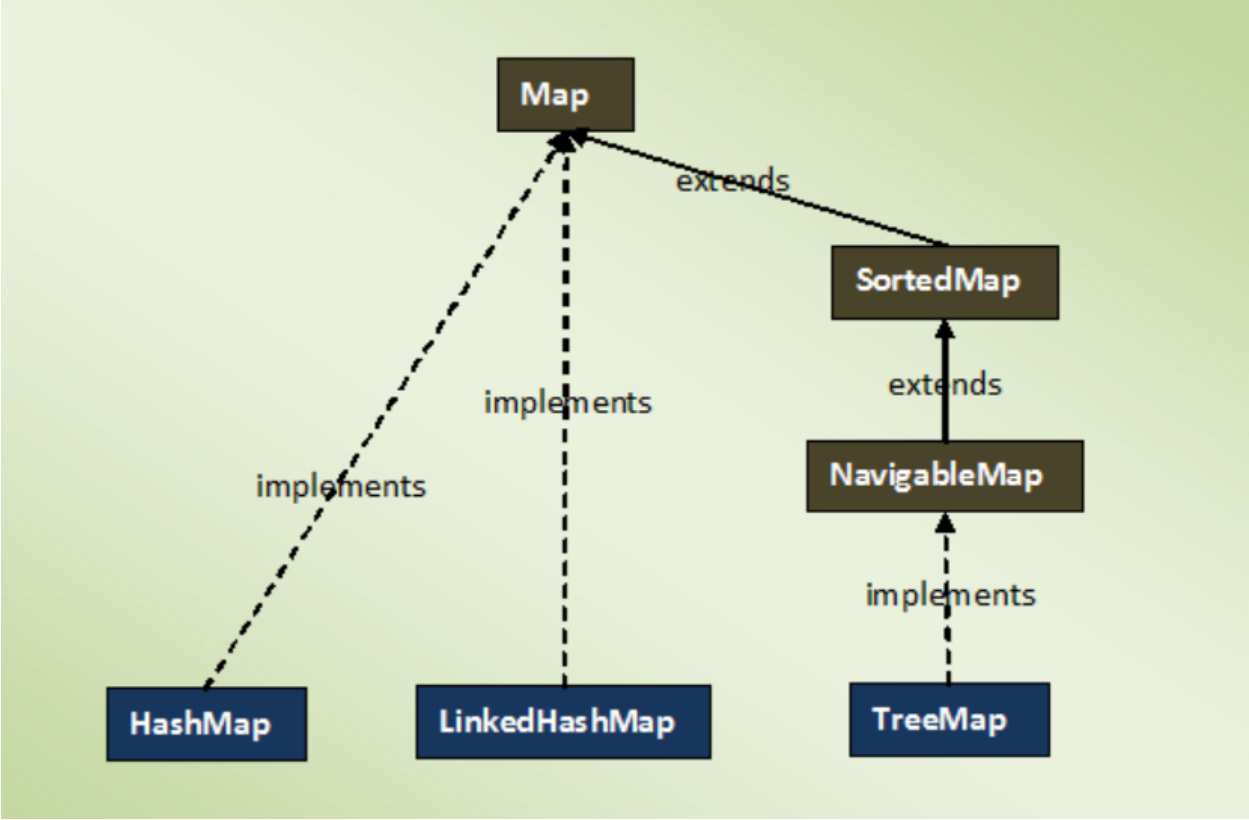
# Map

## Overview

- Map is an interface
- Map is **not a child interface of Collection**.
- It represents a group of Key-Value pairs.
- Both Keys and Values are Objects Only.
- Duplicate Keys are Not allowed but duplicate values are allowed.
- Each Key- Value Pair is Called an Entry.




## Methods

| Sr. No. | Method | Description |
|---|---|---|
| 1 | V put(Object key, Object value) | It is used to insert an entry in the map. |
| 2 | void putAll(Map map) | It is used to insert the specified map in the map. |

