1. Programming start. 2. Three (or 4) -person warmup  3. Three (or 4)-person programming game  
If you didn't miss Lab2, you can skip indicated steps only.

HU-25 Lab login: NetId and password (same used for MyUAAlbany and Blackboard) are needed.
1. Click the boxy gray-black left-most SSH to ITS icon at the bottom of the screen.
2. Type your NetId at the Username: prompt (lower case letters and some digits); press Enter.
3. Type your password at the Password: prompt--just keep typing it carefully even though you'll see nothing being echoed back. Unix turns off echoing during password typing so other's can't steal it. (Avoid or undo setting Caps Lock; Upper vs lower case really matters!)
4. The directory/folder that the shell starts you in is the very same as your UA S: drive.

In many labs you will get training in IT with a command line shell a little at a time. Using a command line shell is a valuable IT skill for programmers because it is more flexible, easier to document, programmable and actually faster for repeated operations than point and clicking. Let's start:

Going to your CSI201 directory/folder, creating it if you don't have it from Proj1.

Command: cd CSI201
(After C, S and letter I, there is two, zero and one. Press Enter after every command.)
This asks the shell to try to change directory to CSI201
Those of you who just see another prompt (no error messages and no confirmation messages) already have the CSI201 directory/folder from doing Proj1. Those people will see:

unix2% cd CSI201
If you see this, right now jump to the Seeing ... step below. (Lab2 doers: jump right now.)

What if you didn't already have the directory/folder named CSI201? Those who don't have their CSI201 directory/folder will see:

unix2% cd CSI201
-bash: cd: CSI201: No such file or directory

IF you got the No such ... error message, please command: mkdir CSI201
That makes directory CSI201. (No confirmation is printed!) Now give again the command: cd CSI201

Seeing if you have the book Classes in your CSI201 directory/folder from Proj1, getting them now if not.

Command: ls -l
(Lower case letter ell, a space, a hyphen another lower case letter ell) This stands for list the contents of the shell's current working directory. The letter ell after the hyphen specifies the option to do a long listing which shows details in addition to names.
If you did Proj1 on your S: drive, you should see something like what's on the next page. The sample
came from what I demonstrated in the first few lectures using my S: drive. Your's will of course have your Unix user name (your NetId) instead of sdc, and your group name won't be faculty. The file sizes, dates and times will differ too. I added lines that might have resulted from Lab2 and Proj3

```
unix2% ls -l
total 9526
drwxr-xr-x 3 sdc faculty 512 Jan 30 17:04 bookClasses-7-22-09
-rwxr--r-- 1 sdc faculty 4853453 Jan 30 17:01 bookClasses-7-22-09.zip
-rwxr--r-- 1 sdc faculty 648 Jan 28 17:29 MyFirstProgram.class
-rwxr--r-- 1 sdc faculty 375 Jan 28 17:29 MyFirstProgram.java
-rwxr--r-- 1 sdc faculty 130 Jan 23 17:38 MyFirstProgram.java~
-rwxr--r-- 1 sdc faculty 883 Jan 30 17:27 TurtleDemo.class
-rwxr--r-- 1 sdc faculty 563 Jan 30 17:27 TurtleDemo.java
-rwxr--r-- 1 sdc faculty 111 Jan 30 16:53 TurtleDemo.java~
-rwxr-xr-x 1 sdc faculty 512 Feb 3 10:15 Lab2
-rwxr-xr-x 1 sdc faculty 512 Feb 4 10:21 Proj3
unix2%
```

On the other hand, if you just made your CSI201 directory/folder, it is empty so you'll see:

```
unix2% ls -l
total 0
unix2%
```

If you see bookClasses, go now to the next page on Using DrJava in HU-25. People who have done Lab2 successfully should just skip to Using DrJava.

Otherwise, type the following command exactly then press <Enter>. Exactly means to reproduce exactly which letters are capitalized (only the C of Classes in this example) and to note there are are spaces ONLY between the unzip token and the rest of the line. The token following unzip is an example of a "full pathname" which locates a file. It consists of simple names separated by the / separator (or the \ separator in Windows. In our lab, use / only because it is UNIX!).

```
unzip /usr/local/depts/cs/geintro/bookClasses.zip
```

If all goes well, you should see a stream of names as the archive gets unzipped and copied. If not, many of you will see one of these two error messages:

```
unzip: cannot find or open
If so, review what you did type and try it again, with correction. You probably mis-copied the full path name above.
---OR---
-bash: something: command not found
You mistyped or miss-capitalized unzip Please try again; get help to get through this step fast!
When all goes well, you will see:
Archive: /usr/local/depts/cs/geintro/bookClasses.zip
  creating: bookClasses/
  inflating: bookClasses/rightArrow.gif
  inflating: bookClasses/SimplePictureProperties.txt
and many more lines...just watch until it's done.
```
Using DrJava in HU-25

Type in EXACTLY what you see below, which is a line of lower-case letters and slash characters with no spaces in-between. It is our UNIX shell command to start DrJava in the lab, so you must press ENTER at the end of the line: (TIP: Type /usr/loc TAB and see what happens! Pros work fast!)

