CSC172 LAB 17

HASH MAPS

1 Introduction

The labs in CSC172 will follow a pair programming paradigm. Every student is encouraged (but not strictly required) to have a lab partner. Labs will typically have an even number of components. The two partners in a pair programming environment take turns at the keyboard. This paradigm facilitates code improvement through collaborative efforts, and exercises the programmers cognitive ability to understand and discuss concepts fundamental to computer programming. The use of pair programming is optional in CSC172. It is not a requirement. You can learn more about the pair programming paradigm, its history, methods, practical benefits, philosophical underpinnings, and scientific validation at http://en.wikipedia.org/wiki/Pair_programming.

Every student must hand in their own work, but every student must list the name of their lab partner if any on all labs.

This lab has six parts. You and your partner(s) should switch off typing each part, as explained by your lab TA. As one person types the lab, the other should be watching over the code and offering suggestions. Each part should be in addition to the previous parts, so do not erase any previous work when you switch.

The textbook should present examples of the code necessary to complete this lab. However, collaboration is allowed. You and your lab partner may discuss the lab with other pairs in the lab. It is acceptable to write code on the white board for the benefit of other lab pairs, but you are not allowed to electronically copy and/or transfer files between groups.

2 HashMap Application

The Hash table ADT supports various set operations. In this lab, you will use the Java Standard Library HashMap implementation of hash table to efficiently look up city names given a US ZIP Code.

1. Begin this lab by reading the Java documentation of HashMaps.
2. Using proper generics, declare a HashMap that maps Integers into Strings. We will be storing key-value pairs of <ZIP Code, City Name> into the HashMap.
3. Write a main method that reads in the `zipcodes.csv` file and inserts each ZIP code mapping into the HashMaps. For example, the entry for UR's campus in the CSV file is “14627,Rochester,NY”, which should insert <14627, “Rochester, NY”> into the HashMap.

Once you have finished inserting all of the ZIP codes, print out the size of the HashMap.

4. To demonstrate that your HashMap is working, add a loop to your main method that asks the user to input a ZIP code and print out the corresponding city and state for the user. You should gracefully handle invalid inputs (such as 4-digit numbers, text, zip codes that do not exist in the HashMap, etc.). When the user enters “quit”, the loop should terminate.
5. Next, we want to know how many ZIP codes a given city has (but not necessarily which ZIP codes). Create a second HashMap that maps Strings to Integers. Iterate through the values (i.e. city names) of the original zip code mapping and add an entry for each city into the new HashMap with a frequency of 1. If the city has already been inserted, increment its frequency in the HashMap instead.
6. Add a new loop to the main method to let the user enter city names and print the total number of ZIP codes associated with that city. For example, if the user enters “Rochester, NY”, your program should print 47. Pleasant error messages should be printed if the user enters a city name that is not in the HashMap. Again, quit the loop when the user enters “quit”.

3 Hand In

Hand in the source code from this lab at the appropriate location on the BlackBoard system at my.rochester.edu. You should hand in a single zip file (compressed archive) containing your source code, README, and OUTPUT files, as described below.

1. A plain text file named README that includes your contact information, your partner's name, a brief explanation of the lab (A one paragraph synopsis. Include information identifying what class and lab number your files represent.), and one sentence explaining the contents of all the other files you hand in.
2. Java source code file(s) representing the work accomplished for this lab. All source code files should contain author and partner identification in the comments at the top of the file.
3. A plain text file named OUTPUT that includes author information at the beginning and shows the compile and run steps of your code. The best way to generate this file is to cut and paste from the command line.

4 Grading

90% Functionality

45% ZIP code to City HashMap implementation and testing

45% City to Zip Code Frequency implementation and testing

10% README and OUTPUT files