### **Constructors in HashMap is as follows:**

HashMap provides 4 constructors and the access modifier of each is public which are listed as follows:

1. HashMap()
2. HashMap(int initialCapacity)
3. HashMap(int initialCapacity, float loadFactor)
4. **HashMap(Map map)**

Now discussing above constructors one by one alongside implementing the same with help of clean java programs.

**Constructor 1:** HashMap()

It is the default constructor which creates an instance of HashMap with an initial capacity of 16 and load factor of 0.75.

**Syntax:**

HashMap<K, V> hm = new HashMap<K, V>();

### **Important Features of HashMap**

To access a value one must know its key. HashMap is known as HashMap because it uses a technique called Hashing. [Hashing](https://www.geeksforgeeks.org/hashing-set-1-introduction/) is a technique of converting a large String to small String that represents the same String. A shorter value helps in indexing and faster searches. [HashSet](https://www.geeksforgeeks.org/hashset-in-java/) also uses HashMap internally.

Few important features of HashMap are:

* HashMap is a part of java.util package.
* HashMap extends an abstract class AbstractMap which also provides an incomplete implementation of Map interface.
* It also implements [Cloneable](https://docs.oracle.com/javase/7/docs/api/java/lang/Cloneable.html) and [Serializable](https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html) interface. K and V in the above definition represent Key and Value respectively.
* HashMap doesn’t allow duplicate keys but allows duplicate values. That means A single key can’t contain more than 1 value but more than 1 key can contain a single value.
* HashMap allows null key also but only once and multiple null values.
* This class makes no guarantees as to the order of the map; in particular, it does not guarantee that the order will remain constant over time. It is roughly similar to HashTable but is unsynchronized.

### **Internal Structure of HashMap**

Internally HashMap contains an array of Node and a node is represented as a class that contains 4 fields:

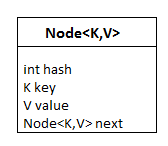
1. int hash
2. K key
3. V value
4. Node next

It can be seen that the node is containing a reference to its own object. So it’s a linked list.

**HashMap:**

array

**Node:**



### **Performance of HashMap**

Performance of HashMap depends on 2 parameters which are named as follows:

1. Initial Capacity
2. Load Factor

**1. Initial Capacity** – It is the capacity of HashMap at the time of its creation (It is the number of buckets a HashMap can hold when the HashMap is instantiated). In java, it is 2^4=16 initially, meaning it can hold 16 key-value pairs.

**2. Load Factor** – It is the percent value of the capacity after which the capacity of Hashmap is to be increased (It is the percentage fill of buckets after which Rehashing takes place). In java, it is 0.75f by default, meaning the rehashing takes place after filling 75% of the capacity.

**3. Threshold** – It is the product of Load Factor and Initial Capacity. In java, by default, it is (16 \* 0.75 = 12). That is, Rehashing takes place after inserting 12 key-value pairs into the HashMap.

**4. Rehashing** – It is the process of doubling the capacity of the HashMap after it reaches its Threshold. In java, HashMap continues to rehash(by default) in the following sequence – 2^4, 2^5, 2^6, 2^7, …. so on.

If the initial capacity is kept higher then rehashing will never be done. But by keeping it higher increases the time complexity of iteration. So it should be chosen very cleverly to increase performance. The expected number of values should be taken into account to set the initial capacity. The most generally preferred load factor value is 0.75 which provides a good deal between time and space costs. The load factor’s value varies between 0 and 1.

***Note:*** *From Java 8 onward, Java has started using Self Balancing BST instead of a linked list for chaining. The advantage of self-balancing bst is, we get the worst case (when every key maps to the same slot) search time is O(Log n).*

### **Synchronized HashMap**

As it is told that HashMap is unsynchronized i.e. multiple threads can access it simultaneously. If multiple threads access this class simultaneously and at least one thread manipulates it structurally then it is necessary to make it synchronized externally. It is done by synchronizing some object which encapsulates the map. If No such object exists then it can be wrapped around Collections.synchronizedMap() to make HashMap synchronized and avoid accidental unsynchronized access. As in the following example:

Map m = Collections.synchronizedMap(new HashMap(...));

Now the Map m is synchronized. Iterators of this class are fail-fast if any structure modification is done after the creation of iterator, in any way except through the iterator’s remove method. In a failure of iterator, it will throw ConcurrentModificationException.

**Time complexity of HashMap:** HashMap provides constant time complexity for basic operations, get and put if the hash function is properly written and it disperses the elements properly among the buckets. Iteration over HashMap depends on the capacity of HashMap and a number of key-value pairs. Basically, it is directly proportional to the capacity + size. Capacity is the number of buckets in HashMap. So it is not a good idea to keep a high number of buckets in HashMap initially.

**Applications of HashMap:** HashMap is mainly the implementation of hashing. It is useful when we need efficient implementation of search, insert and delete operations. Please refer to the [applications of hashing](https://www.geeksforgeeks.org/applications-of-hashing/) for details.

