One-on-One Sessions (optional)

Next 1.5 weeks, we are holding 1/2-hour one-on-one sessions on
a computer with each student in CS1110.

**Purpose:** Help you develop a class as preparation for A1, give
you a chance to ask questions. Not counted in final course grade.

**Sign up on the course CMS** (http://www.cs.cornell.edu/courses/cs1110/2010fa/staff.html):
Click on assignment One-on-one, find the schedule of times/instructors.
Choose one.

**Bring to the 1-on-1:** the book; laptop w. DrJava if you have one.

Students with little or no programming experience report that
these sessions are extremely helpful!

**Office hours:** http://www.cs.cornell.edu/education/officehours.html
Already started. Prof Gries & Prof Lee (Tu/Th 10:10-10:55 Hollister 202); consultants
TAs: will begin the week of Sept. 20; check the URL then

Declarations of fields

### Field: a variable that is in each folder of a class.

- id
- lname
- ssn
- boss

#### Example of fields:

```java
public class Worker {

    private String lname;
    private int ssn;
    private Worker boss;

    // Last name ("" if none; never null)
    // Social security #: in 0..999999999
    // Immediate boss (null if none)

    public Worker(String lname, int ssn, Worker boss) {
        this.lname = lname;
        this.ssn = ssn;
        this.boss = boss;
    }

    public Worker(String lname) {
        this.lname = lname;
    }

    // Return the worker's last name
    public String getName() {
        return lname;
    }

    // Set worker's last name to n (can't be null, can be "")
    public void setName(String n) {
        if (n == null) {
            throw new IllegalArgumentException();
        }
        this.lname = n;
    }

    // Get/set worker's immediate boss
    public Worker getBoss() {
        return boss;
    }

    public void setBoss(Worker b) {
        this.boss = b;
    }

    // Get/set worker's social security number
    public int getSsn() {
        return ssn;
    }

    public void setSsn(int n) {
        if (n < 0 || n > 999999999) {
            throw new IllegalArgumentException();
        }
        this.ssn = n;
    }

    // Are we the same worker?
    public boolean equals(Worker w) {
        return this.lname.equals(w.lname) &&
               this.ssn == w.ssn;
    }

    public String toString() {
        return this.lname;
    }
}
```

Quiz 1 on assignment

Suppose we have a recipe language, with statements like this:

```java
bowl <-- 1 cup flour   +  1 tsp salt   +   baking powder;
```

**Meaning:** Put the flour and salt and baking powder in the bowl.

### Quiz 2 on Tuesday 14 Sept

**Purpose of a constructor (slide 6); Evaluating a new expression (slide 8)**

Assignment A1 out today, due Sat., 18 Sept. on the CMS.
Submit A1 earlier if you can so that we can start the iterative feedback
process going.

Labs and one-on-ones (schedule yours on CMS) will help you with it.

**Collaboration rules for assignment A1**

- **Work alone or with one partner**—partners “group themselves” on the
  CMS well before submission; only one person submits the files.

  Partners must do the work together, sit next to each other, with each
taking turns “driving” (handling the mouse and keyboard). It is against
the rules for one partner to develop code and later show it to the other.

- **Never** look at someone else’s code or show yours to someone else.

Never be in possession of someone else’s code (except your partner).

Field: a variable that is in each folder of a class.

- id
- lname
- ssn
- boss

#### Example of fields:

```java
public class Worker {

    private String lname;
    private int ssn;
    private Worker boss;

    // Last name ("" if none; never null)
    // Social security #: in 0..999999999
    // Immediate boss (null if none)

    public Worker(String lname, int ssn, Worker boss) {
        this.lname = lname;
        this.ssn = ssn;
        this.boss = boss;
    }

    public Worker(String lname) {
        this.lname = lname;
    }

    // Return the worker's last name
    public String getName() {
        return lname;
    }

    // Set worker's last name to n (can't be null, can be "")
    public void setName(String n) {
        if (n == null) {
            throw new IllegalArgumentException();
        }
        this.lname = n;
    }

    // Get/set worker's immediate boss
    public Worker getBoss() {
        return boss;
    }

    public void setBoss(Worker b) {
        this.boss = b;
    }

    // Get/set worker's social security number
    public int getSsn() {
        return ssn;
    }

    public void setSsn(int n) {
        if (n < 0 || n > 999999999) {
            throw new IllegalArgumentException();
        }
        this.ssn = n;
    }

    // Are we the same worker?
    public boolean equals(Worker w) {
        return this.lname.equals(w.lname) &&
               this.ssn == w.ssn;
    }

    public String toString() {
        return this.lname;
    }
}
```
Initialize fields when a folder is first created

We would like to be able to use something like

```
new Worker("Obama", 1, null)
```

to create a new Worker, set the last name to “Obama”, the SSN to 000000001, and the boss to null.

For this, we use a new kind of method, the constructor.

**Purpose of a constructor:** to initialize (some) fields of a newly created object.

This initialization should make the class invariant true.

**In the class definition of Worker:**

```
/** Constructor: an instance with last name n (can’t be null, can be ""), SSN s (an int in 0..999999999), and boss b (null if none) */
public Worker(String n, int s, Worker b)
    lname= n;
    ssn= s;
    boss= b;
}
```

```
// Test the new definition using JUnit

Bug: Error in a program.
Testing: Process of analyzing, running program, looking for bugs.
Test case: A set of input values, together with the expected output.
Debugging: Process of finding a bug and removing it.

Get in the habit of writing test cases for a method from the method’s specification — even before writing the method’s body.

A feature called JUnit in DrJava helps us develop test cases and use them. You have to use this feature in assignment A1.

Here are two test cases

1. `w1 = new Worker("Obama", 1, null);`
   Name should be: “Obama”; SSN: 1; boss: null.

2. `w2 = new Worker("Biden", 2, w1);`
   Name should be: “Biden”; SSN: 2; boss: w1.

Need a way to run these test cases, to see whether the fields are set correctly. We could use the interactions pane, but then repeating the test is time-consuming.

To create a testing framework: select menu File item new JUnit test case…. At prompt, put in class name WorkerTester. This creates a new class with that name. Save it in same directory as class Worker.

The class imports junit.framework.TestCase, which provides some methods for testing.

Test case template created by DrJava

```
/** A JUnit test case class. */
public class WorkerTester extends TestCase {
    /** A test method. */
    public void testX() {
        // Replace "X" with a name describing the test. Write as many "testSomething" methods in this class as you wish, and each one will be called when testing. */
        assertEquals(x, y); /* which tests whether expected value x equals computed value y. */
    }
}
```
A testMethod to test constructor (and getter methods)

/** Test first constructor (and getter methods getName,
getSSN4, and getBoss) */

public void testConstructor() {
  Worker w1=new Worker("Obama", 123456789, null);
  assertEquals("Obama", w1.getName());
  assertEquals(6789, w1.getSSN4());
  assertEquals(null, w1.getBoss());
  Worker w2=new Worker("Biden", 2, w1);
  assertEquals("Biden", w2.getName());
  assertEquals(2, w2.getSSN4());
  assertEquals(w1, w2.getBoss());
}

Every time you click button Test in DrJava, this method (and all other testX methods) will be called.

A few other methods that can be used are listed on page 488.

assertEquals(x,y):
  test whether x (expected) equals y (computed);
  print an error message and stop the method if they are not equal.