Welcome to CS61B!

- Please see the rather extensive information on sections, Covid-19 policy, sections, labs, initial assignments, and the presemester survey on the Spring 2022 CS61B Piazza site.

- Labs start today. In (or preferably before) lab this week, get a CS61B Unix account from [https://inst.eecs.berkeley.edu/webacct](https://inst.eecs.berkeley.edu/webacct).

- Try logging in remotely to one of the instructional servers (...@X.berkeley.edu, where X is ashby.cs, derby.cs, cedar.cs, cory.eecs, and others).

- The course homepage ([https://inst.eecs.berkeley.edu/ cs61b/sp22](https://inst.eecs.berkeley.edu/ cs61b/sp22)) is our central distribution site for assignments, lecture slides, course policy, and much else.

- Lectures will be recorded and screencast. The recordings should become available in the bCourses Media Gallery sometime after the lecture.
Crowding, etc.

- 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 9 February—28 January if you wish to avoid a fee).

- As you know, Dwinelle will not hold us all, which is why there are both offline and online lectures. Lecture seating is on a first-come-first-seated basis. Definitely not ideal, but we hope that after the first few weeks, those of you who prefer in-person lectures will be able to have it.

- Lectures, etc., will be entirely online for at least the first two weeks, due to the current outbreak.
Texts

- There are two readers currently on-line (see the website).
- Textbook (for first part of the course only) is *Head First Java*. It’s kind of silly, but has the necessary material.
Course Organization I

- You read; we illustrate.

- 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 it's reasonably easy to get full credit: use it as you see fit and *turn it in!* You should 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).
Course Organization II

- Use of tools is part of the course. Programming 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.

- Tests are challenging: better to stay on top than to cram.

- Tests, 40%; Projects, 50%; HW, 10%

- Stressed? Tell us!
Pandemic Considerations

• It’s everyone’s responsibility to look out for each other.

• This semester, in particular, this means adhering to certain inconvenient practices mandated by the University.

• These include wearing masks indoors, as well as staying home when sick.

• Please observe the mask mandate; if anyone refuses, I can and will be forced to simply end the day’s lecture, and you’ll all have to rely on the on-line slides for the material.
Academic Dishonesty

- Sadly, the incidence of academic dishonesty seems to have increased over the years.
- To an extent, this is our fault: the minimum GPA threshold policy for L&S majors puts people under a lot of stress,
- Nevertheless, we can’t afford to tolerate cheating. The Course Info tab on the course homepage contains our policy on cheating and the penalties we impose; please read them.
- By keeping up with the course and starting assignments early, you can reduce any perceived need to cheat.
- Also, this course is not curved, so you are not disadvantaged by other people’s dishonesty.
Programming, not Java

- Here, we learn *programming*, not Java (or 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.
For next time

• Please read Chapter 1 of *Head First Java*, 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.
Acronyms of Wisdom

DBC

RTFM
A Quick Tour through the First Program

In Python, we would write

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

But in Java,

```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!");
   }
 }
```
/** 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 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, having no effect, but various tools interpret them as providing documentation for the things that follow them. They’re generally a good idea and our style 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 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,
**Methods (Functions)**

```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!");
    }
}
```

- 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 returns nothing.

- Functions named “main” and defined like the example above are special: they are what get called when one runs a Java program (in 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, $E.N$ means “the thing named $N$ that is in or that applies to the thing identified (or computed) by $E$.”

• 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’.”
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 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 variables) or package (for 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!");
   }
}

- 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.