Collections

Table Of Contents

[1.1 Hash Map 3](#_Toc519596830)

[1.2 Iterating Hash Map 3](#_Toc519596831)

[1.2.1 For loop 3](#_Toc519596832)

[1.2.2 For Each Loop 3](#_Toc519596833)

[1.2.3 While Loop 4](#_Toc519596834)

[1.1 Hash code and equals method significance on Hash map 4](#_Toc519596835)

[1.2 Hash Table and Hash Map Comparison 6](#_Toc519596836)

[1.1 Hash Table and Hash Set 6](#_Toc519596837)

[1.2 Hash Table Comparison of Hash Set, Linked Hash Set and Tree Set 7](#_Toc519596838)

[1.3 Hash Map VS Concurrent Hash Map 8](#_Toc519596839)

[1.4 Concurrent Hash Map 9](#_Toc519596840)

[1.5 Blocking Queue 9](#_Toc519596841)

[1.6 Fail fast vs. Fail safe 10](#_Toc519596842)

[Ref Link 11](#_Toc519596843)

*Collections*

## Hash Map

It stores the data in the form of key value pair, whenever a new hash map object is created, the object has a default configuration of 16 buckets and threshold of 75% of the initial value which is 12.

When the put method is called to place the data into hash map the first step would be to check if the key is null, if yes then the value will be placed on to the bucket with index zero.

In the event if the key is not null, the first step would be top get the hash code of the key then the custom hash function will compute a hash value.

Method signature: h ^= (h >>> 20) ^ (h >>> 12)

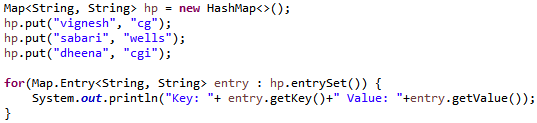
h ^ (h >>> 7) ^ ( h>>>4)

One the initial has value is generated, then we derive the index from the generated value using the below implementation ( h & number of bucket - 1).

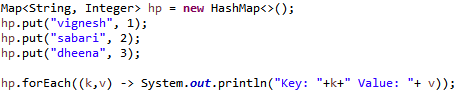
Similar logic is used while retrieving the data as well.

## Iterating Hash Map

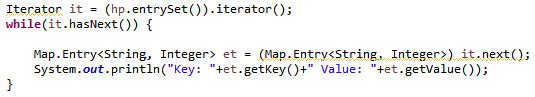
### For loop



### For Each Loop



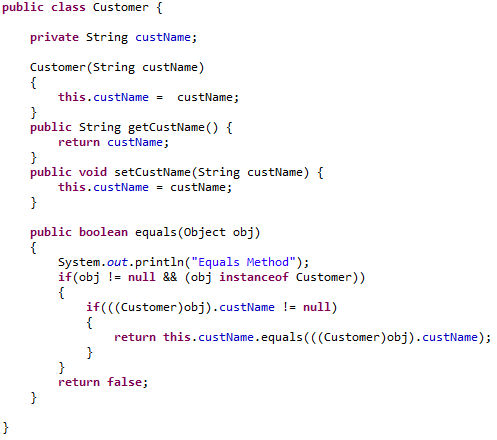
### While Loop

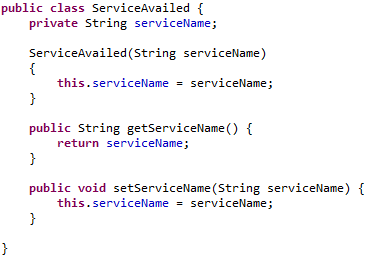


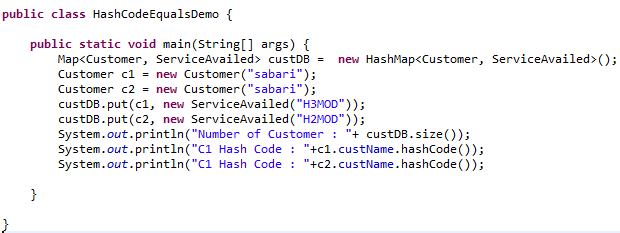
## Hash code and equals method significance on Hash map

The equals method defined in the object class has the default implementation evaluate if the given two objects are equal based on the value in the case of primitive data types in the event if it is non-primitive it would be based on the reference address.

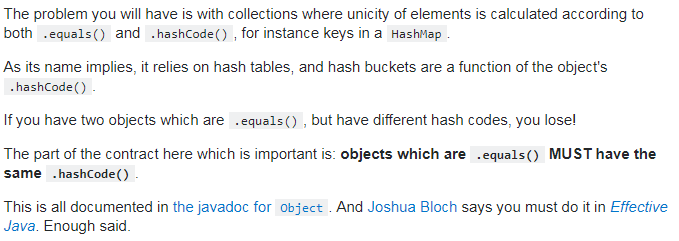
If default implementation of the method equals is used, observe the output of the programs.



