CS 121 – Intro to Programming:Java - Lecture 1
Professor Robert Moll (+ TAs)
CS BLDG 278 - 545-4315
moll@cs.umass.edu

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

enrolling – I’ll try to get everyone in.. Special questions? See me after class. Honors section - see me after class

Questions - Computer Access? Are you or are you considering a CS major? Freshman?
Course Materials:

No textbook. We’re using an electronic book that I am in the process of creating.

Additional features:

    the Wiki - the web world for course administration

        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 do, consider 191B..)

• Do you know your way around your computer? (RAM, downloading, text files, applications, spreadsheets, Excel, secondary storage, byte, Internet, www…) If many of these term/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
More Administration

Section issues: The class is organized into four discussion sections, and there is a separate 1 credit honors section, which will meet at a time TBA

The grading formula:
- Programming assignments: 20%
- OWL assignments: 15%
- Midterm exam: 25%
- Final Exam: 40%

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

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

System: You’ll need Java 1.5 (sometimes called 5.0!). And you’ll need Dr Java. Information about this online
(Secret) Agenda

understand, appreciate object-oriented programming, its aims, methods, and (we hope) joys.
become a skilled beginning Java programmer

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 about brains

passivity: this course in **not** a big-needle shot of knowledge in the head
The layout of the course: Tuesday: a big-picture lecture day

Thursday - the so-called discussion class - 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.

Bring your brains.

For next Tuesday: read ch 1 of the text, do the first owl assignment (this will all be explained on the wiki)
Computer programming - the short version

High Level Languages - human-like; concept of a subprogram

Language levels, language translation

The road to Java:

  Controlling complexity - hygiene, recycling (object-oriented - main currency: objects)

  Machinery for dealing with web

  Hardware neutral (more or less)

  *

  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:

Knitting

Origami

Travel instructions

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

    come home

The Java model

    more like a collection of how-to books with blueprints:

    e.g., how to build a shed

    contains lots of lists describing how to get things done..
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

...
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 she’s modest”

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 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

```
System.out.println(5 + 3;
```

```
System.out.println(5/0);
```

```
System.out.println("area = " + " " + (3 * radius));
```
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);
    }
}

The Wittgenstein Joke Principle:

“Smiles is the longest word in the world”