### **Methods in HashMap**

* **K** – The type of the keys in the map.
* **V** – The type of values mapped in the map.

| METHOD | DESCRIPTION |
| --- | --- |
| [clear()](https://www.geeksforgeeks.org/hashmap-clear-method-in-java/#:~:text=HashMap.,mappings%20from%20a%20specified%20HashMap.&text=Parameters%3A%20The%20method%20does%20not,does%20not%20return%20any%20value.) | Removes all of the mappings from this map. |
| [clone()](https://www.geeksforgeeks.org/hashmap-clone-method-in-java/) | Returns a shallow copy of this HashMap instance: the keys and values themselves are not cloned. |
| [compute(K key, BiFunction<? super K,](https://www.geeksforgeeks.org/hashmap-compute-method-in-java-with-examples/) [? super V,? extends V> remappingFunction)](https://www.geeksforgeeks.org/hashmap-compute-method-in-java-with-examples/) | Attempts to compute a mapping for the specified key and its current mapped value (or null if there is no current mapping). |
| [computeIfAbsent(K key, Function<?](https://www.geeksforgeeks.org/hashmap-computeifabsent-method-in-java-with-examples/) [super K,? extends V> mappingFunction)](https://www.geeksforgeeks.org/hashmap-computeifabsent-method-in-java-with-examples/) | If the specified key is not already associated with a value (or is mapped to null), attempts to compute its value using the given mapping function and enters it into this map unless null. |
| [computeIfPresent(K key, BiFunction<? super K,](https://www.geeksforgeeks.org/hashmap-computeifpresentkey-bifunction-method-in-java-with-examples/) [? super V,? extends V> remappingFunction)](https://www.geeksforgeeks.org/hashmap-computeifpresentkey-bifunction-method-in-java-with-examples/) | If the value for the specified key is present and non-null, attempts to compute a new mapping given the key and its current mapped value. |
| [containsKey(Object key)](https://www.geeksforgeeks.org/hashmap-containskey-method-in-java/) | Returns true if this map contains a mapping for the specified key. |
| [containsValue(Object value)](https://www.geeksforgeeks.org/hashmap-containsvalue-method-in-java/) | Returns true if this map maps one or more keys to the specified value. |
| [entrySet()](https://www.geeksforgeeks.org/hashmap-entryset-method-in-java/) | Returns a Set view of the mappings contained in this map. |
| [get(Object key)](https://www.geeksforgeeks.org/hashmap-get-method-in-java/) | Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key. |
| [isEmpty()](https://www.geeksforgeeks.org/hashmap-isempty-method-in-java/) | Returns true if this map contains no key-value mappings. |
| [keySet()](https://www.geeksforgeeks.org/hashmap-keyset-method-in-java/#:~:text=util.,the%20key%20elements%20in%20them.) | Returns a Set view of the keys contained in this map. |
| [merge(K key, V value, BiFunction<? super V,](https://www.geeksforgeeks.org/hashmap-mergekey-value-bifunction-method-in-java-with-examples/#:~:text=The%20merge(Key%2C%20Value%2C,as%20bucket1%20and%20so%20on.) [? super V,? extends V> remappingFunction)](https://www.geeksforgeeks.org/hashmap-mergekey-value-bifunction-method-in-java-with-examples/#:~:text=The%20merge(Key%2C%20Value%2C,as%20bucket1%20and%20so%20on.) | If the specified key is not already associated with a value or is associated with null, associates it with the given non-null value. |
| [put(K key, V value)](https://www.geeksforgeeks.org/hashmap-put-method-in-java/) | Associates the specified value with the specified key in this map. |
| [putAll(Map<? extends K,? extends V> m)](https://www.geeksforgeeks.org/hashmap-putall-method-in-java/#:~:text=HashMap.,from%20one%20map%20into%20another.&text=Parameters%3A%20The%20method%20takes%20one,we%20want%20to%20copy%20from.) | Copies all of the mappings from the specified map to this map. |
| [remove(Object key)](https://www.geeksforgeeks.org/hashmap-remove-method-in-java/) | Removes the mapping for the specified key from this map if present. |
| [size()](https://www.geeksforgeeks.org/hashmap-size-method-in-java/#:~:text=HashMap.,or%20mappings%20in%20the%20Map.&text=Parameters%3A%20The%20method%20does%20not,pairs%20present%20in%20the%20map.) | Returns the number of key-value mappings in this map. |
| [values()](https://www.geeksforgeeks.org/hashmap-values-method-in-java/) | Returns a Collection view of the values contained in this map. |

### **Methods inherited from class java.util.AbstractMap**

| METHOD | DESCRIPTION |
| --- | --- |
| [equals()](https://www.geeksforgeeks.org/abstractmap-equals-method-in-java-with-examples/#:~:text=The%20AbstractMap.,of%20this%20map%20or%20not.) | Compares the specified object with this map for equality. |
| [hashCode()](https://www.geeksforgeeks.org/abstractmap-hashcode-method-in-java-with-examples/) | Returns the hash code value for this map. |
| [toString()](https://www.geeksforgeeks.org/abstractmap-simpleentry-tostring-method-in-java-with-examples/) | Returns a string representation of this map. |

### **Methods inherited from interface java.util.Map**

| METHOD | DESCRIPTION |
| --- | --- |
| [equals()](https://www.geeksforgeeks.org/map-equals-method-in-java-with-examples/) | Compares the specified object with this map for equality. |
| 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 action throws an exception. |
| getOrDefault(Object key, V defaultValue) | Returns the value to which the specified key is mapped, or defaultValue if this map contains no mapping for the key. |
| [hashCode()](https://www.geeksforgeeks.org/map-hashcode-method-in-java-with-examples/) | Returns the hash code value for this map. |
| putIfAbsent(K key, V value) | If the specified key is not already associated with a value (or is mapped to null) associates it with the given value and returns null, else returns the current value. |
| remove(Object key, Object value) | Removes the entry for the specified key only if it is currently mapped to the specified value. |
| replace(K key, V value) | Replaces the entry for the specified key only if it is currently mapped to some value. |
| replace(K key, V oldValue, V newValue) | Replaces the entry for the specified key only if currently mapped to the specified value. |
| replaceAll(BiFunction<? super K,  ? super V,? extends V> function) | Replaces each entry’s value with the result of invoking the given function on that entry until all entries have been processed or the function throws an exception. |