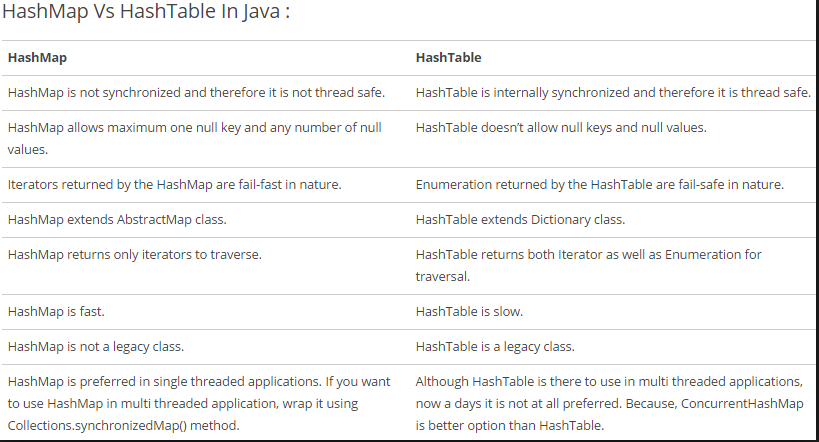




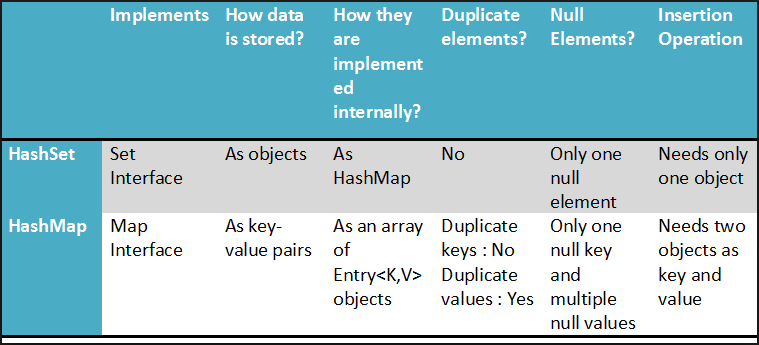
In the absence of the overridden equal and hashcode method the size of the hash map would be 2. In the presence of the same the size of the hash map will be one.



