Hello, people in my (Professor Parson's) Java classes.
You owe me an email before class on February 7, 2013 showing a screen dump of gmake test clean below working from your account. It is worth 10% of project 1.

The Java tools path we want to use is **/usr/jdk/jdk1.6.0_02/bin**.

I have walked through the setup steps for both C Shell (csh) and bash to make sure you have correct instructions for either. Most people will use the instructions for C shell, so I will give them first. A "shell," by the way, is just the command interpreter that runs when you log into your UNIX account. It accepts input from your terminal and interprets your commands.

WARNING: DON'T copy-then-paste lines from the on-line copy of this file into bill's command window. Many of the non-alphabetic characters paste as dots or dashes into bill.

1. HOW TO DETERMINE WHICH SHELL YOU ARE RUNNING

Immediately after logging in, type "ps" and you will see something like THIS for C SHELL:

```
v245-2% ps
   PID TTY         TIME CMD
 24369 pts/19      0:00 csh
 24374 pts/19      0:00 ps
```

or something like THIS for BASH

```
-bash-3.00$ ps
   PID TTY         TIME CMD
 24369 pts/19      0:00 bash
 24373 pts/19      0:00 ps
```

INSTRUCTIONS FOR PEOPLE RUNNING C SHELL:

Immediately after logging in, TYPE THE FOLLOWING LINES, AND YOU SHOULD EVENTUALLY SEE SIMILAR RESULTS. If you do not, follow the instructions here for correcting the missing pieces.

```
v245-2% echo $JAVA_HOME
/usr/jdk/jdk1.6.0_02/bin

v245-2% javac -version
javac 1.6.0_02

v245-2% java -version
java version "1.6.0_02"
Java(TM) SE Runtime Environment (build 1.6.0_02-b05)
Java HotSpot(TM) Server VM (build 1.6.0_02-b05, mixed mode)

v245-2% echo $CLASSPATH
/YOURHOMEDIRECTORY/JavaLang:
```

You almost certainly will **NOT** see a correct CLASSPATH environment variable, because that is specific to this course. The instructions that follow will help you set these.

If you execute "ls -al" in your login directory, you may see a ".login" file. If you do not see it, make sure that you are in your login directory, and try again. If it is not there, create it with a text editor. If it already exists, just add these lines **AT THE TOP OF THE FILE**.

2. If "**echo $JAVA_HOME**" does not display /usr/jdk/jdk1.6.0_02/bin, add the following line to the top of your .login file:

```
setenv JAVA_HOME /usr/jdk/jdk1.6.0_02/bin
```

3. If "**javac -version**" or "**java -version**" does not give the correct result, add the following line to the top of your .login file:

```
setenv PATH "/usr/jdk/jdk1.6.0_02/bin:${PATH}"
```

NOTE: Microsoft Word makes it appear that there is a space after the "{" and "}" characters in the above command line when printing in certain fonts. There is no space before or after "{" or "}" in any of these setenv commands.

4. If CLASSPATH does not include JavaLang within your login directory (LIKELY!), add the following line to the top of your .login file:

```
setenv CLASSPATH "${HOME}/JavaLang"
```

5. Save your updated .login file, log out of your UNIX account, log back in, and run the above tests again. You should now be able to run "javac –version" and "java –version" correctly, and JAVA_HOME and CLASSPATH should be set correctly.

6. Run the following commands to create your JavaLang directory, and then copy a project from my entire directory to your JavaLang directory. Make sure to capitalize the "J" and "L" in JavaLang.

```
mkdir  ~/JavaLang                  # This creates your directory.
chmod 700 ~/JavaLang               # This lets only you access your JavaLang/.
cd  ~/JavaLang                     # This moves you into your JavaLang dir.
cp ~parson/JavaLang/Crossword3.solution.zip Crossword3.solution.zip # Copies.
unzip Crossword3.solution.zip      # Unzip my solution to last year's project.
cd  ./Crossword3                   # Go into that directory
gmake  test  clean                 # Test to see that your Java is set up.
cd  ..                             # Go back up to JavaLang
rm –rf Crossword3.solution.zip Crossword3      # Clean up recovers space.
```

