java.util.Map interface represents a mapping between a key and a value. The Map interface is not a subtype of the Collection interface

- java.util.HashMap
- java.util.Hashtable
- java.util.EnumMap
- java.util.IdentityHashMap
- java.util.LinkedHashMap
- java.util.Properties
- java.util.TreeMap
- java.util.WeakHashMap

HashTable (Obsolete)	HashMap	IdentityHashMap	LinkedHashMap	TreeMap
O(1)	O(1)		O(1)	O(log(n))
Map	Map	Map	Map	Map, SortedMap, NavigableMa
Synchronized	Collection.synchronizedMap			
	Buckets	Compared using ==	HashTable and LinkedList using doubly linked list of buckets	Red-Black Tree

Collision:
Since a hash function gets us a small number for a key which is a big integer or string, there is possibility that two keys result in same value. The situation where a newly inserted key maps to an already occupied slot in hash table is called collision and must be handled using some collision handling technique.

How to handle Collisions?
1) Separate Chaining
- The idea is to make each cell of hash table point to a linked list of records that have same hash function value.

2) Open Addressing
- all elements are stored in the hash table itself.

Simple Chaining - Advantages:
1) Simple to implement.
2) Hash table never fills up, we can always add more elements to chain.
3) Less sensitive to the hash function or load factors.
4) It is mostly used when it is unknown how many and how frequently keys may be inserted or deleted.

Simple Chaining - Disadvantages:
1) Cache performance of chaining is not good as keys are stored using linked list. Open addressing provides better cache performance as everything is stored in same table.
2) Wastage of Space (Some Parts of hash table are never used)
3) If the chain becomes long, then search time can become O(n) in worst case.
4) Uses extra space for links.

Open Addressing:
Insert(k): Keep probing until an empty slot is found. Once an empty slot is found, insert k.

Search(k): Keep probing until slot's key doesn't become equal to k or an empty slot is reached.

Delete(k): Delete operation is interesting. If we simply delete a key, then search may fail. So slots of deleted keys are marked especially as "deleted".

Open Addressing:
1. Linear Probing
    a. Linearly probe for next slot
2. Quadratic probing:
    a. Look for i^2 slot in ith iteration
3. Double Hashing:
    a. Use another hash function hash2(x) and look for i*hash2(x) slot in ith rotation.

Separate Chaining	**Open Addressing**
Chaining is Simpler to implement.	Open Addressing requires more computation.
In chaining, Hash table never fills up, we can always add more elements to chain.	In open addressing, table may become full.
Chaining is Less sensitive to the hash function or load factors.	Open addressing requires extra care for to avoid clustering and load factor.
Chaining is mostly used when it is unknown how many and how frequently keys may be inserted or deleted.	Open addressing is used when the frequency and number of keys is known.
Cache performance of chaining is not good as keys are stored using linked list.	Open addressing provides better cache performance as everything is stored in the same table.
Wastage of Space (Some Parts of hash table in chaining are never used).	In Open addressing, a slot can be used even if an input doesn't map to it.
Chaining uses extra space for links.	No links in Open addressing

Note: Information gathered in this document has been collected from various sources on Internet.