/usr/local/depts/cs/geintro/drjava

After a few moments the DrJava splash screen will appear, followed by the DrJava window itself. If it offers an upgrade, click "Close".

If you ALREADY set preferences in the Lab, just skip to Start NOW!

Like you have to do in the Library computers, you will have to set the "Extra Classpath" preference/resource value to locate your copy of the BookClasses. (In the HU-25 DrJava, preference setting as kept!) Starting from the menu bar, select Edit/Preferences and make sure you're in the resources page. Click Add right and underneath Extra Classpath. Locate and select the bookClasses dir/folder (NOT bookClasses-7-22-10!) Then, as usual click Apply. Please set 2 more preferences, to make your's and everyone else's lab faster:

Preference change 1:
Edit->preferences -> display options -> toolbar buttons: text only
Preference change 2: (HU-25 LAB only)
Edit->preferences->display options->change the "look and feel" option to com.sun.java.swing.plaf.motif.MotifLookAndFeel

As usual, click Apply. Then, click OK. Finally exit (with File/Close or the X button) Dr. Java and restart it by repeating the long typed in command above. (Or, use the shell's history feature by pressing Control-p until you see the command, and then press enter!)

Start Programming for Lab4--you will work today with TWO .java files

DO EACH PROJECT beyond 1 and each LAB in its OWN directory/folder! This will start to count soon! (Your shell should be in the current working directory CSI201; do something to fix that or get help if not)
Back in your original shell window command: mkdir Lab4 (mkdir stands for make directory. As usual, Unix commands print no confirmation when they are successful, but always print error messages when they fail.)

Type in the following short program: (Use nice indentation: That will count too!) Use DrJava's File/Save As... feature to get it saved inside that Lab4 directory/folder, making sure the filename is DoWhatBossWantsApp.java (Dr Java might hide the .java extension/suffix).

```java
public class DoWhatBossWantsApp
{
    public static void main(String[]a)
    {
        World wref = new World();
        BossableTurtle tref = new BossableTurtle( wref );
        tref.helloAndLine( 100 ); //Purpose: Make the computer do what's pre-programmed //in the helloAndLine method.
    }
}
```

When you try to compile it, you WILL get errors; that's OK. You must now type the second short .java given below. Make sure the filename is BossableTurtle.java

Page 3
public class BossableTurtle extends Turtle
{
    public BossableTurtle(World wref)
    { super(wref); }
    public void helloAndLine( int sizeParam )
    {
        System.out.println("Hello."); //Purpose: Demonstrate printing.
        this.forward( sizeParam ); //Purpose: Demonstrate line drawing done when the
               //forward method is called on this Turtle.
        return; //Purpose: Stop taking instructions from here and return to where
               //the method was called from.
    }
}

You are making a complete Java application composed of TWO .java source files. Keep clicking
Compile and fix typing mistakes until there are no more error messages. Then click Run and notice
that it really does print Hello. and draw a line.

Notice that within the helloAndLine method, the Turtle is referred to by this (a Java
keyword) and NOT the variable we used in the main method. Now, go into the BossableTurtle
file (NOT DoWhatBossWantsApp) and pre-program it to do things just a little more creative and
interesting than what we gave you. Test your program to print a message, and make that
BossableTurtle intersperse going forward and turning a few times. If you like, make an int
variable or two, program some little things to do with its value, and use its value to control the amount
of BossableTurtle turning or going forward. ASK THE TAs to write sample statements on the
whiteboard. If you can now, and certainly during the game, you should ALSO try coding within
DoWhatBossWantsApp additional method calls to the helloAndLine method, and eventually,
other methods you have defined in BossableTurtle.

Customize your BossableTurtle as quickly as you can, because in the game the people who have
gone furthest in getting their BossableTurtles to show off can take the starting team position of
Warmup Programmer. (You can very much abbreviate your //Purpose comments to play more
quickly.)
Team Warmup

Please actually start the warmup and the game even though not everybody will have their DrJava set up! During the game, the other team-members (with TA help if needed) should help their team-mates finish the setup quickly.

Reform if possible yourselves into 3-person teams, make a 4-person team or two if that's not possible.

