JAVA - THE MAP INTERFACE

The Map interface maps unique keys to values. A key is an object that you use to retrieve a value at a later date.

- Given a key and a value, you can store the value in a Map object. After the value is stored, you can retrieve it by using its key.
- Several methods throw a NoSuchElementException when no items exist in the invoking map.
- A ClassCastException is thrown when an object is incompatible with the elements in a map.
- A NullPointerException is thrown if an attempt is made to use a null object and null is not allowed in the map.
- An UnsupportedOperationException is thrown when an attempt is made to change an unmodifiable map.

SN	Methods with Description
1	**void clear( )** Removes all key/value pairs from the invoking map.
2	**boolean containsKey(Object k)** Returns true if the invoking map contains k as a key. Otherwise, returns false.
3	**boolean containsValue(Object v)** Returns true if the map contains v as a value. Otherwise, returns false
4	**Set entrySet( )** Returns a Set that contains the entries in the map. The set contains objects of type Map.Entry. This method provides a set-view of the invoking map.
5	**boolean equals(Object obj)** Returns true if obj is a Map and contains the same entries. Otherwise, returns false.
6	**Object get(Object k)** Returns the value associated with the key k.
7	**int hashCode( )** Returns the hash code for the invoking map.
8	**boolean isEmpty( )** Returns true if the invoking map is empty. Otherwise, returns false.
9	**Set keySet( )** Returns a Set that contains the keys in the invoking map. This method provides a set-view

of the keys in the invoking map.

10 **Object put(Object k, Object v)**

Puts an entry in the invoking map, overwriting any previous value associated with the key. The key and value are k and v, respectively. Returns null if the key did not already exist. Otherwise, the previous value linked to the key is returned.

11 **void putAll(Map m)**

Puts all the entries from m into this map.

12 **Object remove(Object k)**

Removes the entry whose key equals k.

13 **int size( )**

Returns the number of key/value pairs in the map.

14 **Collection values( )**

Returns a collection containing the values in the map. This method provides a collection-view of the values in the map.

Example:

Map has its implementation in various classes like HashMap. Following is the example to explain map functionality:

```
import java.util.*;

public class CollectionsDemo {

   public static void main(String[] args) {
      Map m1 = new HashMap();
      m1.put("Zara", "8");
      m1.put("Mahnaz", "31");
      m1.put("Ayan", "12");
      m1.put("Daisy", "14");
      System.out.println();
      System.out.println(" Map Elements");
      System.out.print("\t" + m1);
   }
}
```

This would produce the following result:

```
Map Elements
        {Mahnaz=31, Ayan=12, Daisy=14, Zara=8}
```