My example output for gmake test clean:

```
-bash-3.00$ pwd
/export/home/faculty/parson/JavaLang
-bash-3.00$ unzip Crossword3.solution.zip
Archive:  Crossword3.solution.zip
   creating: Crossword3/
  inflating: Crossword3/english.0.txt
  inflating: Crossword3/FindWord.ref
  inflating: Crossword3/ExampleWord.java
  inflating: Crossword3/ExampleWord.ref
  inflating: Crossword3/makelib
  inflating: Crossword3/makefile
  inflating: Crossword3/HideWord.java
  inflating: Crossword3/FindWord.java
  inflating: Crossword3/HideWord.ref
  inflating: Crossword3/HideWordERR.ref
   creating: Crossword3/listings/
  inflating: Crossword3/HelpWordPuzzle.java
  inflating: Crossword3/WordPuzzle.java
  inflating: Crossword3/TestWord.java
-bash-3.00$ cd ./Crossword3
-bash-3.00$ gmake test clean
/bin/bash -c "javac -g HelpWordPuzzle.java"
/bin/bash -c "javac -g TestWord.java"
/bin/bash -c "javac -g ExampleWord.java"
/bin/bash -c "javac -g FindWord.java"
/bin/bash -c "javac -g HideWord.java"
Note: /export/home/faculty/parson/JavaLang/Crossword3/MyArrayList.java uses
unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
java Crossword3.MyArrayList > MyArrayList.out
diff MyArrayList.out MyArrayList.ref > MyArrayList.dif
java Crossword3.TestWord ExampleWord 10 100 1 < english.0.txt > ExampleWord.out
diff ExampleWord.out ExampleWord.ref > ExampleWord.dif
java Crossword3.TestWord FindWord 10 100 1 < english.0.txt > FindWord.out
diff FindWord.out FindWord.ref > FindWord.dif
java Crossword3.TestWord HideWord 15 100 1 < english.0.txt > HideWord.out
2>HideWordERR.out
diff HideWordERR.out HideWordERR.ref > HideWordERR.dif
diff HideWord.out HideWord.ref > HideWord.dif
grep PLACING HideWordERR.out > solution.out
/bin/rm -f *.o *.class .jar core *.exe *.obj *.pyc
/bin/rm -f *.class *.out *.dif ExampleWord FindWord
-bash-3.00$ cd ..
-bash-3.00$ rm -rf Crossword3.solution.zip  Crossword3
```

PLEASE SEND ME AN EMAIL BEFORE FEBRUARY 7 SHOWING ME A SCREEN DUMP OF gmake test clean THAT IS WORKING PROPERLY. IT IS WORTH 10% OF PROJECT 1.

Also once this is set up, please run the following program interactively by typing this from within the countargs directory. We will go over this in class.

```
cd  ~/JavaLang
cp  -pr  ~parson/JavaLang/countargs  countargs
cd  ./countargs
gmake
java countargs.CountArgString SEARCHSTRING
```

where SEARCHSTRING is a string for which you want to search. At this point you can enter lines of text into the keyboard to be searched; use control-D at the start of a line to terminate input. For example:

```
v245-2% java countargs.CountArgString cat
category catalog
catcatdogcat
cart
^D
String count = 5
Line count = 3
Integer ratio of count to lines = 1
```

```
Double ratio of count to lines = 1.6666666666666667
v245-2%
```

Also, please go over all of file CountArgString.java, including method reportCount(), to make sure you understand how this program works.

9. Someone without prior experience with makefiles was concerned about not knowing GNU make. My goal is to supply you with a makefile template that you can modify for each assignment. You do not need to become an expert in gmake. Basically, each paragraph in a makefile has this form:

```
TARGET:          DEPENDENCIES
                 ACTIONS
```

Where "TARGET" can be one of two things:

a) something you want to accomplish such as "build" or "test"
b) the name of a file that you want to create

"DEPENDENCIES" can likewise be either:

a) a source file upon which a target object or executable file depends
b) a TARGET in another part of the makefile, on which this target depends

"ACTIONS" are always preceded by one or more TAB characters. ACTIONS are just like lines that you would type into the command line. They run the "javac" compiler or "java" JVM-runtime or "diff" or other utilities that are accessible from the UNIX command line.

My advice is to play around with "gmake test" "gmake clean" and "gmake build" in that lecture1/ directory, while also looking at the makefile, to get a feel for how the makefile works.

10. An emacs user reported that the shell could not find emacs. If you want to run emacs and it does not work, add the following to .login

```
setenv PATH "/opt/csw/bin:${PATH}"
```

then log out and log in.
<u>For BASH USERS</u>!!!:

Follow above steps, but use file .bash_profile instead of .login, and instead of using setenv, put the following assignment commands at the bottom of .bash_profile. Steps 2 through 5, and 10, are as follows for bash; the other steps are the same as for csh.

2. If **"echo $JAVA_HOME"** does not display /usr/jdk/jdk1.6.0_02/bin, add the following line to your .bash_profile:

JAVA_HOME=/usr/jdk/jdk1.6.0_02/bin ; export JAVA_HOME

3. If **"javac -version"** or **"java -version"** does not give the correct result, add the following line to your .bash_profile (javac MUST work – it is the compiler):

PATH="/usr/jdk/jdk1.6.0_02/bin:${PATH}" ; export PATH

4. If CLASSPATH does not include JavaLang within your login directory (LIKELY!), add the following line to your .bash_profile:

```
CLASSPATH="${HOME}/JavaLang" ; export CLASSPATH
```

5. Save your updated .bash_profile file, log out of your UNIX account, log back in, and run the above tests again. You should now be able to run “javac –version” and “java –version” correctly, and JAVA_HOME and CLASSPATH should be set correctly.

10. An emacs user reported that the shell could not find emacs. If you want to run emacs and it does not work, add the following to .bash_profile

```
PATH=“/opt/csw/bin:${PATH}” ; export PATH
```

then log out and log in.