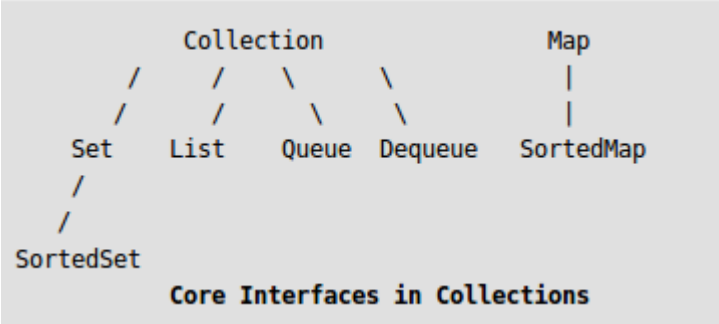
Map interface in Java with examples

The java.util.Map interface represents a mapping between a key and a value. The Map interface is not a subtype of the Collection interface. Therefore it behaves a bit different from the rest of the collection types.



Core Interfaces in Collections

A Map cannot contain duplicate keys and each key can map to at most one value. Some implementations allow null key and null value (HashMap and LinkedHashMap) but some do not (TreeMap).

The order of a map depends on specific implementations, e.g TreeMap and LinkedHashMap have predictable order, while HashMap does not.
Exampled class that implements this interface is HashMap, TreeMap and LinkedHashMap.

Why and **When** Use Maps:
Maps are perfectly for key-value association mapping such as dictionaries. Use Maps when you want to retrieve and update elements by keys, or perform lookups by keys. Some examples:

- 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).

Methods of Map:

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 Set keySet() :- returns the Set view containing all the keys.
7. public Set entrySet() :- returns the Set view containing all the keys and values.

```
// Java program to demonstrate working of Map interface
import java.util.*;
class HashMapDemo
{
   public static void main(String args[])
   {
       HashMap< String,Integer> hm =
                           new HashMap< String,Integer>();
       hm.put("a", new Integer(100));
       hm.put("b", new Integer(200));
       hm.put("c", new Integer(300));
       hm.put("d", new Integer(400));

       // Returns Set view
       Set< Map.Entry< String,Integer> > st = hm.entrySet();

       for (Map.Entry< String,Integer> me:st)
       {
           System.out.print(me.getKey()+":");
           System.out.println(me.getValue());
       }
   }
}
```

Run on IDE

Output:

```
a:100
b:200
c:300
d:400
```

This article is contributed by **Pratik Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.