**MAP :( PART of collection)**

* Pair of object
* Key-value pair of object
* Key can be any object, should be unique
* Value can be any type of object
* Value is mapped by key
* Value is retrieved based on its key
* Each pair is known as an entry.

**Map (I)**

**Sorted Map**

**Hash Table**

**Hash Map**

**Navigable Map**

**LinkedHashMap**

**Tree Map**

Commonly used methods of Map interface:

1. **public Object put(Object key, Object value):** is used to insert an entry in this map.
2. **public void putAll(Map map):** is used to insert the specified map in this map.
3. **public Object remove(Object key):** is used to delete an entry for the specified key.
4. **public Object get(Object key):** is used to return the value for the specified key.
5. **public boolean containsKey(Object key):** is used to search the specified key from this map.
6. **public boolean containsValue(Object value):** is used to search the specified value from this map.
7. **public Set keySet():** returns the Set view containing all the keys.
8. **public Set entrySet():** returns the Set view containing all the keys and values.

Entry

Entry is the sub interface of Map. So we will be accessed it by Map.Entry name. It provides methods to get key and value.

Methods of Entry interface:

1. **public Object getKey():** is used to obtain key.
2. **public Object getValue():**is used to obtain value.

# **HashMap**

* A HashMap contains values based on the key. It implements the Map interface and extends AbstractMap class.
* It contains only unique elements.
* It may have one null key and multiple null values.
* It maintains no order.

### What is difference between HashSet and HashMap?

HashSet contains only values whereas HashMap contains entry(key and value).

# **LinkedHashMap**

* A LinkedHashMap contains values based on the key. It implements the Map interface and extends HashMap class.
* It contains only unique elements.
* It may have one null key and multiple null values.
* It is same as HashMap instead maintains insertion order.

# [**TreeMap**](http://www.javatpoint.com/TreeMap-class-in-collection-framework)

* A [TreeMap](http://www.javatpoint.com/TreeMap-class-in-collection-framework) contains values based on the key. It implements the NavigableMap interface and extends AbstractMap [class](http://www.javatpoint.com/TreeMap-class-in-collection-framework).
* It contains only unique elements.
* It cannot have null key but can have multiple null values.
* It is same as HashMap instead maintains ascending order.

What is difference between HashMap and [TreeMap](http://www.javatpoint.com/TreeMap-class-in-collection-framework)?

|  |  |
| --- | --- |
| 1) HashMap is can contain one null key. | [TreeMap](http://www.javatpoint.com/TreeMap-class-in-collection-framework) can not contain any null key. |
| 2) HashMap maintains no order. | [TreeMap](http://www.javatpoint.com/TreeMap-class-in-collection-framework) maintains ascending order. |

# **Hashtable**

* A Hashtable is an array of list.Each list is known as a bucket.The position of bucket is identified by calling the hashcode() method.A Hashtable contains values based on the key. It implements the Map interface and extends Dictionary class.
* It contains only unique elements.
* It may have not have any null key or value.
* It is synchronized.

**Difference between three Maps:**

|  |  |  |
| --- | --- | --- |
| Hash Map | Hash Table | Tree Map |
| 1. Null can be key 2. Method are not synchronized 3. Not a thread safe 4. Unsorted map 5. Performance high | 1. Null can’t be key 2. Method are synchronized 3. Thread safe class 4. Unsorted map 5. Performance low | 1. Null can’t be key 2. Method are not synchronized 3. Not a thread safe 4. Sorted map 5. Performance high |