


Factory method to create immutable Map in Java 9
Polymorphism in Java
Maximum absolute difference in an array
What's the connection between Java and Blockchain?
Difference between static and non-static method in Java
Spring Dependency Injection with Example
Count occurrences of a given character using Regex in Java
Difference between static and non-static variables in Java
File Handling in Java with CRUD operations
What is the Role of Java in the IT Industry?
BigInteger multiply() Method in Java with Examples
Difference between concat() and + operator in Java
Difference between String and Character array in Java
Recursion in Java
Different Approaches to Concurrent Programming in Java
Java program to check whether a string is a Palindrome
How to create a Java HashMap of user defined class type?
Differences between Interface and Class in Java
How to read a Matrix from user in Java?
BigInteger add() Method in Java with Examples
Creating an Asynchronous Multithreaded chat Application in Java
Implementing next_permutation() in Java with Examples
Difference between while and do-while loop in C, C++, Java
Aspect Oriented Programming and AOP in Spring Framework
ClassLoader in Java
Different ways for String to Integer Conversions in Java
Count the pairs of vowels in the given string
How to Clone a List in Java?
How to overload and override main method in Java



### Absolutely free from Baddies

0% added perfume, 0% colour, 0% harsh chemicals. Gentle on even the most sensitive skin.

➔

Classroom program in NIOSA Starting from 21<sup>st</sup> September 2019

## Machine Learning FOUNDATION WITH PYTHON

₹10,999

Register Now

2<sup>nd</sup> Batch

✓ Mentored by Industry Expert

Works in ADOR& AI, ANIMOL, ANIMOL

Good for foundations Machine Learning - Rohit, Ray

The course was excellent! All basic topics of ML are covered

ANIMOL, SHARMA

## WeakHashMap class in Java

WeakHashMap is the Hash table based implementation of the Map interface, with weak keys. An entry in a WeakHashMap will automatically be removed when its key is no longer in ordinary use. More precisely, the presence of a mapping for a given key will not prevent the key from being discarded by the garbage collector, that is, made finalizable, finalized, and then reclaimed. When a key has been discarded its entry is effectively removed from the map, so this class behaves somewhat differently from other Map implementations. Few important features of a weak hashmap are:

- Both null values and null keys are supported in WeakHashMap.
- It is not synchronised.
- This class is intended primarily for use with key objects whose equals methods test for object identity using the == operator.

### Constructors in WeakHashMap:

- WeakHashMap()**: This constructor is used to create an empty WeakHashMap with the default initial capacity(16) and load factor (0.75).
- WeakHashMap(int initialCapacity)**: This constructor is used to create an empty WeakHashMap with the given initial capacity and the default load factor (0.75).
- WeakHashMap(int initialCapacity, float loadFactor)**: This constructor is used to create an empty WeakHashMap with the given initial capacity and the specified load factor.

### Methods in WeakHashMap:

- void clear()**: The method removes all of the mappings from this map. The map will be empty after this call returns.

**Syntax:** public void clear().  
**Returns:** NA.  
**Exception:** NA.
- boolean containsValue(Object value)**: This method returns true if this map maps one or more keys to the specified value.

**Syntax:** public boolean containsValue(Object value).  
**Returns:** true if this map maps one or more keys to the specified value.  
**Exception:** NA
- boolean containsKey(Object key)**: This method returns true if this map contains a mapping for the specified key.

**Syntax:** public boolean containsKey(Object key).  
**Returns:** true if there is a mapping for key; false otherwise.  
**Exception:** NA
- put(K key, V value)**: Associates the specified value with the specified key in this map. If the map previously contained a mapping for this key, the old value is replaced.

**Syntax:** public put(K key, V value).  
**Returns:** the previous value associated with key, or null if there was no mapping for key. (A null return can also indicate that the map previously associated null with key.)  
**Exception:** NA
- boolean isEmpty()**: Returns true if this map contains no key-value mappings. This result is a snapshot, and may not reflect unprocessed entries that will be removed before next attempted access because they are no longer referenced.

