Map [Interface]: HashTable

Definition

This class implements a hash table, which maps keys to values. Any non-null can be used as key or as a value. To successfully store and retrieve objects from a hashtable, the objects used as keys must implements the hashcode method and the equals method. Hash function helps to determine the location for a given key in the bucket list. To determinate whether two objects are equal or not, hashtable makes uses of the equals() methods. HashMap implements the Map interface.

An instance of HashTable has two parameters that affect its performance: initial capacity and load factor. The capacity is the number of the buckets in the hashtable, and the initial capacity is simply the capacity at the time the hashtable is created. The load factor is a measure of how full the hash table is allowed to get before its capacity its automatically increased.

The default load factor is 0.75 offers good tradeoff between time and space cost. The initial capacity controls a tradeoff between wasted space and the need for rehash operations, which are time consuming. No rehash operations will occur if the initial capacity is greater than the maximum of entries the HashTable will contain divided by its load factor.

Features of HashTable

- It is similar to HashMap, but is synchronized
- HashTable stores key/ value pair in hash table.
- In HashTable we specify an object that is used as a key, and the value we want to associate to that key. The key is hashed and the resulting hash code is used as index in which value is stored within the table.
- The initial default capacity of HashTable is 11 whereas loadfactor is 0.75.
- HashMap doesn't provide any enumeration.

Type parameters

- K – The type of keys maintained by this map.
- V – the type of mapped values

Constructor summary

HashTable()	Creates an empty hashtable with the default load factor of 0.75 an initial capacity of 11.
HashTable(int initalCapacity)	Creates an hashTable that has an initial size specified by initialcapacity and the default load factor is 0.75.
Hashtable(int size, **float** fillratio)	Creates a Hash Table with initial size specified by size and fill ratio specified by fillratio.

HashTable(**Map** < ? **extends K**, ? **extends V**>**m**)	This creates a hash Table that is initialized with the elements in m.

Methods summary

Modifier and Type	Method and Description
V	**put**(**k** key**, V** value) Maps the specified key to the specified value in this hashtable.
V	**remove**(**Object** key) Removes the key and its corresponding value from this hashtable.
V	**replace**(**K** key, **V** value) Replace the entry for the specified key only if it is currently mapped to some value.
void	**forEach**(**BiConsumer** < ? super **K**, ? super V> action) Performs the given action for each entry in this map until all entries have been processed or the actions throws an exception.
boolean	**containsValue**(**Object** value) Returns true if this hashtable maps one or more keys to this value.
Enumeration<V>	**elements**() Returns an enumeration of the values in this hashtable.
Collection<V>	**values**() Returns a Collection view of the values contained in this map.

When is its use recommended?

- It recommended for password verification where a hash value of your password is sent to the server and compares that hash with the hash value in the stored password. It's authenticated when its equals.
- Used when comparing an pattern in a pool of strings, can be used to detect plagiarism.
- HashMap is recommended when only unique keys are available for the data we want to store.
- Is used when searching for items based on a key and when it requires quick access time.