Index of /~cs61b/labs/lab1

Name	Last modified	Size	Description
Parent Directory	30-Aug-98 15:13	-	
Names.java	20-Aug-98 14:30	2k	
roster.txt	20-Aug-98 14:30	3k	

```
                          CS61B Lab 1
                   August 24 - August 26, 1998

The goal of this first lab is to make sure you can use two crucial tools
in this course: the emacs editor and the java system.

There are a couple of things you will need to checkoff to get credit for
this lab.  You can do that all at once when you are finished, but don't
wait until the last minute -- another lab will start as soon as yours
ends, so ask to be checked off when there is still plenty of time.

PART I: Getting started (1 point)
_________________________________

0. Pick up an account form from your Lab TA or Lab Assistant (LA).
   They will have an account form only for the people who are officially
   enrolled in the lab.

1. Login to a machine using the temporary password on your
   account form.  Answer the questions.  Use whatever the
   registrar uses for your given name and surname, and the name
   you prefer to be called for your nickname.  The first one is
   used for grading and the second will show up when other people
   "finger" your account.

2. Warning: do not, at this time or any point in the future,
   delete the .login, .cshrc, and .emacs files in your directory.
   If you would like to customize your account, do so by adding
   new commands to the end of these files, not replacing the files.

3. The rest of this lab is a TEAM project.  Introduce yourself to
   the person to your left and right (there is one point associated
   with knowing the names of two other people in the class).  Find
   the nearest unpartnered person.  If you don't have a partner, go
   ahead and begin, but be sure to help your partner catch up when
   he or she arrives.

PART II: Compile-time errors (1 point)
______________________________________

After logging in, you and your partner may find it easier to work
on one machine for a while.

1. Make a lab1 subdirectory:
        mkdir lab1
```

2. Connect to your lab1 directory:

```
cd lab1
```

3. Copy the lab files by typing:

```
cp  $master/labs/lab1/* .
```

(Note: the $master variable refers to the 61b home directory, which is /home/ff/cs61b. If the above command doesn't work, it probably means that something is messed up in your login scripts, so talk to your TA or LA.)

4. Load the program file (Names.java) into emacs using C-x C-f. (C-x is read control x, and is typed by holding down the control ctrl key while typing "x".)

Names.java is a simple program for performing various string operations on a name. It almost works, but you need to make some changes so that it compiles and runs correctly.

5. Compile the program within emacs using:

```
C-x C-e
```

This will build a command line to run the java compiler:

```
javac -g
```

javac is the name of the compiler and -g is a flag to tell the compiler that you may want to run the debugger so it should produce debugging information. You need to fill in the command line by filling in the name of the file to be compiled:

```
javac -g Names.java
```

You may also type the above command in a shell window, assuming you're in your copy of the lab1 directory.

6. You will find that javac cannot compile the program. The code contains two syntax errors which you are quite likely to make when you will write programs. Using the control sequence

```
C-x `
```

to jump directly to errors found by javac. (Be sure to use the single forward quote, not the backwards quote, which is an apostrophe.)

PART III: run-time error (1 point)

1. Once javac is able to compile the code, it will create a file called Names.class in the same direcotyr. You can run this program using the java interpreter. In a shell (another non-emacs window or a buffer of emacs called *shell*) type the following:

```
java Names
```

In this case, the .class file extension on Names.class is assumed. If you accidentally type "java Names.class" it will look for a file called "Names.class.class".

Can you tell there is something wrong? This type of error, which occurs at run-time, tends to be significantly more difficult to correct than compile-time errors. (It's still somewhat easier than discovering an error in which the program appears to finish without problems, but is doing some computation incorrectly.) The error message may be hard to read at first, but it will allow you to answer certain questions: What is the method that generated an error? What is the line number within the Names.java file?

The error is in one of the methods in the String class, which is a standard Java library. There is a bit of documentation on the Java library in your textbooks, but the best source is the online documentation. To find this, start netscape program by typing:

```
netscape &
```

Find the "open location" command and find the class home page:

```
http://www-inst.eecs.berkeley.edu/~cs61b
```

From there, you can find a pointer to the "Java Package Documentation." There are two libraries that you will be using early in the semester, the java.lang package and the java.io package. Documentation on String and other standard data types is found in java.lang, so go there and find the String class. There is a lot of information, not all of which will make sense right now, but you should be able to find a description of the problematic String method.

When you think you have found the error: correct it, save the file, recompile and execute it to see if the problem is solved.

Aside: You may think that the file produced by javac is named Names.class because the input file is named Names.java. However, it is based on the class name in the file. To experiment with this, change the line "class Names {" to "class NAMES {" and recompile using javac. What new file was created?

PART IV: emacs help (1 point)

1. Load the file 'roster.txt' in emacs. You will see an unsorted list of names with each line in the form <last name, first names>.

You should use the help facilities of emacs (apropos and info) to find out how to sort all these names by last name. In other words, you want to sort the lines of the file. If you are unfamiliar with the help facilities type C-h ? and emacs will guide you through them.

Save the result using C-x C-s

Check-off

When you're all done, show your TA or LA that the Names program works and that you have a sorted version of the roster.txt file. They may also ask some questions, such as the names of your neighbors in the room and what you discovered about bugs in the original version of the Names program.

If you felt you had trouble with either Unix commands or emacs, you are strongly encouraged to attend one of the help sessions on these topics held by the CSUA. See the class home page for a pointer to the schedule.