

CS 4: Lecture 1
 Wednesday, January 18, 2006
 Prof. Jonathan Shewchuk, jrs@cory.eecs
 TA: Clint Ryan, ryanc@cs
 Email to prof & TA at once (preferred): cs4@cory.eecs

Handout: Course Overview (also available from CS 4 Web page)

CS 4 (aka CS 39L) is an introduction to programming, like CS 3 or E 77, for students with no (or hardly any) programming experience. Unlike CS 3, most of our programming examples are drawn from engineering and science. Unlike CS 3 or E 77, CS 4 teaches you Java as your first computer language.

Read the Course Overview and the CS 4 Web page as soon as possible!

>>> <http://www.cs.berkeley.edu/~jrs/4> <<<

YOU are responsible for keeping up with the readings and assignments. I won't always remind you. Today's reading: Chapman, Sections 1.1, 1.3-1.6, 2.10
 Finally, the course newsgroup is REQUIRED reading: ucb.class.cs4

Labs

Labs (in 277 Soda) start tomorrow. There are two labs each week:

Mondays 6:30-8:30 pm

Thursdays 5:00-8:00 pm

Attendance (in both) is mandatory. At the beginning of each lab, we will assign you a random partner (different every lab), and you will work together with your partner to complete the lab.

Obtain a keycard from the CS main office as soon as possible! Soda Hall is locked after 6:30 pm.

Please buy a USB Memory Stick as soon as possible to save your labs and homeworks. They can usually be had for less than \$20. We'll also provide server space for you to save your files, but it's important to have a backup--every student nukes an important file at least once a year.

Textbook

Stephen J. Chapman, Java for Engineers and Scientists, Second Edition, Pearson Prentice Hall, 2004. ISBN # 0-13-033520-7. The first edition is not recommended.

Grading

40 pts	Labs & lab quizzes	380-400: A+	360-379: A	340-359: A-
80 pts	Homeworks	310-339: B+	290-309: B	260-289: B-
40 pts	Project	240-259: C+	220-239: C	200-219: C-
100 pts	Midterm	170-199: D	0-169: F	
140 pts	Final Exam			

 400 pts There is NO CURVE. CS 4 is not a competition.
 Late homeworks and labs will not be accepted.

Cheating ...will be reported to the Office of Student Conduct.

- 1) "No Code Rule": Never have a copy of someone else's program in your possession and never give your program to someone else.
- 2) Discussing an assignment without sharing any code is generally okay. Helping someone to interpret an error message is another example of permissible collaboration. However, if you receive a significant idea from someone, acknowledge them in your assignment.
- 3) No discussion whatsoever in exams, of course.
- 4) In pair assignments, you share code freely with your partner, but not with other partnerships.

JAVA

====

A computer program consists of `_instructions_` (often just called `_code_`). Your computer's `_processor_` (or `_CPU_`, for central processing unit) normally just walks through the instructions and executes them in the order they're written. (There are instructions that change the order in which the processor executes instructions, but we'll learn about those later.)

One of the simplest things you can do in Java is write an instruction that prints a number on the screen.

```
System.out.println(3);
System.out.println(2 + 7);
```

When the processor executes this code, it prints:

```
3
9
```

Observe that the two instructions are executed in order--the top one, then the bottom one. The command `"System.out.println"` is regrettably long and awkward. This should give you good motivation to learn how to cut-and-paste in your text editor.

Expressions

The invention of modern computers was motivated largely by the desire to do mathematical calculations very quickly--for instance, to break the ciphers of the German Luftwaffe and navy during World War II. The computer science equivalent of the Nobel Prize, the `_Turing_Award_`, is named after Alan Turing, who designed computers to do exactly that.

Java tells your processor to do calculations when you write expressions, like the `"2 + 7"` above. You can use your computer and Java as a bulky calculator.

```
System.out.println(12 / 4 * (9 - 3) + 5);
```

Here, `"/"` means division and `"*"` means multiplication. We'll learn more operators later.

Variables

If you've ever owned an MP3 player or digital camera, you know that computers have memories (which are not, sadly, infinitely large). Java lets you store numbers in memory so you can save them for later use. But first you have to ask Java to assign you a piece of memory. There are instructions for that, called `_declarations_`. Here's an example:

```
int dollars;
```

This declaration does two things.

- (1) It allocates a chunk of memory big enough to store an integer, which Java calls an "int".
- (2) It gives the chunk of memory a name: "dollars". Any time you want to store an integer in that memory, or check what integer is stored there, you have to refer to the memory chunk by name.

"dollars" is called a `_variable_`, a chunk of memory with an associated name.

After you declare a variable, you can store a number in it by writing an `_assignment_statement_`.

```
dollars = 3;
```

This looks like mathematical equality, but that's misleading. It's really an instruction that says, "Java, please store the integer 3 in the variable named 'dollars'." The fact that Java uses an equal sign for assignment is a design flaw that confuses many students. (Java inherited this flaw from the language C. No computer language is without flaws.)

What does this Java code print?

```
int size;
size = 6;
System.out.println(size);
size = 3 * size;
System.out.println(size);
```

Remember that these lines of code are executed in order from top to bottom. The first line creates a variable named "size". The second line stores the integer 6 in that variable.

```
-----
| 6 | size
-----
```

The next line prints the value of size--that is, it prints "6" on the screen.

The fourth line is tricky, because it both reads and writes the variable "size". Let's look at that line again:

```
size = 3 * size;
```

There are two things you must understand:

- (1) Java evaluates (computes) the expression "3 * size" first, before it performs the assignment.
- (2) When Java stores the result in size, the new value erases the old value completely.

So Java `_reads_` the value of the variable "size", which is currently 6. Then Java multiplies it by 3, giving 18. Then the assignment writes the result 18 into the variable "size", erasing the old value of 6.

```
-----
|18 | size
-----
```

The fifth line prints "18" on the screen. So the output is:

```
6
18
```