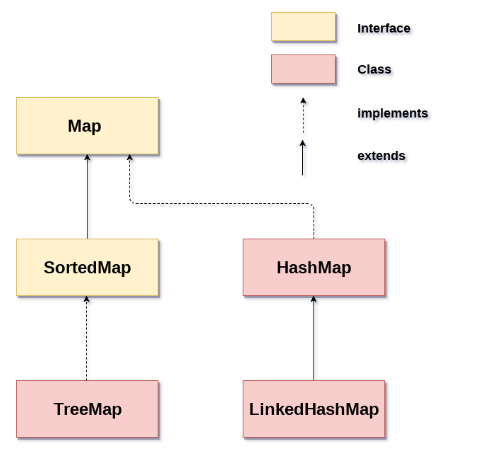
# **Java Map Interface**

A map contains values on the basis of key, i.e. key and value pair. Each key and value pair is known as an entry. A Map contains unique keys.

A Map is useful if you have to search, update or delete elements on the basis of a key.



A Map doesn't allow duplicate keys, but you can have duplicate values. HashMap and LinkedHashMap allow null keys and values, but TreeMap doesn't allow any null key or value.

common scenarios are as follows:

* A map of error codes and their descriptions.
* A map of zip codes and cities.
* A map of managers and employees. Each manager (key) is associated with a list of employees (value) he manages.
* A map of classes and students. Each class (key) is associated with a list of students (value).

|  |  |
| --- | --- |
| Method | Action Performed |
| [clear()](https://www.geeksforgeeks.org/map-clear-method-in-java-with-example/) | This method is used to clear and remove all of the elements or mappings from a specified Map collection. |
| [containsKey(Object)](https://www.geeksforgeeks.org/map-containskey-method-in-java-with-examples/) | This method is used to check whether a particular key is being mapped into the Map or not. It takes the key element as a parameter and returns True if that element is mapped in the map. |
| [containsValue(Object)](https://www.geeksforgeeks.org/map-containsvalue-method-in-java-with-examples/) | This method is used to check whether a particular value is being mapped by a single or more than one key in the Map. It takes the value as a parameter and returns True if that value is mapped by any of the key in the map. |
| [entrySet()](https://www.geeksforgeeks.org/map-entryset-method-in-java-with-examples/) | This method is used to create a set out of the same elements contained in the map. It basically returns a set view of the map or we can create a new set and store the map elements into them. |
| [equals(Object)](https://www.geeksforgeeks.org/map-equals-method-in-java-with-examples/) | This method is used to check for equality between two maps. It verifies whether the elements of one map passed as a parameter is equal to the elements of this map or not. |
| [get(Object)](https://www.geeksforgeeks.org/map-get-method-in-java-with-examples/) | This method is used to retrieve or fetch the value mapped by a particular key mentioned in the parameter. It returns NULL when the map contains no such mapping for the key. |
| [hashCode()](https://www.geeksforgeeks.org/map-hashcode-method-in-java-with-examples/) | This method is used to generate a hashCode for the given map containing keys and values. |
| [isEmpty()](https://www.geeksforgeeks.org/map-isempty-method-in-java-with-examples/) | This method is used to check if a map is having any entry for key and value pairs. If no mapping exists, then this returns true. |
| [keySet()](https://www.geeksforgeeks.org/map-keyset-method-in-java-with-examples/) | This method is used to return a Set view of the keys contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. |
| [put(Object, Object)](https://www.geeksforgeeks.org/map-put-method-in-java-with-examples/) | This method is used to associate the specified value with the specified key in this map. |
| [putAll(Map)](https://www.geeksforgeeks.org/map-putall-method-in-java-with-examples/) | This method is used to copy all of the mappings from the specified map to this map. |
| [remove(Object)](https://www.geeksforgeeks.org/map-remove-method-in-java-with-examples/) | This method is used to remove the mapping for a key from this map if it is present in the map. |
| [size()](https://www.geeksforgeeks.org/hashmap-size-method-in-java/) | This method is used to return the number of key/value pairs available in the map. |
| [values()](https://www.geeksforgeeks.org/hashmap-values-method-in-java/) | This method is used to create a collection out of the values of the map. It basically returns a Collection view of the values in the HashMap. |
| [getOrDefault(Object key, V defaultValue)](https://www.geeksforgeeks.org/hashmap-getordefaultkey-defaultvalue-method-in-java-with-examples/) | Returns the value to which the specified key is mapped, or defaultValue if this map contains no mapping for the key. |
| [merge(K key, V value, BiFunction<? super V,? super V,? extends V> remappingFunction)](https://www.geeksforgeeks.org/hashmap-mergekey-value-bifunction-method-in-java-with-examples/) | If the specified key is not already associated with a value or is associated with null, associates it with the given non-null value. |
| [putIfAbsent(K key, V value)](https://www.geeksforgeeks.org/hashmap-putifabsentkey-value-method-in-java-with-examples/) | 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 curassociaterent value. |

