HashMap in Java

HashMap is a part of Java's collection since Java 1.2. It provides the basic implementation of the Map interface of Java. It stores the data in (Key, Value) pairs. Unlike HashSet which stored only the key, a HashMap stores both key-value pairs.
Performance of HashMap depends on 2 parameters:

- Initial Capacity
- Load Factor

Few important features of HashMap are:

- HashMap is a part of java.util package.
- HashMap extends an abstract class AbstractMap which also provides an incomplete implementation of Map interface.
- HashMap doesn't allow duplicate keys but allows duplicate values. That means A single key can't contain more than 1 value but more than 1 key can contain a single value.
- Does not maintains the track and order of insertion of elements.

Important Methods in HashMap and there working.

Method	Description	Time Complexity
put(Object key, Object value)	It is used to insert a particular mapping of key-value pair into a map.	O(1) on average
size()	It is used to return the size of a map.	O(1) on worst case
isEmpty()	Used to check whether the map is empty or not. Returns true if the map is empty.	O(1) on worst case

Dash
All
Articles
Videos
Problems
Quiz
Contest

Example 1: Working of put(), size() and Traversal of HashMap.

java

```
// Java program to illustrate
// Java.util.HashMap

import java.util.HashMap;
import java.util.Map;

public class GFG {
    public static void main(String[] args)
    {
        // Create an empty hash map
        HashMap<String, Integer> m
            = new HashMap<>();
```

```
        // Add elements to the map
        m.put("gfg", 10);
        m.put("ide", 15);
        m.put("courses", 20);

        // Print size and content
        System.out.println(m);
        System.out.println(m.size());

        // Iterating over HashMap
        for(Map.Entry<String, Integer>e : m.entrySet())
            System.out.println(e.getKey() + " " + e.getValue());
    }
}
```

Output:

```
{courses=20, gfg=10, ide=15}
3
courses 20
gfg 10
ide 15
```

Note: The order of output in key-value pairs varies.

Method	Description	Time Complexity
containsKey(Object key)	Used to return True if for a specified key, mapping is present in the map.	O(1) on average
remove(Object key)	It is used to remove the a particular key in the Map and returns the corresponding value. If the key is not present then the function returns Null.	O(1) on average

Example 2: Working of containsKey(), remove().

java

```
// Java program to illustrate
// Java.util.HashMap

import java.util.HashMap;
import java.util.Map;

public class GFG {
    public static void main(String[] args)
    {
        // Create an empty hash map
```

```
        HashMap<String, Integer> m
            = new HashMap<>();

        // Add elements to the map
        m.put("gfg", 10);
        m.put("ide", 15);
        m.put("courses", 20);

        // Check for a key
        if (m.containsKey("ide"))
            System.out.println("Yes");
        else
            System.out.println("No");

        // Remove key "ide"
        // and returns the associated value 15
        m.remove("ide");
        System.out.println(m.size());
    }
}
```

Output:

```
Yes
2
```

Method	Description	Time Complexity

containsValue(Object value)	**Used to return true if one or more key is mapped to a specified value.**	**O(n)**
get(Object key)	**It is used to retrieve or fetch the value mapped by a particular key. If the key is not found then it returns Null.**	**O(1) on average**

Example 3: Working of containsValue() and get().

java

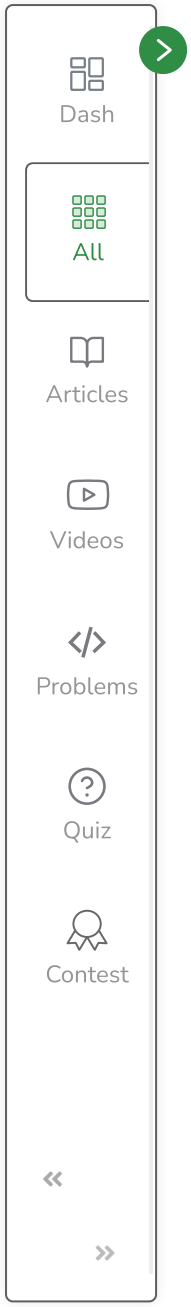
```
// Java program to illustrate
// Java.util.HashMap

import java.util.HashMap;
import java.util.Map;

public class GFG {

        // Create an empty hash map
        HashMap<String, Integer> m
            = new HashMap<>();

        // Add elements to the map
        m.put("gfg", 10);
        m.put("ide", 15);
        m.put("courses", 20);
```

```
        // Check for a Value
        if (m.containsValue(15))
            System.out.println("Yes");
        else
            System.out.println("No");

        // Get value corresponding to passed key
        // <"ide", 15>
        System.out.println(m.get("ide"));

        // The given key is absent
        System.out.println(m.get("practice"));

    }
}
```

Output:

```
Yes
15
null
```

Mark as Read



Report An Issue

If you are facing any issue on this page. Please let us know.