically Typed

Dynamic typing (Python): Infer their types at runtime
Danger: might discover that a requested operation is invalid during the middle of running a program

Static typing (Java): expressions in source code have a type known at compile-time
Invalid operations flagged before program ever runs
Requirements For Static Typing

1. Variables must be declared with a type before use
2. Functions ("methods") must specify their return type
Types of data

<Type> <name>;              [Declaration]
<Type> <name> [=] <value>;  [Initialization]

```java
int x = 11;
int y = 10;
print(x + y);
```
Annotating Types

```java
public static void print(int z) {}
```

```java
String x = "hello";
String y = "zee";
print(x + y);
```

Red squigles are compile time errors
Types in Java

**Primitive Types**
- Built into Java
- Lower-case names (`int`)
  - Can’t be extended
  - Can’t make new ones
  - Stored efficiently
  - Matches CPU instructions
  - Building blocks of classes

**Classes**
- Names start with capital letter (`String`)
  - User-defined types
  - Complex operations
Primitive Types

**int:** values: \(-2^{31} \ldots 2^{31}-1\)
operations: +, −, *, /, %, unary −
bitwise operations: &, |, ^, ~

**double:** values like: \(-22.51e6, 24.9\)
operations: +, −, *, /, %, unary −

**char:** values like: ‘a’, ‘$’, \n
operations: none (see tomorrow’s lab)

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

\[
b \mod c : \text{remainder when } b \text{ is divided by } c.
\]

\[67 \mod 60 = 7\]

Write values in “scientific notation”

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

Can’t use integers as booleans!
Enter Objects and Classes

- You can generate your own Types in Java!!!
- Java also gives you some Object Types
- Just think of these as more complex pieces of data with customized behaviours associated with them.
- Everything that’s not a primitive type in java is an Object
Demo Time!

Let’s start looking at some code to digest these concepts
Reading / Writing code is a craft

- Even senior software engineers sometimes have trouble with complex code.

- Don’t complicate things. Always attempt to organize your understanding and simplify.
Getting closer to an actual Java program

Demo Time!
Some tricks from someone who’s been around the block a bit

- Have a tunable censor button in your head when you see jargon you don’t understand.
  - First examine the parts you do understand
  - then try to fill the gaps to the best of your ability (or as much as needed)
- Always keep a working state model of your code in your head.
  - When in doubt just start tracing the program along with the state
Deconstructing the rules

- Rules:
  1. All code must be written inside of a function ("method") – no scripts
  2. All methods must be defined inside of a "class"

Any more?
Deconstructing the rules

- Rules:
  1. All code must be written inside of a function ("method") – no scripts
  2. All methods must be defined inside of a “class”
  Any more?
  3. All data must be associated with a type
  4. All functions must be invoked with the right type.
  5. Data of one type can be converted into another type with explicit casting
More terms

Return type: double
Operator: squareAreaFromPerim
Parameter: double perim

/** Return area of square with perimeter */

double squareAreaFromPerim(double perim)
{
    double sideLength;
    sideLength = perim / 4;
    double area = sideLength * sideLength;
    return area;
}

← Specification
← Method declaration
← Variable declaration
← Assignment
← Initialization
← Return statement

Color code:
Keyword (blue)  Type (orange)  Literal (yellow)
Notes On Casting

This expression

25.2 + 1

should evaluate to

26.2

25.2: type is double

1: type is int
Notes On Casting

narrow  wider
byte   short  int  long  float  double
char   int

byte-to-short, short-to-int, int-to-long, float-to-double — no info is lost.

Java automatically converts narrow-to-wider. Note: int-to-float or long-to-double can cause loss of information, but Java will do them automatically anyway.

Called casting.
Method Signatures

- **Signature** of method:
  - **name**, and
  - **argument types**, but
  - not its return type

- **Method**: `void reset() { count= 0; }
**Signature**: `reset()`

- **Method**: `void reset(int i) { count= i; }
**Signature**: `reset(int)`
Notes On Overloading

Class Math, which comes with Java, has lots of methods for basic numeric operations.

```java
int abs(int a) { ... }
long abs(long a) { ... }
double abs(double a) { ... }
float abs(float a) { ... }
```

Space used: remember, a byte is 8 bits.

```
int: 4 bytes      long: 8 bytes
float: 4 bytes    double: 8 bytes
```
Next Lecture

- More about Classes and Objects
- Some features of Java, Setting up your IDE and getting started with A1
Homework for today

- Think about whether this class is for you at this time
  - Where are you in your journey as a programmer/computer scientist?
  - What are your goals for the summer? (practical and personal)
  - How much time do you actually have on you?

- If yes, read the whole syllabus very carefully
  - Make notes if necessary
  - Take your calendar and add all the deadlines on there (customize your schedule if that helps)
  - Bonus: Read transition to Java and Set up IntelliJ