CS 122 – Lab #11 – Using Java's Hashtable data structure

In today's lab you will complete the implementation of a useful program that makes use of an interesting data structure: Java's Hashtable, which is analogous to the dictionary in Python. In other words, we want to reference individual elements in our collection by means of a "key" rather than an index number. The Hashtable will store data as ordered pairs (key, value), in which each key is unique. Behind the scenes, this key is converted into a hashCode using a numerical process, and then this hashCode is converted into an index to be used by the actual array representation. The value in the (key, value) pair is inserted into the array at this index.

Please create a new folder called lab11 and copy all my files from the lab11 folder on the class Web site http://cs.furman.edu/~chealy/cs122/ into your folder. You only need to make changes to Driver.java. Follow the specific directions given in the comments.

You may want to briefly review the syntax on how to use the Hashtable class. The online Java API documentation is an excellent source. Also, I have created a hashtable folder on the class Web site showing an example program using a Hashtable. Some of the useful methods include get, put, replace, remove, containsKey and keys.

The purpose of this program is to read an input file that stores user login information for a computer system. Each line of the file contains a user's name and an amount of time logged in. See the file login.txt. The program should read the file, and create a Hashtable of users. The keys will be the user names, and the values will be the minutes of computer time that the person was logged in.

As you read in the data, you need to note if you have ever seen this person before. If the person is already in the hash table, then update that person's value. Otherwise insert a new name/minutes pair into the hash table.

At the end of the program, dump the contents of the hash table into an ArrayList and sort the users in descending order of login time. I have provided a comparator class UserComparator.java to help you do this. Finally, print the users' info, one person per line.

Save and run your program, and make sure the output makes sense.

Finally, consider this question. Why did we use a Hashtable, instead of using an ArrayList throughout? If we had used an ArrayList from the beginning, what performance disadvantage would we incur? In other words, can you point to a place in your implementation where using a Hashtable is faster than doing the same operation using an ArrayList? What is it? _____

_____

_____