CS 121 - Intro to Programming: Java - Lecture 2

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

Read chapters 1, 2 - See CourseWork link at website
Embedded problems due Wed; OWLs due Friday
First prog assignment up soon, also due next week
Survey is up: please complete!

Friday drop-in site: TBA
Course Materials:
No textbook. We’re using a (free!) electronic book (with embedded problems)

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?
• know how to program? (191P)

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

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

• Do you want to take this class? This is the hardest R2.
More Administration

The class is organized into five discussion sections.

The grading formula:

- Embedded Questions: 8%
- Programming assignments: 20%
- OWL assignments: 15%
- Midterm exams: 24%
- Final Exam: 33%

Also: to get C in course, you must get C on final!

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

System: You’ll need Java 1.5 or 1.6, + IDE (DrJava)
The layout of the course

General Pattern will be Tuesday/Thursday Lectures on a subject... then Monday discussion

There are 3 kinds of work in the course (+3 exams):

• Embedded Questions
• OWL Hwk Problems
• Programs

CourseWeek link on website gives work due in the coming week or two
Key link: CourseWeek
The pieces of the class
The textbook
The IDE - Dr Java
The website (has: office hours, assignments, etc)
OWL weekly hwk
Programming projects
2 midterm exams / a final exam
Lecture
Discussion
Office hours - TBA
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

Fact of Life: in computer world, hardware can vary!
Why Java?

• Designed for big projects, complexity control

• Machinery for programming the web

• Hardware neutral (more or less)

• Object-oriented means: main currency are objects (rather than simply statements)

• Objects are realizations of blueprints (classes); realizations called objects

• There are libraries of thousands of these, waiting to be recycled
A typical Java program
A very, very simple application

Class G

main
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);
    }
}

Prints:
Welcome to 121
3 + 5
8
Languages, translators, and computing

Program is actually incomprehensible to a computer

How can a language for programming that’s fairly natural for humans (e.g. Java) be faithfully converted into machine language gibberish?

For languages such as Java, the translator is called a compiler - but wait: there’s more.

01011010101011
01011010101011 <--- machine language!
11010101101010
100000000000011
...

There’s a problem:

My machine language (a Mac, with a PowerPC G4 processor)
Is different from your machine language
(a Windows machine, with an Intel processor)

Can a program that’s been compiled on my machine run on your machine (which has a very different machine language)??
Two translation styles

Compiler (War & Peace, Russian -> English)

Interpreter (Simultaneous translator at the U.N.)

Compiled translation presents problems:

a compiled program on my computer may not run on your computer - my machine language may be different from your machine language.

Java has a solution..

Two step translation from Java source code to machine level object code
Step 1: Source code is compiled into a universal machine language called bytecode.

Step 2: Each machine invokes its special interpreter - its JVM (Java Virtual Machine) - for bytecode, which produces that computer's running machine code.
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));
How + works

3 + 5  ->  8

“three” + “five”  -> threefive

“three” + 5  -> three5

“three “ + 5  -> three 5

“three” + (5 + 4)  -> three9

“three” + 5 + 4  -> three54

5 + “three” + 4  -> 5three4
Java's Object Model

This subject will occupy us for a good deal of the next month or so!
The Object Model

We model “things” as objects

Objects have attributes, and behaviors

Trip (attributes: start, end, days, distance)
  (behaviors: getDays, setDays, distPerDay ..)

Horse (attributes: name, breed, age, height)
  (behaviors: getAge, setAge ..)

Car

House

Student

Tree
State

• Objects have state
• State is situational
• Sometimes behaviors change state
• Sometimes behaviors merely report state
Where are all of these characteristics written down?

Answer: In a class definition

A class definition for an object is different from an object, in the same way that the blueprint for a house is different from a house.

Still -

How do you make an object?

How do you invoke its behaviors?
Infant objects

Attributes

- name
- age (in months)

(These represent the state of an Infant)

Behaviors

- getName
- getAge

- anotherMonth (make kid one month older)
public class InfantTester{

public static void main (String[] args){
    Infant myKid = new Infant("Kit", 4);
    System.out.println("name: " +
                        myKid.getName());
    myKid.anotherMonth();
    System.out.println("my kid is now " +
                        myKid.getAge());
}
}
Broadly, two kinds of behaviors:

Get behaviors -
• How old is the kid
• What’s the kid’s name
• Get behaviors do NOT change the calling object

Mutating behaviors - Alter calling object (alter state)
• Name change
• Age change

When a mutator does what it does, the state of the calling object changes.
public class Infant{ // the Infant class definition

    private String name;
    private int age; // in months

    public Infant(String who, int months){
        name = who;
        age = months;
    }

    public String getName(){return name;}

    public int getAge(){return age;}

    public void anotherMonth(){age = age + 1;}
}
public class Infant{

    private String name;
    private int age;    // in months

    public Infant(String who, int months){
        name = who;
        age = months;
    }

    public String getName(){return name;}

    public int getAge(){return age;}

    public void anotherMonth(){age = age + 1;}
}

Attributes

Constructor
public class Infant{
    pri String name;
    pri int age;
    public Infant(...){
        name = ..
        age = ..
    }
    pub int getName(){
        return name;
    }
}

public class InfTes{
    public static ....{
        Inf myKid =
        new Infant(.);
        S.o.p.
        myKid.getName());
    }
}
What you should be working on?

Follow CourseWeek - read, do embedded, OWLs for ch 1,2

UNIX for beginners, CS 197U, MW 4-5 four wks, 1 credit
Assignment Statements and Identifiers

An identifier is the name of a variable (or method, or class...)

```java
int number = 7; // number now "holds" 7
number = 4; // number now "holds" 4
number = number + 2; // number now "holds" 6
```

Assignment is NOT equality!

Assignment is an **action** operator: **Compute** the RHS, then **copy** the result to variable named on the LHS