cussion section using the SignUpGenius poll, available from the course website. If you can't find a slot, attend any section you can (although you have second priority for seating).

- Labs start today. In (or preferably before) lab this week, get a CS61B Unix account from `https://inst.eecs.berkeley.edu/webacct`.
- Because labs will be crowded, you might want to bring your laptop.
- If you plan to work from home, try logging in remotely to one of the instructional servers.
- We'll be using Piazza for notices, on-line discussions, questions.
- General information about the course is on the home page (grading, lateness, cheating policy, etc.).

mit any Concurrent Enrollment students. If you choose not to take this course please drop it as soon as possible for the benefit of others (the add/drop deadline is 18 September—6 September if you wish to avoid a fee).

the website).

- You could do without printed versions, but might want to print out selected portions for exams (since we don't allow computers in tests).
- Textbook (for first part of the course only) is *Head First Java*. It's kind of silly, but has the necessary material.

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

ming takes place in a *programming environment:*

  - Handles editing, debugging, compilation, archiving versions.
  - Personally, I keep it simple: Emacs + gjdb + make + git, (documented in one of the readers and on-line). But we'll look at 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!

Unix, or Windows, or...)

- 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 systems, important ideas are the same.

plus §1.1–1.9 of the on-line 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 read something we assign:
  - Who knows? We might have made a mistake.
  - Feel free to ask at the start of lectures, by email, or by Piazza.

DBC

RTFM

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!");
    }
}
```



```
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 of the line (like '#' in Python), or they can extend over any number of lines, bracketed by '/*' and '*/'.
- I don't use the '//' comments, except for things that are supposed to be replaced, and our style checks will flag them.
- The second, multiline kind of comment includes those that start with '/**', which are called *documentation comments* or *doc comments*.
- Documentation comments are just comments,

any a good idea and our style checks require them.

```
/** @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 *class*.
- These are like Python's classes, but with (of course) numerous differences in detail.
- All classes, in turn, belong to some *package*. The `Hello` class belongs to the *anonymous package*.
- We'll see named packages later,


```
/** @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 those in Python. They specify the *types* of values *returned* by the function and taken as *parameters* to the functions.
- The "type" `void` has no possible values; the *main* function here returns nothing. The type `String` is like Python's `str`. The trailing '`[]`' means *array of*. Arrays are like Python lists, except that their size is fixed once created.
- Hence, *main* takes a list of strings and re-

get called when one runs a Java program (in Python, the main function is essentially anonymous).


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


```
/** @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* indicating what pieces of code may mention it.
- In particular, *public* classes, methods, and variables may be referred to anywhere else in the program.
- We sometimes refer to them as *exported* from their class (for methods or varialbles) or package (for classes).


```
/** @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 (outside of any class) or a function in a Python class that is annotated `@staticmethod`.
- A static variable is like a Python variable defined outside of any class or a variable selected from a class, as opposed to from a class instance.
- Other variables are local variables (in functions) or instance variables (in classes), and these are as in Python.