Computer Science 121

Intro to Programming:Java - Lecture 1

This is an introductory programming class in Java / 4 credits

Professor Robert Moll (+ TAs) CS BLDG 276 - 545-4315
moll@cs.umass.edu

Course home page:

http://twiki-edlab.cs.umass.edu/bin/view/Moll121/WebHome

enrolling: try to get everyone in.. See me after class

Special questions? .. after class.

Questions - Computer Access?
Course Materials

No textbook.

We’re using a (free!) electronic book - iJava

Additional features: we’ll run the course using a “Wiki” website

OWL - heavyweight assignment system

IDE - Dr Java is the class’s integrated development environment
Is this the right class for you?

Do you know how to program in any language? (if you already know some Java, consider 191B..).

One programming class only, for sure: do CS 191P

Do you know your way around your computer? (RAM, downloading, text files, applications, spreadsheets, secondary storage, byte, Internet, www...) If many of these terms/concepts are a stretch, consider taking CS 105, CS 120, CS 145..

How’s your math? You need to be comfortable with basic math, logic, compound interest, simple logic.

Do you *want* to take this class? If you’re here because you need an R2, you’ve got the wrong course. This one’s too hard.

Do an easier R2. (Also - if you’re a computer art student, see me after class)
Who are you? FR / Soph / Jr / Sr / Grad / Staff / HS

Major or probably major:
- CS / Sciences / Engineering / Soc Sci - Hum / SOM

Interested in the IT-Minor?

Never / ever programmed? Some Java / C++ / VB / Other
More Administration, Details

The grading formula:

- Embedded Questions: 5%
- Freestyle Projects: 5%
- Programming assignments: 20%
- OWL assignments: 15%
- Midterm exam: 20%
- Final Exam: 35%

Also: to get C in the course, you must get at least a C on final!

Collaboration - unless otherwise stated: conceptual collaboration ok, do your own coding (more on this later)

System: You’ll need Java 1.5 You’ll need Dr Java. Information about this online
Agenda

understand, appreciate object-oriented programming, its aims, methods, and (we hope) joys;
become a skilled beginning Java programmer
Learn about some additional aspects of the computing landscape

Success?
Software is a hybrid endeavor..

Who fails, and why.. phrasebook Java
don’t know what you don’t know (come to class!)
In general CS is as much about technique as it is about brains
iJava - how to succeed… the flaw in working backwards
Layout of the Course

Tuesday: a big-picture lecture day
Thursday: lecture, fancy topics, much interaction
Wednesday - Discussion - will be just that: discussion, + a fast-paced interactive problem-solving experience.
You are expected to attend class - but roll isn’t taken.
Miss classes and lectures at your own peril.
If you come, bring your brains.
The pieces of the class

The textbook - narrative; embedded questions
The website
OWL weekly homework
Programming projects
Freestyle projects
Two exams
Lecture
Discussion
Office hours
Computer Programming

High Level Languages- human-oriented: accessible syntax; built-in conceptual decomposition

High level languages require language translation

What Java brings..

- Object-oriented
- Controlling complexity
- Hygiene
- Recycling (object-oriented - main currency: objects)
- Machinery for dealing with web (November!)
- Hardware neutral (more or less)

But note: -----> Java is hard: it’s for pros; it’s a language with a message
Hardware / Software

Hardware is easy - it’s the physical computer - the chips, the buses, and so forth.

Software is more subtle - it’s the pattern of instructions that directs the hardware. Think of:

Knitting
Origami
Driving directions
Chili recipe
Early model of a computer program: roughly speaking, a sequence of instructions for shopping:

go to store

buy milk

if bananas cost < 50 cents, buy 6

pay

pay

come home

....

The Java model is more like a library of how-to books with blueprints: e.g., how to frame a house, how to install its windows, etc.

Each volume gives sequences of instructions for doing specific jobs..
public class Howto{
    // a baby intro example
    public static void main(String args[]) {
        System.out.println("Welcome to 121");
        System.out.println("3 + 5");
        System.out.println(3 + 5);
    }
}

Notice: this Java program (application) consists of a single class
That class consists of a single (main) method (subprogram)
That method consists of three “write something to the console” statements, or instructions
Languages, Translators, and Computing

Our little program is actually incomprehensible to a computer.

Machine language instructions are VERY primitive.

One aspect of computer science focuses on the translation process -- most importantly, how can a language for programming that’s fairly natural for humans (e.g. Java) be faithfully converted into machine language, a seemingly patternless sea of incomprehensible gibberish?

For languages such as Java, the translator is called a compiler.

01011010101011 01011010101011 <--- machine language!
11010101101010 10000000000011 ...

11010101101010 10000000000011 ...
Syntax and Semantics In English

“Tomorrow I’ll come”, and “I’ll come tomorrow” mean the same thing (are semantically the same), even though they’re syntactically different (the parts have been rearranged).

This is a big deal for linguists. Also: natural language (English) is often semantically ambiguous:

“Jay doesn’t kiss his wife because he’s worried about her”

Computer languages are far less flexible. The rules of form for Java (say) are absolutely precise and may not be messed with.

The semantics, likewise, of a Java statement, are unvarying (a println statement prints!).
Errors

- Compile-time errors - syntax, type errors
- Run-time errors - divide by 0
- Logic errors - Everything works fine - get wrong answer

```java
System.out.println(5 + 3; System.out.println(5/0);

System.out.println(“area = “ + “ “ + (3 * radius * radius));
```
Assignment for Thursday

1) Have a look at the course web site
   http://twiki-edlab.cs.umass.edu/bin/view/Moll121/WebHome

2) Download Java (JDK), and DrJava (IDE)

3) Get your OWL/textbook account going: instructions on website
   (look under “TextbookStart”)

4) Read the Preface, and Chapter 1

(Note: this is a lot. Do as much as you can, bring your questions to “discussion” on Thursday)