

last time in cs1007...

- course objectives
- policies
- academic integrity
- resources
 - WEB PAGE: *<http://www.columbia.edu/~cs1007>*
- **NOTE CHANGES IN ASSESSMENT — 5 EXTRA CREDIT POINTS ADDED**

recitations.

- sign up for a recitation (sign up sheet circulating in class)

day	date	time	room
recitation #1	Mon	9.00am – 10.00am	252 ET – Dali
recitation #2	Mon	7.00pm – 8.00pm	407 Math – Steve
recitation #3	Mon	8.00pm – 9.00pm	407 Math – Jonah
recitation #4	Tue	7.00pm – 8.00pm	407 Math – Don
recitation #5	Wed	12.00pm – 1.00pm	252 ET – Lyndon
recitation #6	Thu	6.00pm – 7.00pm	407 Math – Min

- ET = Engineering Terrace
- starting next week (week of Sep 9)
- this is where you will get back your homeworks!!

today's topics.

- computer basics
- creating your first program
- editing, compiling, linking, running
- output
- data types
- reading: *ch 1*, *ch 2.1-2.4*

computer commands.

- computer follows commands
commands = series of instructions
- you will learn how to *command* a computer
command = program = write instructions
- you understand the commands,
but does the computer?
that's a question of cognition...
→ Artificial Intelligence, Cognitive Science

computer components.

- computer = hardware + software
- a computer is organized into *logical units*:
 - input
 - output
 - memory
 - arithmetic and logic (ALU)
 - central processing (CPU)
 - secondary storage

computer instructions.

- set of instructions = *program*
- types of instructions:
 - machine language
 - assembly language
 - high-level language (e.g., C, C++, Java)
- program is *compiled* into machine language and then *executed* (*ran*)
- *executing* (*running*) *program* = *job* = *process* = *task*

machine language.

- lowest level
 - numeric
- computer is comprised of zillions of *switches* or *relays*
 - switches = ON or OFF
 - relays = OPEN or CLOSED
- hardware position is abstracted into software as 1's and 0's
- 1's and 0's \Rightarrow *base 2*, or *binary*

assembly language.

- medium level, but still pretty low; i.e., hard to read and understand
- “English” words and abbreviations
- examples:

LOAD

ADD

SHIFT

STORE

high-level languages.

- examples: C, BASIC, FORTRAN, Pascal, C++, Java, LISP, Scheme
- even more like “English”
- high-level languages are
 1. *compiled* into machine language or *object code*
 2. *linked* into *executable code*
 3. *executed* or *ran* as programs

language examples.

- machine language:
+1300042774
+1400593419
+1200274027
- assembly language:
LOAD BASEPAY
ADD OVERPAY
STORE GROSSPAY
- high-level language:
 $\text{grossPay} = \text{basePay} + \text{overTimePay};$

Java.

- Java is an *object-oriented* language: it is structured around *objects* and *methods*, where a method is an action or something you do with the object
- Java programs are divided into entities called *classes*
- some Java classes are *native*
but you can also write classes yourself
- Java programs can run as *applications* or *applets*

your first application.

“hello world”

- typical first program in any language
- output only (no input)

the application source code.

file name = hello.java

```
/*-----  
   Prof Sklar, 05-Sep-02, hello.java  
  
   This class demonstrates output from a Java application.  
-----*/  
public class hello {  
    public static void main ( String[] args ) {  
        System.out.println( "hello world!\n" );  
    } // end of main()  
} // end of class hello()
```

output.

- like filling in graph paper

- *methods*

```
System.out.println( )
```

```
System.out.print( )
```

- *arguments*

- those things inside the parenthesis ()
- one or more Strings, separated by “+” ’s
- escape sequences: \n, \t
- also called *parameters*

- example

```
System.out.println( "The quick" + ", brown " + "fox" );
```

things to notice.

- Java is CASE sensitive
- punctuation is really important!
- *whitespace* doesn't matter for compilation
- *BUT* whitespace DOES matter for readability and your grade!
- file name is same as class name

try it yourself.

1. log into CUNIX
2. create the application source code file,
using the *emacs* editor
3. compile the source code,
using the *javac* command
4. execute the program using the *java* command

quick and dirty UNIX.

- UNIX is an operating system,
 - *Linux* is a version of UNIX
- command-line interface
 - commands have options, also called *switches*
- here are some commands:

```
ls      -- list the files in the current directory
cp      -- copy a file
mv      -- rename a file
rm      -- delete (remove) a file
cd      -- change directory
pwd     -- show the current directory
man     -- help
chmod   -- change file protections
```

quick and dirty *emacs*.

- at the UNIX prompt: `unix> emacs hello.java`
- emacs is a “control key” editor
- here are some commands:

<code>Ctrl-B</code>	-- move cursor back
<code>Ctrl-F</code>	-- move cursor forward
<code>Ctrl-P</code>	-- move cursor to previous line
<code>Ctrl-N</code>	-- move cursor to next line
<code>Ctrl-D</code>	-- delete character under cursor
<code>Ctrl-X Ctrl-S</code>	-- save the file
<code>Ctrl-X Ctrl-C</code>	-- exit emacs
<code>Ctrl-H</code>	-- help
<code>ESC</code>	-- escape! gets you out of trouble!

data types.

- programs = objects + methods
- objects = data
- data must be *stored*
- all storage is numeric (0's and 1's)


memory.

- think of the computer's memory as a bunch of boxes
- inside each box, there is a number
- you give each box a name
⇒ defining a *variable*
- example:

program code:

```
int x;
```

computer's memory:

x → 

variables.

- variables have:
 - name
 - type
 - value
- naming rules:
 - names may contain letters and/or numbers
 - but cannot begin with a number
 - names may also contain underscore (_) and dollar sign (\$)
 - underscore is used frequently; dollar sign is not too common in Java
 - can be of any length
 - cannot use Java keywords
 - Java is *case-sensitive!!*

primitive data types.

- numeric

byte	8 bits	$-128 = -2^7$	$127 = 2^7 - 1$
short	16 bits	$-32,768 = -2^{15}$	$32,767 = -2^{15} - 1$
int	32 bits	-2^{31}	$2^{31} - 1$
long	64 bits	-2^{62}	$2^{63} - 1$
float	32 bits	$\approx -3.4\text{E}+38, 7 \text{ sig dig}$	$\approx 3.4\text{E}+38, 7 \text{ sig dig}$
double	64 bits	$\approx -1.7\text{E}+308, 15 \text{ sig dig}$	$\approx 1.7\text{E}+308, 15 \text{ sig dig}$

- boolean

boolean	1 bit
---------	-------

- character

char	16 bits
------	---------

assignment.

- = is the assignment operator
- example:

program code:

```
int x; // declaration  
x = 19; // assignment
```

or

```
int x = 19;
```

computer's memory:

x → 19

to do.

- get the textbook, and read chapter 1 and 2.1 – 2.4
- sign up for a rectiation
- try logging into your CUNIX account
- check out the class web page:
<http://www.columbia.edu/~cs1007>

Have a good weekend!