Osayi Odiase

Ladelle Diekhoff

Computer Science 121

Submission: 5/1/2023

Final Exam HashMap Lab Report

Introduction

A data structure in programming is a way to store, organize, access update, and process data from a computer memory. They are important in the study of computer science because they allow us to store and access arbitrary objects for efficient use in making software.

The data structure I will be doing my lab report on is Java's HashMap data structure. The HashMap is a collection type data structure that stores and accesses data with key value pairs.

Implementation

The way a HashMap works to the naked eye is by storing key value pairs into the HashMap structure and retrieving the data efficiently. On the inside though, HashMap works by using a hashing function that turns the key into a string of numbers or characters that associates with the value. The size of the hash code is what is searched for to correspond with its value. So, if the result of hashing gives a 32-bit string of characters when indexing, we will search for a key that is 32-bits in size. The time complexity of a HashMap is O(1) constant time. That means that no matter the size of a HashMap the time it takes to insert and search for an element takes the same amount of time to complete.

There are multiple functions and ways to hash values like SHA or MD5, and different data structures, algorithms, and programs may use multiple hashing methods to map values. A hash value would look something like this:

SHA-1 hashing “CS121”

‘6c8d251737950e8b24ab1f2226240181676412bc’

MD5 hashing “CS121”

‘4134b68ce81b485bbd96b72b01e8f8db’

Here are a few common methods a to use on a HashMap, of course these not being all of the methods of a HashMap:

.put(Object key, Object value)

- Put items into a HashMap

.remove(Object key), .remove(Object key, Object value)

- Remove a value from a HashMap by referring to its key
- Also remove an item from a HashMap by referring to its key and mapping

.clear()

- Removes all items from the HashMap

.get(Object key)

- Get the value of an object by referring to its key

.isEmpty()

- Returns a Boolean to check weather a HashMap is empty or not
- True if empty and False if not empty

Of course, the advantage of using the HashMap data structure is that you can store data in key value pairs, and the key can't be duplicated which means you can have data that is similar and stored to one key. Another advantage, or disadvantage, if you know anything about threads and processes, is that a HashMap does not use multi-threading which could cause an unsynchronized HashMap or corrupt your data since you cannot change data in two or more different threads.

A disadvantage is that when hashing, the return value is based on the size of the hashing function, not the actual value that is input, which causes collisions. For example:

You have a key of “banana” and a key of “123”

When put into the hash function they may return different hash codes but are both 32 bits in size.

When indexing through the HashMap, the two keys could be mapped to the same index, which means two keys being mapped to the same value, which is a collision.

There are of course ways the HashMap resolves this problem, one way that the HashMap deals with this is that the hash is turned into a Node head and the values are then linked together to form a linked list. The key is then hashed again and then indexes the linked list for the corresponding value, but this can be slow if the linked list gets too long while indexing for the correct value.

Real Work Problem

Being able to have unique items or data is useful in the real world where you need to keep things ‘personalized’. The HashMap can be used for registrations where you need a unique login or key to be able to access the user's data. As an example, registrations such as Student Registrations for school is an example. In order to access your account(value), you need a login(key) to access it. The login is unique because only one student can have an account at one time, but the accounts may have similar data like there are students in the same year, or students in the same class. Alternatively, you could store the students' data by year(key) and search for the students(value), via linked list, in that year to find the student you are looking for.

Conclusion

To summarize my findings for the HashMap Lab Report, are how hashing works and how the keys are stored by size and not by the actual key value, how the different methods work to store or access data, and how the data structure could be used to store large data sets in constant time. I had a rough idea of HashMaps, but this really made things clear on how keys and values are able to store objects not single data types. I also learned about how hashing works and how data is indexed with a code size rather than the actual key value. The HashMap and Hash family is one of the core data structures in your journey as a software engineer and this lab report made this clear about the HashMap for me.