Homework 4 - Due Saturday, June 25th

Eclipse Suggestion

We suggest you create a Homework4 Java project in Eclipse and unzip hw4.zip into the src folder. Use the same process as stated in homework 3 for setting up JUnit.

Part 1: Completing HashtableMap

You have been given a skeleton for HashtableMap along with two classes full of JUnit tests. Your task is to complete the HashtableMap class so that it implements a hashtable that handles collisions using chaining. To simplify the implementation (and to practice using LinkedList and iterators) you must use an ArrayList of LinkedLists as the underlying bucket table. This has already been declared for you in the skeleton. We have also given you a getBucket function that computes the bucket index when given a key object and the size of the bucket table (i.e., getBucket is the hash function).

Some implementation suggestions:

1. Write getEntry early on since most of the other methods can be made easier by calling getEntry.

2. Values can be null and this must be handled in containsValue. You can shorten your code in containsValue by using Objects.equals. Look it up in the Java API documentation for more info.

3. A few of the methods will require iterating over LinkedLists, so review the iterator examples from Lecture 8.

Resizing

Your table must implement resizing. After any put method that adds a new key, you must check if the new load factor is strictly larger than the threshold specified in the constructor. If so, you should double the size of the table and appropriately fix all of the chains. As a suggestion, create a new ArrayList of the correct size and then move all the entries over into their correct positions.

Part 2: Using Java’s HashMap

You must implement a Java program called PortfolioStats that reads a list of Stock trades and outputs your final positions. The list of trades will be given to your program using standard input (i.e., System.in). Each line of the input will have the form
StockName, TradeSize

StockName will be a string of capital letters and TradeSize will be an integer (possibly negative). You must output to standard output with each line having the form

StockName, FinalPosition

where FinalPosition is the sum of all the TradeSizes for that given StockName. There should be no extra spaces in the output. Each distinct StockName in the input should appear exactly once in the final output. You are guaranteed that all final and intermediate positions will fit into an int. The output must be ordered in the following way:

1. A Stock whose position has a larger absolute value comes before a Stock whose position has a smaller absolute value.

2. Break ties in the previous rule by ordering the StockNames in alphabetical order (this is what the instance method String.compareTo does already).

Here are some implementation suggestions:

1. Use a HashMap to track the positions.

2. Once you have collected all of the final positions, you can iterate over the HashMap by first getting the entrySet() and then iterating over that.

3. Create an ArrayList that stores the StockName-FinalPosition pairs, and create the appropriate Comparator/Comparable that lets you sort them as desired.

Testing Part 2

We have supplied two input files testdata1in.txt and testdata2in.txt with the corresponding correct output files testdata1out.txt and testdata2out.txt, respectively. You can test your code on the command-line using the techniques you learned in homework 2. This will be exactly how your program is tested by the grader.

Mac Version

java PortfolioStats < testdata1in.txt > mydata1out.txt
diff mydata1out.txt testdata1out.txt

Windows Version

java PortfolioStats < testdata1in.txt > mydata1out.txt
FC mydata1out.txt testdata1out.txt

What you submit

You should submit HashtableMap.java and PortfolioStats.java in a single zip file that is uploaded to NYU Classes. Please name your zip file hw4_yourNetID.zip