In each team, the person who got furthest along in finishing her or his DoWhatBossWantsApp should play Warmup Programmer, and finishes if necessary her or his work.
Name___________________________

Warmup Code Reviewer or Predictor reads but does not see run the program written by the Warmup Programmer, and tries to predict what the computer will do when the program eventually runs. Name of Warmup Code Reviewer and Predictor_____________________________

Warmup Tester(s) actually run the program and the team discusses whether and how what the Programmer intended, what the Predictor predicted, and what the Tester(s) observed were different.
Name_________________________________ (Name___________________________________)

The Programming Game

There are 3 roles: The players shift or take turns, so each player plays each role.
Boss or Designer: Decides what the computer should do and explains it to the Programmer. The Boss also will eventually actually run and test the program, and records the results on the report sheet. The Boss should choose things not too difficult for the team but not too boring.
Programmer: Programs what the Boss told him or her; compile to find and fix syntax errors BUT DOES NOT RUN THE PROGRAM!
Code Reviewer or Predictor: Looks at the program code as and after the Programmer types it and tries to predict what the computer will do.
The Warmup Programmer should be the first round Boss or Designer. Figure who plays what role in each round so you won't be confused as you play! (p, q, r and sometimes s symbolize a person in the roster.)
1st Round Boss_________________________________________ p p
1st Round Programmer___________________________________ q q
1st Round Predictor______________________________________ r r

2nd Round Boss_________________________________________ q s
2nd Round Programmer___________________________________ r p
2nd Round Predictor______________________________________ p q

3rd Round Boss__________________________________________ r r
3rd Round Programmer____________________________________ p s
3rd Round Predictor______________________________________ q p

(have 4 rounds only if 4 people are on the team.)
4th Round Boss_________________________________________ q
4th Round Programmer____________________________________ r
4th Round Predictor______________________________________ s
4 person team report form (Try to avoid 4 person teams!)
Give ONE single form for the whole team. TA will base grade on completions, NOT number of YESs vs NOs!

I (name) __________________________ am the first team-mate to play the role of Boss (or Designer)

Name of the round one Programmer ________________________________
Name of the round one Code Reviewer (and Predictor) ____________________________

The Boss should now tell the Programmer what he or she wants the computer to do.
The Programmer should program what the Boss asked for.
The Predictor should look at the code and predict what it will make the computer do WITHOUT RUNNING IT TO SEE!

Do the Programmer and the Predictor agree on their predictions of what the computer will do? ☐ YES ☐ NO
Next the Boss (or Designer) should TEST the program by running it, and record from the team's discussion:

Did the program do what the Boss wanted? ☐ YES ☐ NO
Did the program do what the Predictor predicted? ☐ YES ☐ NO
Did the program do what the Programmer expected or intended? ☐ YES ☐ NO

I (name) ____________________________ am the second team-mate to play the role of Boss (or Designer)

Name of the round two Programmer ________________________________
Name of the round two Code Reviewer (and Predictor) ____________________________

The Boss should now tell the Programmer what he or she wants the computer to do.
The Programmer should program what the Boss asked for.
The Predictor should look at the code and predict what it will make the computer do WITHOUT RUNNING IT TO SEE!

Do the Programmer and the Predictor agree on their predictions of what the computer will do? ☐ YES ☐ NO
Next the Boss (or Designer) should TEST the program by running it, and record from the team's discussion:

Did the program do what the Boss wanted? ☐ YES ☐ NO
Did the program do what the Predictor predicted? ☐ YES ☐ NO
Did the program do what the Programmer expected or intended? ☐ YES ☐ NO

I (name) ____________________________ am the third team-mate to play the role of Boss (or Designer)

Name of the round two Programmer ________________________________
Name of the round two Code Reviewer (and Predictor) ____________________________

The Boss should now tell the Programmer what he or she wants the computer to do.
The Programmer should program what the Boss asked for.
The Predictor should look at the code and predict what it will make the computer do WITHOUT RUNNING IT TO SEE!

Do the Programmer and the Predictor agree on their predictions of what the computer will do? ☐ YES ☐ NO
Next the Boss (or Designer) should TEST the program by running it, and record from the team's discussion:

Did the program do what the Boss wanted? ☐ YES ☐ NO
Did the program do what the Predictor predicted? ☐ YES ☐ NO
Did the program do what the Programmer expected or intended? ☐ YES ☐ NO

I (name) ____________________________ am the fourth team-mate to play the role of Boss (or Designer)

Name of the round two Programmer ________________________________
Name of the round two Code Reviewer (and Predictor) ____________________________

The Boss should now tell the Programmer what he or she wants the computer to do.
The Programmer should program what the Boss asked for.
The Predictor should look at the code and predict what it will make the computer do WITHOUT RUNNING IT TO SEE!

Do the Programmer and the Predictor agree on their predictions of what the computer will do? ☐ YES ☐ NO
Next the Boss (or Designer) should TEST the program by running it, and record from the team's discussion:

Did the program do what the Boss wanted? ☐ YES ☐ NO
Did the program do what the Predictor predicted? ☐ YES ☐ NO
Did the program do what the Programmer expected or intended? ☐ YES ☐ NO