| | | |
|---|---|---|
| 3 | V putIfAbsent(K key, V value) | It inserts the specified value with the specified key in the map only if it is not already specified. |
| 4 | V remove(Object key) | It is used to delete an entry for the specified key. |
| 5 | boolean remove(Object key, Object value) | It removes the specified values with the associated specified keys from the map. |
| 6 | Set keySet() | It returns the Set view containing all the keys. |
| 7 | Set<Map.Entry<K,V>> entrySet() | It returns the Set view containing all the keys and values. |
| 8 | void clear() | It is used to reset the map. |
| 9 | V compute(K key, BiFunction<? super K,? super V,? extends V> remappingFunction) | It is used to compute a mapping for the specified key and its current mapped value (or null if there is no current mapping). |
| 10 | V computeIfAbsent(K key, Function<? super K,? extends V> mappingFunction) | It is used to compute its value using the given mapping function, if the specified key is not already associated with a value (or is mapped to null), and enters it into this map unless null. |
| 11 | V computeIfPresent(K key, BiFunction<? super K,? super V,? extends V> remappingFunction) | It is used to compute a new mapping given the key and its current mapped value if the value for the specified key is present and non-null. |
| 12 | boolean containsValue(Object value) | This method returns true if some value equal to the value exists within the map, else return false. |
| 13 | boolean containsKey(Object key) | This method returns true if some key equal to the key exists within the map, else return false. |
| 14 | boolean equals(Object o) | It is used to compare the specified Object with the Map. |
| 15 | void forEach(BiConsumer<? super K,? super V> action) | It performs the given action for each entry in the map until all entries have been processed or the action throws an exception. |
| 16 | V get(Object key) | This method returns the object that contains the value associated with the key. |
| 17 | V getOrDefault(Object key, V defaultValue) | It returns the value to which the specified key is mapped, or defaultValue if the map contains no mapping for the key. |
| 18 | int hashCode() | It returns the hash code value for the Map |
| 19 | boolean isEmpty() | This method returns true if the map is empty; returns false if it contains at least one key. |
| 20 | V merge(K key, V value, BiFunction<? super V,? super V,? extends V> remappingFunction) | If the specified key is not already associated with a value or is associated with null, associates it with the given non-null value. |
| 21 | V replace(K key, V value) | It replaces the specified value for a specified key. |

| | | |
|---|---|---|
| 22 | boolean replace(K key, V oldValue, V newValue) | It replaces the old value with the new value for a specified key. |
| 23 | void replaceAll(BiFunction <? super K,? super V,? extends V> function) | It 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. |
| 24 | Collection values() | It returns a collection view of the values contained in the map. |
| 25 | int size() | This method returns the number of entries in the map. |

**Note**
To traverse Map we need **keySet** or **entrySet** method or using **iterator**.

eg., of keySet

```
Map<Integer, Integer> mp = new HashMap<>();
mp.put(1, 1276);
mp.put(2, 783);
mp.put(1, 99);

for(Integer i : mp.keySet())
{
        System.out.println(i);
}
```

eg., of entrySet

```
Map<Integer, Integer> mp = new HashMap<>();
mp.put(1, 1276);
mp.put(2, 783);
mp.put(1, 99);

for (Map.Entry<Integer , Integer> entry : mp.entrySet())
{
        System.out.println();
        System.out.println(entry.getKey() +" "+ entry.getValue());
}
```

eg., using **iterator**

```
Map<Integer, Integer> mp = new HashMap<>();
mp.put(1, 1276);
mp.put(2, 783);
```

```
mp.put(1, 99);

Set<Entry<Integer, Integer>> entries = mp.entrySet();
Iterator<Entry<Integer, Integer>> itr = entries.iterator();

while(itr.hasNext())
{
        Entry<Integer, Integer> e = itr.next();
        e.getKey();
        e.getValue();
}
```

eg., using **foreach** loop

```
Map<Integer, Integer> mp = new HashMap<>();
mp.put(1, 1276);
mp.put(2, 783);
mp.put(1, 99);

Set<Entry<Integer, Integer>> entries = mp.entrySet();

for (Entry<Integer, Integer> entry : entries)
{
        entry.getKey();
        entry.getValue();
}
```

## Map.Entry Interface

- Without an existing map, entry can not exist hence entry is a subinterface of Map.

**Method**

| Sr. No. | Method | Description |
|---|---|---|
| 1 | K getKey() | It is used to obtain a key. |
| 2 | V getValue() | It is used to obtain value. |
| 3 | int hashCode() | It is used to obtain hashCode. |
| 4 | V setValue(V value) | It is used to replace the value corresponding to this entry with the specified value. |

| | | |
|---|---|---|
| 5 | boolean equals(Object o) | It is used to compare the specified object with the other existing objects. |
| 6 | 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. |
| 7 | 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. |
| 8 | 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. |
| 9 | 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. |

**Sort Map using stream API**

```
Map<Integer, Integer> mp = new HashMap<>();
mp.put(1, 1276);
mp.put(2, 783);
mp.put(5, 999);

mp.entrySet().stream().sorted(Map.Entry.comparingByKey()).forEach(System.out::println);
System.out.println("******************");

mp.entrySet().stream().sorted(Map.Entry.comparingByKey(Comparator.reverseOrder())).forEach
(System.out::println);
System.out.println("******************");

mp.entrySet().stream().sorted(Map.Entry.comparingByValue()).forEach(System.out::println);
System.out.println("******************");

mp.entrySet().stream().sorted(Map.Entry.comparingByValue(Comparator.reverseOrder())).forEa
ch(System.out::println);
```