Welcome to CS61B!

be signed up for a lab and discussion section using the s poll, available from the course website. If you can't attend any section you can (although you have second seating).

oday. In (or preferably before) lab this week, get a account from `https://inst.eecs.berkeley.edu/webacct`.

will be crowded, you might want to bring your laptop.

o work from home, try logging in remotely to one of the servers.

g Piazza for notices, on-line discussions, questions.

rmation about the course is on the home page (grading, ating policy, etc.).

be screencast.

Crowding

, I don't if we will be able to admit any Concurrent En- dents. If you choose not to take this course please drop possible for the benefit of others (the add/drop dead- otember—6 September if you wish to avoid a fee).

Texts

vo readers currently on-line (see the website).

without printed versions, but might want to print out tions for exams (since we don't allow computers in tests).

or first part of the course only) is *Head First Java*. It's but has the necessary material.

Course Organization I

illustrate.

portant: exercise of programming principles as well as ty details go there. Generally we will give you homework oing them.

important, but really not graded: use it as you see fit ! You get points for just putting some reasonable effort

ojects are *really* important! Expect to learn a lot. Projects efforts (that's for later courses).

Course Organization II

is part of the course. Programming takes place in a *environment:*

diting, debugging, compilation, archiving versions.

, I keep it simple: Emacs + gjdb + make + git, (doc- in one of the readers and on-line). But we'll look at lab, and Eclipse is OK, too.

allenging: better to stay on top than to cram.

Projects, 50%; HW, 10%

ell us!

Programming, not Java

rn *programming,* not Java (or Unix, or Windows, or. . .)

principles span many languages

connections.

+y *vs.* `(+ x y)`) is superficial.

non, and Scheme have a lot in common.

use GUIs, text interfaces, or embedded systems, im- s are the same.

For next time

Chapter 1 of *Head First Java*, plus §1.1–1.9 of the on-line *Reference*, available on the class website.

erview of most of Java's features.

ooking at examples on Friday.

mber the questions that come up when you read some-ign:

s? We might have made a mistake.

to ask at the start of lectures, by email, or by Piazza.

Acronyms of Wisdom

DBC

RTFM

uick Tour through the First Program

ould write

```
al first program
o, world")
```

```
al first program.
 N. Hilfinger */
ello {
greeting. ARGS is ignored. */
tic void main(String[] args) {
m.out.println("Hello, world!");
```


Commentary

```
l first program.
 N. Hilfinger */
ello {
greeting. ARGS is ignored. */
tic void main(String[] args) {
m.out.println("Hello, world!");
```

nts can either start with '//' and go to the end of the n Python), or they can extend over any number of lines, y '/*' and '*/'.

he '//' comments, except for things that are supposed ed, and our style checks will flag them.

multiline kind of comment includes those that start with re called *documentation comments* or *doc comments*.

on comments are just comments, having no effect, but s interpret them as providing documentation for the ollow them. They're generally a good idea and our style re them.

Classes

```
l first program.
 N. Hilfinger */
ello {
greeting. ARGS is ignored. */
tic void main(String[] args) {
m.out.println("Hello, world!");
```

on and variable in Java is contained in some *class*.

ke Python's classes, but with (of course) numerous dif-detail.

n turn, belong to some *package*. The `Hello` class belongs *mous package*.

ned packages later,

Methods (Functions)

```
l first program.
 N. Hilfinger */
ello {
greeting. ARGS is ignored. */
tic void main(String[] args) {
m.out.println("Hello, world!");
```

aders in Java contain more information than those in y specify the *types* of values *returned* by the function *parameters* to the functions.

oid has no possible values; the *main* function here re-g. The type `String` is like Python's `str`. The trailing '`[]`' *of*. Arrays are like Python lists, except that their size created.

takes a list of strings and returns nothing.

med "main" and defined like the example about are spe-re what get called when one runs a Java program (in main function is essentially anonymous).

Selection

```
l first program.
  N. Hilfinger */
ello {
 greeting. ARGS is ignored. */
tic void main(String[] args) {
.out.println("Hello, world!");
```

$\mathcal{E}.N$ means "the thing named N that is in or that applies identified (or computed) by $\mathcal{E}$."

`m.out`" means "the variable named '`out`' that is found in ned '`System`'."

`stem.out.println`" means "the method named '`println`' to the object referenced by the value of variable '`System.out`'."

Access

```
l first program.
  N. Hilfinger */
ello {
 greeting. ARGS is ignored. */
tic void main(String[] args) {
.out.println("Hello, world!");
```

ods and variables are "one-of" things.

hod is just like an ordinary Python function (outside of a function in a Python class that is annotated `@staticmethod`.

iable is like a Python variable defined outside of any riable selected from a class, as opposed to from a class

bles are local variables (in functions) or instance vari- ses), and these are as in Python.

Access

```
l first program.
  N. Hilfinger */
ello {
 greeting. ARGS is ignored. */
tic void main(String[] args) {
.out.println("Hello, world!");
```

ed entity in Java has *access permissions* indicating what de may mention it.

, *public* classes, methods, and variables may be referred else in the program.

es refer to them as *exported* from their class (for arialbles) or package (for classes).