CS 310 Data Structures
Programming Assignment 0
Spring 2021

Due Wed., Feb 17 by midnight (i.e., end of this day) See Delivery section at end of this document for starred files due at this time. Note Mon., Feb.16 is a holiday.

This assignment aims to help you:

- Learn about Maps (Symbol Tables), and get experience with their JDK classes
- Think about performance and memory use

Reading

Read S&W Section 3.1 on symbol tables (maps), Section 3.5 on applications including text indexes. If you have Weiss, read Weiss Chap. 6 through Section 6.8 on Collection classes. Read the JDK API docs on at least Map, HashMap, and TreeMap.

Project Description: also see Suggested Steps for more information.

A simple application using JDK collection classes.
The main server at your company maintains 500 terminal lines, numbered 1 through 500 inclusive. Every 10 minutes the system appends to a log file the terminal line numbers and user name of the person currently logged on, one terminal line number and username per line of text. (Not all lines are in use at every moment.) A fragment of the log file looks like this:

9 ALTEREGO
12 ALIMONY
433 HOTTIPS
433 USERMGR
12 BLONDIE
433 HOTTIPS
354 ALIMONY

This log file shows HOTTIPS was on line 433 twice but USERMGR was on that line at an intervening time. It also shows that ALIMONY was on line 12 at one time and line 354 later on. The log does not display the time at which each line was written. Your job is to write a program in Java that meets the following specifications.

The program first reads a log file into memory, storing the input data as it encounters it. Assume IO redirection when your program is invoked, so it can read from System.in rather than by actually opening a named textfile. User names are ASCII (plain text) strings with no embedded whitespace and a maximum length of 40 characters.

After all the data has been read your program should print data about terminal line usage: a header line and then one line of output for each terminal line showing the terminal line number, the most common user of that terminal line (in the event of a tie, choose one user), and a count of how many times that user was on that terminal line. Here is sample output:

Line Most Common User Count

1 OPERATOR 1749
2 HANNIBAL 432
3 NONE 0
4 HOTTIPS 945
...

Implementation. Since the lines are numbered consecutively, it is easy to use an array with one spot for each line, leaving spot 0 empty. Each line needs a container of user-count data, i.e. how many times each user has been seen on that line. Create a class named Usage to hold a single username and its count together. Note it is almost the same as WordUsage of homework 1. Create a class named LineUsage to hold all the data on one particular terminal line. In this LineUsage class, use a Map from the JDK (HashMap or TreeMap), mapping each username to its count, an Integer. See the provided program FrequencyCounter.java for an example of using a Map to hold a count for each of many strings. LineUsage should have methods addObservation(String username) and Usage findMaxUsage(), which returns the Usage object for the user with the highest count. Note that Usage is not used in the map, only in the delivery of results once the Map is full of data. TestLineUsage is a simple test of LineUsage: just create one LineUsage object, add data to it, and print it out. The top-level class LineReport has an array of LineUsage objects to hold all the data on all the lines. See page 72 of S&W for discussion of an array of objects. LineReport.java should have several methods (at least two object methods or static methods), including a main() to run the program. Each method should be commented (a comment above the method to describe what it does).

Directory setup. In accordance with current Java programming practice, each project in this class will held in a directory system with a top level directory named by the project, pa0 for this initial project, using zero as its number, and within that directory, there will be a src directory for all Java sources. If you are using an IDE, the binaries will be separately held in directory bin. But if not, it's OK to allow the .class files to intermix with the .java sources in src. The simple case (no IDE in use) will have the following files, on users.cs.umb.edu, or at home. The cs310 directory is supplied for you on users.cs.umb.edu, so you only need to create the pa0 within it, then the src within pa0.

- cs310/pa0/src/Usage.java
- cs310/pa0/src/LineUsage.java
- cs310/pa0/src/TestLineUsage.java
- cs310/pa0/src/LineReport.java

Here forward slashes are used for directories. Under Windows, back slashes are used for directories, but we will consistently the Linux/Mac syntax of forward slashes: just reverse the slashes for Windows. You can create these on your home system, then transfer the whole pa0 directory system to users.cs.umb.edu. See AccessToCSSystems for file transfer instructions.

To build LineReport, cd to the src directory and use the following command, on Linux/Mac/Windows. The reason to be cd'd to this directory is that this is the easiest way to get the right java "classpath" in use, the search path for input files to the compiler (here *.java) or the java command (here LineReport.class). The default classpath for java and javac is the current directory, and you've cd'd to the right place to set the desired classpath this way.

```
javac *.java    (while cd'd to src)
java LineReport < inputfile
```

For an IDE like eclipse, set up the directories as above (with src and bin), put the sources in src, and then set up a project in the IDE. Eclipse can "open project from file system" once it is set up.

memo.txt

In the file memo.txt, answer the following questions, in one to two pages (60-120 lines) of text. Use complete sentences, as you would for an English class

1. Discuss your experiences in writing these programs. What development tools (IDE, etc.) did you use? Did you develop on cs.umb.edu servers, or if on your own PC/Mac, did you have any problem recompiling and running on Linux?

2. Analyze the big-O CPU performance of LineReport, for N input lines and O(N) different users, and thus O(N) entries on each list once partially done.

3. Show the command line compile and run using commands as shown above, and display of directories by the Windows tree command or Linux/Mac du command.

Delivery

Before the due time, assemble files in the pa0 (lowercase pa, number 0) subdirectory of your provided cs310 directory on our users.cs.umb.edu Linux site. Make sure they compile and run on Linux! They should, since Java is very portable. It's mainly a test that the file transfer worked OK.

- pa0/memo.txt (plain txt file, try "more memo.txt" on users to make sure it's readable on Linux)
- pa0/src/Usage.java
- pa0/src/LineUsage.java
- pa0/src/TestLineUsage.java (correction to original list)
- pa0/src/LineReport.java

Check your filenames: login on cs.umb.edu Linux: cd cs310/pa0, then ls should show memo.txt, src, and possibly bin. ls src should show the three .java files. Remember that case counts on UNIX/Linux!