Map Internals

Map stores Key and value pairs.

time complexity is o(1);

methods: get, set, delete

Map uses the array of nodes( named as table), where node has fields like the key value (and much more).

Here, the Node represented by MapEntry structure.

hash map calculates the index where data is stored using hash functions.

hash function can return same index for different pair of data, because of limited capacity,

when this happens, we called this as a Collision.

Yes, any 2 elements can have the same index of the table array.

Take for example the HashMap has initial size = 16(Default size)

Now suppose we insert one element (let it be named elementOne) and suppose index is calculated as 2 and hence it will be placed at index 2 of the table array.

If we insert one more (let it be named elementTwo) and suppose its index also comes out to be 2, then what will happen in that case?

It is the collision scenario.

To make HashMap work even in cases of collision we have LinkedList at the array indexes, i.e. as in the above example at index 2 we will have

elementTwo -> elementOne

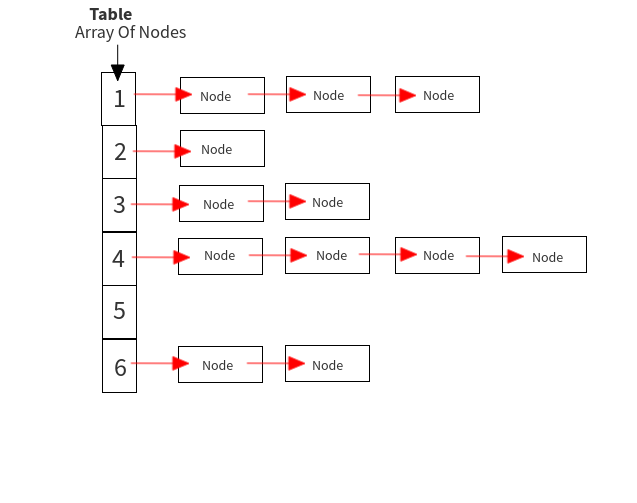
elementTwo

|  |  |
| --- | --- |
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |

elementOne

So as you must have guessed the Node structure must have these things at least:  
a. key  
b. value  
c. next

Here next is for the next element in the LinkedList.  
Apart from these 3, it also has the other field i.e. the **hash**. Hash is used while comparing 2 Nodes and for calculating the index of the Node



***Threshold****has the value which is****loadfactor\*capacity****Capacity is the maximum size of the HashMap*

*Threshold is set while the table is inflated as below:*

***float thresholdFloat = capacity \* loadFactor;  
if (thresholdFloat >*MAXIMUM\_CAPACITY *+ 1) {  
thresholdFloat =*MAXIMUM\_CAPACITY *+ 1;  
}  
threshold = (int) thresholdFloat;***

*So the threshold means if the current size exceeds this value, then the HashMap needs to be resized.  
Now threshold is 0.75\*capacity i.e. if the current of the hashmap is more than 0.75 of the total size set, then resize it.  
Also we can set the loadFactor, but its default value is 0.75*

*Why we need to****resize****the HashMap?  
Its because if the size is fixed and as more and more elements are added to the hashmap, then there will be more and more collisions and hence longer linkedlists maintained, hence iterating over them will not be with constant time anymore but may go upto O(n). So to spread the Nodes more evenly, resize the hashmap once the total number of nodes in it exceeds some threshold value.*

new Hashmap with new capacity (= double the previous capacity) is created, the table now points to it, and the new threshold is created.  
Finally, the old elements are transferred to this new Hashmap as

**What is hashing?**

Hashing is the process of transforming any given key or a string of characters into another value.

The most popular use for hashing is the implementation of hash tables. A hash table stores key and value pairs in a list that is accessible through its index.

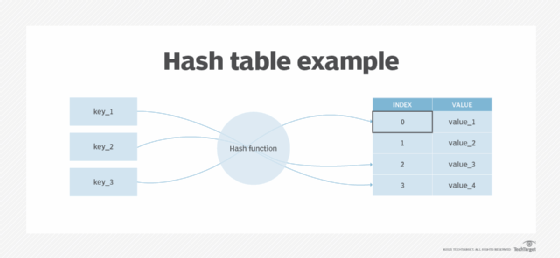
### What is hashing used for?

Hashing uses functions or algorithms to map object data to a representative integer value.

For example, in hash tables, developers store data -- perhaps a customer record -- in the form of key and value pairs. The key helps identify the data and operates as an input to the hashing function, while the hash code or the integer is then mapped to a fixed size.

Hash tables support functions that include the following:

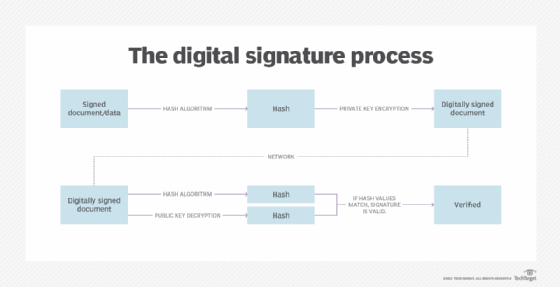
* insert (key, value)
* get (key)
* delete (key)



#### Digital signatures

In addition to enabling rapid data retrieval, hashing helps encrypt and decrypt [digital signatures](https://www.techtarget.com/searchsecurity/definition/digital-signature) used to authenticate message senders and receivers. In this scenario, a hash function transforms the digital signature before both the hashed value (known as a message digest) and the signature are sent in separate transmissions to the receiver.

Upon receipt, the same hash function derives the message digest from the signature, which is then compared with the transmitted message digest to ensure both are the same. In a one-way hashing operation, the hash function indexes the original value or key and enables access to data associated with a specific value or key that is retrieved.



### What is a collision?

Hashing in cybersecurity demands unidirectional processes that use a one-way hashing algorithm. It is a crucial step in stopping threat actors from reverse engineering a hash back to its original state. At the same time, two keys can also generate an identical hash. This phenomenon is called a [collision](https://www.techtarget.com/searchnetworking/definition/collision).

A good hash function never produces the same hash value from two different inputs. As such, a hash function that comes with an extremely low risk of collision is considered acceptable.

Open addressing and separate chaining are two ways of dealing with collisions when they occur. Open addressing [handles collisions](https://www.geeksforgeeks.org/hashing-set-3-open-addressing/) by storing all data in the hash table itself and then seeking out availability in the next spot created by the algorithm.

Open addressing methods include:

* double hashing
* linear probing
* quadratic probing

**Separate chaining**:

by contrast, [avoids collisions by making](https://www.geeksforgeeks.org/hashing-set-2-separate-chaining/) every hash table cell point to linked lists of records with identical hash function values.

To further ensure the uniqueness of encrypted outputs, cybersecurity professionals can also add random data into the hash function. This approach, known as "[salting](https://www.techtarget.com/searchsecurity/definition/salt)," guarantees a unique output even when the inputs are identical.

Salting obstructs bad actors from accessing non-unique passwords. This is because each hash value is unique, even when users reuse their passwords. Salting adds another layer of security to thwart [rainbow table](https://www.techtarget.com/whatis/definition/rainbow-table) attacks.

Hashing can also be used when analyzing or preventing file tampering. This is because each original file generates a hash and stores it within the file data. When a receiver receives the file and hash together, it can check the hash to determine if the file was compromised. If someone manipulated the file in transit, the hash would reflect that change.