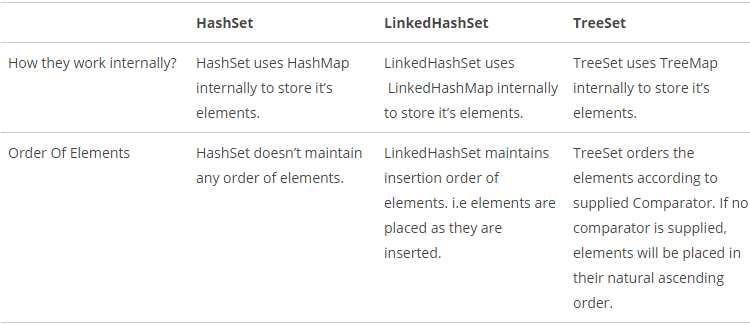
## Hash Table and Hash Map Comparison

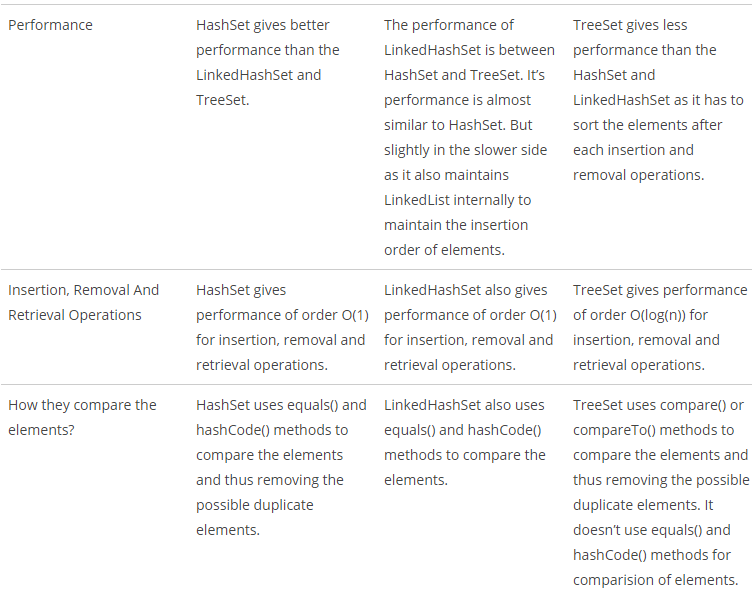


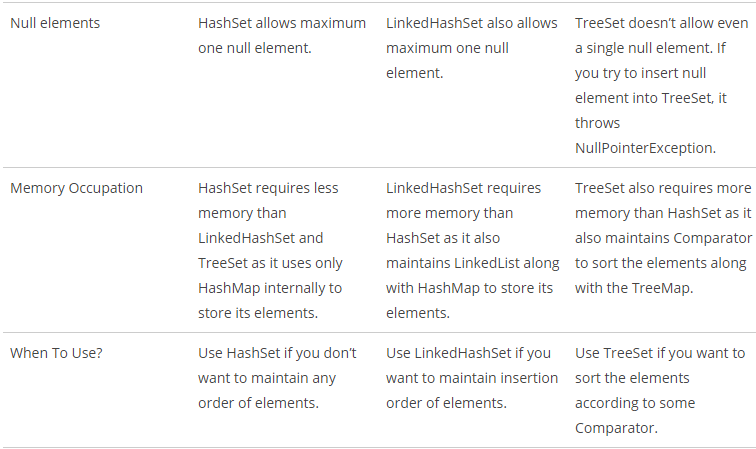
## Hash Table and Hash Set

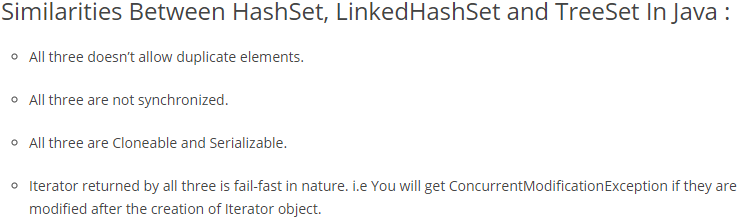


## Hash Table Comparison of Hash Set, Linked Hash Set and Tree Set









## Hash Map VS Concurrent Hash Map

The main difference between has map and concurrent has h map is that hash map is not synchronized whereas the concurrent hash map is synchronized.

The hash map can be synchronized using external synchronization (*by Collections.synchronize()* ) but we would end synchronizing the complete hash map which is not required. But when it comes to concurrent hash map only few segments are synchronized like the write operation and read operation is not synchronized.

In conventional Hash Map during the traversal if there is any write operation on the object say addition or modification or deletion, the action would result in a ConcurrentModification Exception. Whereas if the concurrent hah map incurs a change during traversal no exceptions will be generated But there are possibilities where the traversal will not be happening on the latest snap shot.

## Concurrent Hash Map

It basically allows concurrent access to the Map. Part of the map called “*Segment” (Internal Data Structure)* is only locked while performing insertion or updating operations. So the concurrent hash map allows multiple threads to read the map without locking at all hence increasing the overall performance.

***Concurrency Level***: The term defines the maximum number of threads that are allowed to write or update the contents of the map at any given point in time. By default the value is configured to 16 and can be modified through the constructor during the instantiation.

The concurrent hash map has internal final class called segments in other words by default the number of segments are equal to the number of buckets in the map with every bucket implemented with a lock.

When every a put method is called the hash code of the key is compared with the key that is already contained in the bucket if the hash code is same then the next step would be to compare the key itself if it is found to be true then the value is updated.

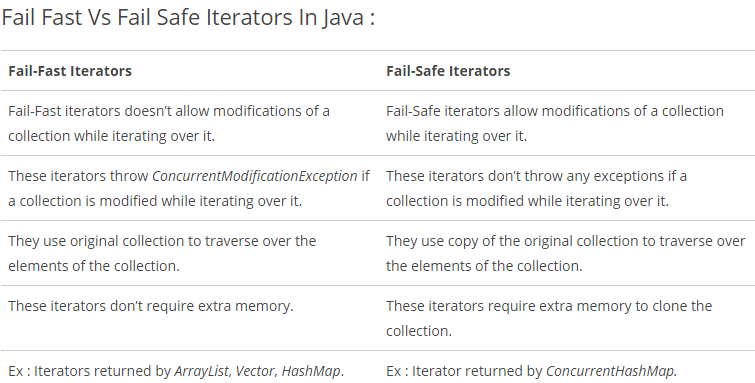
Concurrent hash map does not allow null keys or null values in order to avoid null pointer exception during run time.

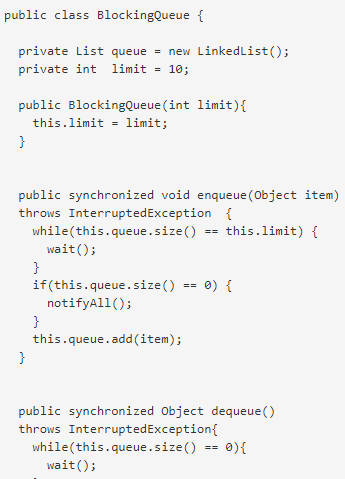
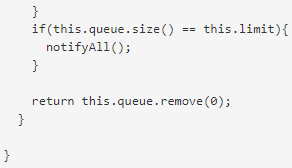
## Blocking Queue

It is a queue that blocks the thread when it tries to DE queue value when the queue is empty and grants access to the queue once the queue has some value which can be DE queue’ d and also it stops the thread from inserting value (EN queue operation) when the queue is full and provides access once the queue has space to accommodate value.

The internal implementation of the blocking queue is very much similar to the bounded semaphore concept.

## Fail fast vs. Fail safe



## Ref Link

1. Importance of hashCode and equals method:

https://www.youtube.com/watch?v=ghswNpRv2t0