CSCI 1300

Intro to Computing

Gabe Johnson

Lecture 18 Feb 25, 2013

Java!

Lecture Goals

1. Java

(that’s it!)

Upcoming Test

Exam 2 **Friday, March 1**

Exam 2

This will cover everything up to and including last Friday's lecture (except subclasses, which we'll get to with Java). Old material is fair game! New stuff includes dictionaries, while loops, boolean operations, classes, and objects. There are sample questions up on GitHub.

Java: Same Concepts

In Java, we have almost the same set of concepts that we had in Python:

Variables
Flow Control
Boolean Arithmetic
Lists
Functions
Classes

… with added ‘stuff’

Some may replace ‘stuff’ with *complexity*, or *features*, or *requirements*, or *hurdles*. Depends on your outlook, what you’re doing, and who you’re doing it with.

Semicolons! Curly Braces!

In Python, the whitespace told the interpreter where the statements ended, and how they were grouped together.

In Java, we end statements with semicolons and group things together with curly braces.

Like the following...

Java Example

```
int getSumOfPostitiveCubes(int a, int b) {
   int sum = 0;
   if (a > 0) {
      sum = sum + (a * a * a);
   }
   if (b > 0) {
      sum = sum + (b * b * b);
   }
   return sum;
}
```

Whitespace doesn't matter

Java doesn't care (mostly) about whitespace. You have to separate tokens with spaces, but it doesn't matter if it is a space, a tab, a newline, or some strange combination of the two. These are the same:

```
public int
x
(List myList)
{
  return myList.size();
}
```

```
public int x(List myList) {
    return myList.size();
}
```

Python is *dynamically typed*

In Python you can declare an initialize something by just giving it a value. The type is implicit. And you can change it.

```
x = 4
y = x * x
x = “Wheee!”
print x
```

Java is *statically* typed

You have to explicitly tell the Java compiler what kind of item each variable is. Also: you can't change types once you've declared something.

```
int x = 4;
int y = x * x;
String otherThing = “Wheee!”;
System.out.println(otherThing);
```

Source File Format

Java source files must be named in this format:

MyThing.java

This guarantees that the file contains a class called MyThing. Not *my_thing*. Not *mything.* Not *myThing*. Capitalization matters. It must be a class called *MyThing*.

MyThing example

```
// MyThing.java
public class MyThing {
  // statements go here
}
```

Java Compiler

Java uses a *Compiler*, and a separate *Interpreter.*

The compiler converts source files like **`MyThing.java`** into bytecode output with the same name but using the '.class' suffix. MyThing.java compiles into **`MyThing.class`**.

```
javac MyThing.java
```

>> outputs MyThing.class

Java Interpreter

To run a class file, we run it through the interpreter called `java` (lower case j).

The class must contain a special function called ‘main’ in order to run.

```
java MyThing
```

(notice: there’s not a .class at the end. It is just MyThing)

Main Function

The entry point to the program has the signature:

```
public static void main(String[] args) {
    // your code here
}
```

Ignore the extra stuff for now

```
public static void main(String[] args) {
    // your code here
}
```

This syntax is new to you right now, but there are actually concepts you already know. We just type it differently.

This is a function called ‘main’. It takes an input param called ‘args’, and returns nothing.

Coding Session

In the coding session we will look at similarities between Python and Java:

- data types: ints, Strings
- flow control: for-loops
- functions
- running programs