Welcome to CS61B!

- You should be signed up for a lab and discussion section us
  SignUpGenius poll, available from the course website. If y
  find a slot, attend any section you can (although you have
  priority for seating).
- Labs start today. In (or preferably before) lab this week
  CS61B Unix account from `https://inst.eecs.berkeley.edu`
- Because labs will be crowded, you might want to bring your l
- If you plan to work from home, try logging in remotely to on
  instructional servers.
- We'll be using Piazza for notices, on-line discussions, questio
- General information about the course is on the home page (
  lateness, cheating policy, etc.).
- Lectures will be screencast.

Crowding

- At this time, I don't if we will be able to admit any Concurr
  rollment students. If you choose not to take this course plea
  it as soon as possible for the benefit of others (the add/dro
  line is 18 September—6 September if you wish to avoid a fe

Texts

- There are two readers currently on-line (see the website).
- You could do without printed versions, but might want to p
  selected portions for exams (since we don't allow computers i
- Textbook (for first part of the course only) is *Head First J*
  kind of silly, but has the necessary material.

Course Organization I

- You read; we illustrate.
- Labs are important: exercise of programming principles as practical dirty details go there. Generally we will give you ho points for doing them.
- Homework is important, but really not graded: use it as you and *turn it in!* You get points for just putting some reasonable into it.
- Individual projects are *really* important! Expect to learn a lo are *not* team efforts (that's for later courses).

Course Organization II

- Use of tools *is* part of the course. Programming takes pl
  programming environment:
  - Handles editing, debugging, compilation, archiving version
  - Personally, I keep it simple: Emacs + gjdb + make + gi
    umented in one of the readers and on-line). But we'll
    IntelliJ in lab, and Eclipse is OK, too.
- Tests are challenging: better to stay on top than to cram.
- Tests, 40%; Projects, 50%; HW, 10%
- Stressed? Tell us!

Programming, not Java

- Here, we learn *programming,* not Java (or Unix, or Windows
- Programming principles span many languages
  - Look for connections.
  - Syntax (`x+y` *vs.* `(+ x y)`) is superficial.
  - Java, Python, and Scheme have a lot in common.
- Whether you use GUIs, text interfaces, or embedded syste portant ideas are the same.

For next time

- Please read Chapter 1 of *Head First Java*, plus §1.1–1.9 of the book *A Java Reference*, available on the class website.
- This is an overview of most of Java's features.
- We'll start looking at examples on Friday.
- Always remember the questions that come up when you rea thing we assign:
  - Who knows? We might have made a mistake.
  - Feel free to ask at the start of lectures, by email, or by

Acronyms of Wisdom

DBC

RTFM

A Quick Tour through the First Program

In Python, we would write

```
# Traditional first program
print("Hello, world")
```

But in Java,

```
/** Traditional first program.
 *  @author P. N. Hilfinger */
public class Hello {
    /** Print greeting. ARGS is ignored. */
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Commentary

```
/** Traditional first program.
 *  @author P. N. Hilfinger */
public class Hello {
    /** Print greeting.  ARGS is ignored.  */
    public static void main(String[] args) {
         System.out.println("Hello, world!");
    }
}
```

- Java comments can either start with '//' and go to the end
  line (like '#' in Python), or they can extend over any number
  bracketed by '/*' and '*/'.
- I don't use the '//' comments, except for things that are s
  to be replaced, and our style checks will flag them.
- The second, multiline kind of comment includes those that st
  '/**', which are called *documentation comments* or *doc comm*
- Documentation comments are just comments, having no eff
  various tools interpret them as providing documentation
  things that follow them. They're generally a good idea and o
  checks require them.

Classes

```
/** Traditional first program.
 *  @author P. N. Hilfinger */
public class Hello {
    /** Print greeting. ARGS is ignored. */
    public static void main(String[] args) {
         System.out.println("Hello, world!");
    }
}
```

- Every function and variable in Java is contained in some *clas*
- These are like Python's classes, but with (of course) numer ferences in detail.
- All classes, in turn, belong to some *package*. The `Hello` class to the *anonymous package*.
- We'll see named packages later,

Methods (Functions)

```
/** Traditional first program.
 *  @author P. N. Hilfinger */
public class Hello {
    /** Print greeting. ARGS is ignored. */
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

- Function headers in Java contain more information than t
  Python. They specify the *types* of values *returned* by the f
  and taken as *parameters* to the functions.
- The "type" `void` has no possible values; the *main* function h
  turns nothing. The type `String` is like Python's `str`. The tra
  means *array of*. Arrays are like Python lists, except that th
  is fixed once created.
- Hence, *main* takes a list of strings and returns nothing.
- Functions named "main" and defined like the example about
  cial: they are what get called when one runs a Java prog
  Python, the main function is essentially anonymous).

Selection

```
/** Traditional first program.
 *  @author P. N. Hilfinger */
public class Hello {
    /** Print greeting. ARGS is ignored. */
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

- As in Python, $\mathcal{E}.N$ means "the thing named N that is in or tha
  to the thing identified (or computed) by $\mathcal{E}$."
- Thus "`System.out`" means "the variable named '`out`' that is
  the class named '`System`'."
- Likewise, "`System.out.println`" means "the method named '`p`
  that applies to the object referenced by the value of variable

Access

```
/** Traditional first program.
 *  @author P. N. Hilfinger */
public class Hello {
    /** Print greeting. ARGS is ignored. */
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

- Every declared entity in Java has *access permissions* indicati
  pieces of code may mention it.
- In particular, *public* classes, methods, and variables may be r
  to anywhere else in the program.
- We sometimes refer to them as *exported* from their cl
  methods or varialbles) or package (for classes).

Access

```
/** Traditional first program.
 *  @author P. N. Hilfinger */
public class Hello {
    /** Print greeting. ARGS is ignored. */
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

- Static methods and variables are "one-of" things.
- A static method is just like an ordinary Python function (ou any class) or a function in a Python class that is annotated `@st`
- A static variable is like a Python variable defined outside class or a variable selected from a class, as opposed to from instance.
- Other variables are local variables (in functions) or instan ables (in classes), and these are as in Python.