Course: CSC 115 (J)

Lab 10: Hashtable with Separate Chaining

Objectives

You will write a more advanced hashtable that uses **separate chaining**. Each index in the array will store a linked list of items, leading to better efficiency when inserting items into the table.

During lab 10, you will:

- Evaluate the lab instructors and the lab environment.
- Write a hashtable that uses **separate chaining**.
- Practice navigating the Java API to figure out which LinkedList methods can be used as part of your implementation.
- Consider how this new hashtable compares to the one written in lab 9 that used linear probing to resolve collisions.

Part I - HashtableSC

1. Download *KeyValuePair.java* from connex.
2. Download and complete *HashtableSC.java*.
    - Start by inpsecting the *KeyValuePair* class. The hashtable will store *KeyValuePairs*, and each *KeyValuePair* stores an *Object*, and an integer key that identifies that object.
    - The *HashtableSC* class already includes some basic methods. Most importantly, it stores an array of linked lists of *KeyValuePair* objects which serve as the table, along with integers storing the capacity of the *hashtable (tableSize)*, the number of items currently stored in the *hashtable (size)*, and the number of collisions that occur while inserting into the *table (collisionCount)*.
    - Note that when you compile the code, you will get a **warning**. Ignore this warning; it is a consequence of using an array of generic types.
    - Spend a few moments reading through the provided methods. Make sure you generally understand what they are doing before you begin writing code.
    - Your job is to fill in the *insert*, *find*, and *delete* methods.
        - We are using LinkedList from the Java API for this lab, and you may find it useful to look at the methods they have available: http://docs.oracle.com/javase/6/docs/api/java/util/LinkedList.html. In particular, you will likely use the *addFirst*, *toArray*, and *remove* methods.
        - Make sure to **test** each method as you finish them by compiling and running the program. You will be able to test the different methods by uncommenting parts of the main method.
    - Consider how this hashtable relates to the one you wrote in lab 9. Does this one offer better or worse performance in terms of the number of collisions? What about when you change to using *hashFunction2* instead of *hashFunction1*?

Part II – Assignment 05

This part of the lab is dedicated to work on your last assignment. Ask you TA questions in the assignment OR if you have any queries / concerns in the materials covered previously.

Good luck

Have fun