CS206 Introduction to Data Structures

Lab 3

Reading files (again), using ArrayLists, etc

Thursday, Feb 6

UNIX

Two topic for today: copying files and history.

Unix History: Unix remembers the last 500 (or so) commands you issued in a terminal. To see them all:

```
    history
```

This can be useful if you are trying to answer questions like "where did I put that file" or "how did I do that thing I did last week". Often even more useful is that you can go though the history one item at a time using the up and down arrow buttons. When you get to a command you want, reexecute it by just hitting return! You can also edit the line using right-left arrows and the backspace key. (I use this all the time)

Copying files: the UNIX comment to copy files is cp. For instance

```
    cp x y
```

would make a copy of the file named x under the name y (assuming x exists in the current directory).

```
    cp DIR1/x DIR2/y
```

makes a copy of the file named x that is in the the directory DIR1 and puts that copy into a file named y in the directory DIR2. (Assuming x, DIR1 and DIR2 exist.)

```
    cp DIR1/*.java DIR2/
```

will make a copy of every file that end with .java in DIR1 and put those copies into DIR2 (assuming DIR1 and DIR2 exist.) The * is a UNIX wildcard, it allow you to say something like "all" or in this case all that ends in .java. Further specialization is possible — for instance A*.java would copy copy on those files that start with A and end in .java

Exercise 2: ArrayLists and reading files: The **balance sheet of a Day Trader**

Day Traders buy and sell stocks, frequently. Some will make more than 1000 trades in a day. At the end of the day, they need to know their "position" — that is the number of shares they hold of which stocks. In this lab you will implement a system for tracking day trading positions, and printing out those positions at the end of the day. (Day traders also need to know their cash position, profit/loss etc. We are only going to deal with the number of shares they hold.)

Suppose that there are day traders who record all of their trades in a single file that has one transaction per line. Each line is of the form:

symbol amount

where:

symbol: is a stock symbol (for example IBM)

amount: is an integer, either positive or negative (for example -400).

For example, assuming that the trader starts the day with nothing, and that the file contains:

```
IBM +200
MSFT +100
IBM -400
```

then at the end of the day their position would be:

```
MSFT 150
IBM —200
```

Yes, negative positions are allowed. In the stock market this is called "short selling". For details, watch the movie "The Big Short".

For this exercise write a Java program to read all of the transactions from the file

```
/home/gtowell/Public206/lab03/bigtrades.txt
```

and then print out all of the non-zero positions. You should read the transaction file only once, so you will need to store information in a data structure. Specifically, use an ArrayList.

For easier debugging, there are much smaller versions of this file in:
`/home/gtowell/Public206/lab03/trades.txt`
`/home/gtowell/Public206/lab03/microtrades.txt`
microtrades.txt has the data in the example above

To complete this exercise do the following (this is a suggested set of steps, you can follow your own path instead):

1. Create a class (give it a likely name like Position, you may use any name that works for you) that has instance variables for holding a stock symbol and the number of shares. It should have accessor methods as needed.

2. Create a second class with a likely name like DayTrader. Within that class create an ArrayList that holds instances of Position. Assume that the day trader starts with nothing, so the ArrayList is initially empty.

3. Read the file line by line — you can use much of the code for reading a splitting files from lab 2 or from WordCounter.java from class on Jan 30. Recall (from class) that to split a string based on spaces in java you would write:

```
String line = "To be or not to be";
String[] spSplit = line.split("\\s+");
```

With a line:

   a. Create an instance of Position containing the data on that line
   b. Search through the ArrayList of existing positions for one with the same symbol name.
   c. If one exists, update it with the number of shares in the just created Position instance. Otherwise add the Position to the ArrayList.

4. After reading all lines in the file, print all entries in the ArrayList that have a non-zero number of shares.

When you are complete turn the first page of this lab handout with your name and the final position of the Day Trader based on the bigtrades.txt file. If you could not complete this exercise, turn in a page with a note describing how far you got.