## **Map.Entry** Interface

Entry is the subinterface of Map. we will be accessed it by Map.Entry name. It returns a collection-view of the map, whose elements are of this class. It provides methods to get key and value.

### Methods of Map.Entry interface

|  |  |
| --- | --- |
| **Method** | **Description** |
| K getKey() | It is used to obtain a key. |
| V getValue() | It is used to obtain value. |
| int hashCode() | It is used to obtain hashCode. |
| V setValue(V value) | It is used to replace the value corresponding to this entry with the specified value. |
| boolean equals(Object o) | It is used to compare the specified object with the other existing objects. |
| static <K extends Comparable<? super K>,V> Comparator<Map.Entry<K,V>> comparingByKey() | It returns a comparator that compare the objects in natural order on key. |
| static <K,V> Comparator<Map.Entry<K,V>> comparingByKey(Comparator<? super K> cmp) | It returns a comparator that compare the objects by key using the given Comparator. |
| static <K,V extends Comparable<? super V>> Comparator<Map.Entry<K,V>> comparingByValue() | It returns a comparator that compare the objects in natural order on value. |
| static <K,V> Comparator<Map.Entry<K,V>> comparingByValue(Comparator<? super V> cmp) | It returns a comparator that compare the objects by value using the given Comparator. |

|  |
| --- |
| **import** java.util.\*;    **class** MapExample {  **public** **static** **void** main(String args[])  {  // Creating an empty HashMap  Map<String, Integer> hm = **new** HashMap<String, Integer>();    // Inserting pairs in above Map  hm.put("a", **new** Integer(100));  hm.put("b", **new** Integer(200));  hm.put("c", **new** Integer(300));  hm.put("d", **new** Integer(400));    // Traversing through Map using for-each loop  **for** (Map.Entry<String, Integer> me :hm.entrySet())  {  System.out.print(me.getKey() + ":");  System.out.println(me.getValue());  }  }  } |

***HashMap***

* It provides the basic implementation of the Map interface of Java.
* It stores the data in (Key, Value) pairs.
* To access a value one must know its key.
* This class uses a technique called [Hashing](https://www.geeksforgeeks.org/hashing-data-structure/).
* A shorter value helps in indexing and faster searches.

|  |
| --- |
| **import** java.util.\*;  **public** **class** HashMapExample  {  **public** **static** **void** main(String[] args)  {  Map<String, Integer> map = **new** HashMap<>();    map.put("vishal", 10);  map.put("sachin", 30);  map.put("vaibhav", 20);    // Iterating over Map  **for** (Map.Entry<String, Integer> e : map.entrySet())    System.out.println(e.getKey() + " "+ e.getValue());  }  } |

### ***Operations using Map Interface and HashMap Class***

* **Adding Elements**

hm1.put(1, "Geeks");

hm2.put(**new** Integer(1), "Geeks");

* **Changing Element**

hm1.put(**new** Integer(2), "For");

* **Removing Elements**

hm1.remove(**new** Integer(4));

* **Iterating through the Map**

**for** (Map.Entry<String, Integer> e : map.entrySet())

System.out.println(e.getKey() + " "+ e.getValue());

***LinkedHashMap***

HashMap provided the advantage of quick insertion, search, and deletion but it never maintained the track and order of insertion which the LinkedHashMap provides where the elements can be accessed in their insertion order.

|  |
| --- |
| **public** **class** GFG {  **public** **static** **void** main(String[] args)  {  Map<String, Integer> map = **new** LinkedHashMap<>();    map.put("vishal", 10);  map.put("sachin", 30);  map.put("vaibhav", 20);    **for** (Map.Entry<String, Integer> e : map.entrySet())  System.out.println(e.getKey() + " " + e.getValue());  }  } |