PA8 Generic HashMap

Due: Wednesday 3/30 by 11:30PM
Submission:

- MyHashMap.java - generic implementation of a HashMap.

Overview

The purpose of this assignment is to gain more practice with generics and hashTables. Hashing is an important tool in a computer scientist's toolbelt. Not to mention, it seems to come up in just about every software engineering interview. Your job is to implement a generic implementation of the HashMap class.

Assignment

Written

There are no written questions on the PA. But please make sure that you think about and know the answers to the below questions:

- What are the runtimes of the different functions I implemented for the MyHashMap class?
- Are the above runtimes always the case? Or can things get really bad if we have a bad hash function or are working with very weird data?
- What makes a good hash function?

Code

We did not provide an Eclipse project as starter code for this PA. We want to be sure that you know how to create your own project. Having said that, we are providing you with a very useful file. Take a look at drill06, it was meant as a warm up. The drill contains a file called HashNode.java. We encourage the use of this file on the project. In fact, we will automatically include it in the grader. **So do not make changes to this file.** Those changes will be overwritten with the file from our autograder.

We have gone over how to create a project in Eclipse twice in class, but it was a while ago. I will include a quick demo in Friday's (3/25) class on how to create a project. Although it is a quick/easy process that you should be able to find on google.

Recommended Approach

Remember iterative development? This would be a great project to break down into a few smaller steps. We recommend something like the below steps.

1. Write unit tests for hashmap usage. You could even use old PAs or section problems that use hashMaps. For now, have these tests use the Java HashMap class, so they should all pass.

2. Implement a NON-generic MyHashMap that has String as the key type and Integer as the value type.

- Copy in the given hash function
- Consider which data structures to use to implement the hash table
- Assume a perfect hash initially (will lose some data when keys conflict)
- Implement chaining to deal with collisions.

3. Change the non-generic MyHashMap into a generic MyHashMap.

Required Methods

Your MyHashMap class must have the below methods. If you are curious about the documentation or what the below methods do, consult the documentation.

- clear
- containsKey
- containsValue
- get
- isEmpty
- keySet
- put - note the return type in the documentation above.
- remove - note the Java HashMap class has two versions. You only need to implement the one with the signature: remove(key). Pay attention to the return type in the documentation above.
- size
- printTable - this **prints** the output. It does **not** return a string. Consult the documentation linked above.

The only method not taken from the Java documentation is printTable. printTable() should output how many conflicts occur at each bucket and list the keys in that bucket. **No matter what, your hashTable should always have 8 buckets!** An example output for printTable is shown here:

```
Index 0: (0 conflicts), []
Index 1: (0 conflicts), []
Index 2: (0 conflicts), []
Index 3: (0 conflicts), []
Index 4: (0 conflicts), []
Index 5: (0 conflicts), [ExampleKeyX, ]
Index 6: (0 conflicts), [ExampleKeyY, ]
Index 7: (0 conflicts), []
Total # of conflicts: 0
```

Implementation

In general, there are multiple ways to implement a hash map. Here are three:

- A linked list that contains a linked list for each bucket (2d linked list)
- An array that contains an array for each bucket (a 2d array)
- An array that contains a linked list for each bucket

However, since you are implementing a generic hash table in Java and generics are problematic in arrays (https://docs.oracle.com/javase/tutorial/java/generics/restrictions.html), **we recommend you use an ArrayList of linked lists.** The linked lists can be the Java LinkedList class or your own linked list implementation where the key,value pairs link to each other, whatever you prefer.

Your hash table implementation should be called MyHashMap.

Hash Function

Here is the code for the hash function. You can and should use this exact code in your project. **Make sure you understand what it is doing and how it works.**

```
private int hash(K key) {
  int hashCode = key.hashCode();
  int index = hashCode % numBuckets;
  return Math.abs(index);
}
```

Collisions

This hash map will use chaining to handle the collisions of elements. **Your hash map should only have 8 buckets!** Collisions will happen.

PUT the (key,value) pairs at the FRONT of the list in the bucket.

If the same key is passed in with a different value, do the value update in place.

Grading Criteria

We are not providing testcases for this PA.

We encourage you to write your own JUnit testcases to ensure your classes work properly, but we will not be collecting or grading these test files. We will test your classes with our own testcases.

Your grade will consist of similar style and code clarity points we have looked for in earlier assignments.

Write your own code. We will be using a tool that finds overly similar code. Do not look at other students' code. Do not let other students look at your code or talk in detail about how to solve this programming project. Do not use online resources that have solved the same or similar problems. It is okay to look up, "How do I do X in Java", where X is indexing into an array, splitting a string, or something like that. It is **not** okay to look up, "How do I solve {a programming assignment from CSc210}" and copy someone else's hard work in coming up with a working algorithm.