**Syntax:** public boolean isEmpty()  
**Returns:** true if this map contains no key-value mappings.  
**Exceptions:** NA

```
// Java code illustrating clear(), containsValue()
// containsKey() and isEmpty() method
import java.util.Map;
import java.util.WeakHashMap;

class WeakHashMapdemo
{
    public static void main(String[] arg)
    {
        Map<Number, String> weak = new WeakHashMap<Number, String>();
        weak.put(1, "geeks");
        weak.put(2, "for");
        weak.put(3, "geeks");

        // Checking weak map
        System.out.println("our weak map: " + weak);

        // Checking if "for" exist
        if(weak.containsValue("for"))
            System.out.println("for exist");

        // Checking if 1 exist as a key in map
        if(weak.containsKey(1))
            System.out.println("1 exist");

        // Removing all data
        weak.clear();

        // Checking whether map is empty or not
        if(weak.isEmpty())
            System.out.println("empty map: " + weak);
    }
}
```

**Output:**

```
our weak map: {3=geeks, 2=for, 1=geeks}
for exist
1 exist
empty map: {}
```

- Set entrySet()**: Returns a Set view of the mappings contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setView operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the iterator.remove, Set.remove, removeAll, retainAll and clear operations. It does not support the add or addAll operations.

**Syntax:** public Set entrySet()  
**Returns:** a set view of the mappings contained in this map.  
**Exception:** NA

- Set keySet()**: Returns a Set view of the keys contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation), the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the iterator.remove, Set.remove, removeAll, retainAll, and clear operations. It does not support the add or addAll operations.

**Syntax:** public Set keySet().  
**Returns:** a set view of the keys contained in this map  
**Exception:** NA

- Collection values()**: Returns a Collection view of the values contained in this map. The collection is backed by the map, so changes to the map are reflected in the collection, and vice-versa. If the map is modified while an iteration over the collection is in progress (except through the iterator's own remove operation), the results of the iteration are undefined. The collection supports element removal, which removes the corresponding mapping from the map, via the iterator.remove, Collection.remove, removeAll, retainAll and clear operations. It does not support the add or addAll operations.

```
// Java code illustrating entrySet(), keySet() and Values()
import java.util.Collection;
import java.util.Map;
import java.util.Set;
import java.util.WeakHashMap;

class WeakHashMapdemo
{
    public static void main(String[] arg)
    {
        Map<Number, String> weak = new WeakHashMap<Number, String>();
        weak.put(1, "geeks");
        weak.put(2, "for");
        weak.put(3, "geeks");

        Set set1 = weak.entrySet();

        // Checking set
        System.out.println(set1);

        // Creating set for key
        Set keySet = weak.keySet();

        // Checking keySet
        System.out.println("key set: " + keySet );

        Collection value = weak.values();

        // Checking values of map
        System.out.println("values: " + value);
    }
}
```

**Output:**

```
[3=geeks, 2=for, 1=geeks]
key set: [3, 2, 1]
values: [geeks, for, geeks]
```

- void putAll(Map m)**: Copies all of the mappings from the specified map to this map. These mappings will replace any mappings that this map had for any of the keys currently in the specified map.

**Syntax:** public void putAll(Map m)  
**Returns:** NA  
**Exception:** NullPointerException - if the specified map is null.

- V get(Object key)**: Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.

**Syntax:** public V get(Object key)  
**Returns:** the value to which the specified key is mapped, or null if this map contains no mapping for the key  
**Exception:** NA

- V remove(Object key)**: Removes the mapping for a key from this weak hash map if it is present. More formally, if this map contains a mapping from key k to value v such that (key==null ? k==null : key.equals(k)), that mapping is removed.

**Syntax:** public V remove(Object key)  
**Returns:** the previous value associated with key, or null if there was no mapping for key  
**Exception:** NA.

- int size()**: Returns the number of key-value mappings in this map. This result is a snapshot, and may not reflect unprocessed entries that will be removed before next attempted access because they are no longer referenced.

**Syntax:** public int size()  
**Returns:** the number of key-value mappings in this map.  
**Exception:** NA

```
// Java code remove(), putAll()
// get() and size() method
import java.util.Collection;
import java.util.Map;
import java.util.Set;
import java.util.WeakHashMap;

class WeakHashMapdemo
{
    public static void main(String[] arg)
    {
        Map<Number, String> weak = new WeakHashMap<Number, String>();
        weak.put(1, "geeks");
        weak.put(2, "for");
        weak.put(3, "geeks");

        Map<Number, String> weak1 = new WeakHashMap<Number, String>();
        weak1.putAll(weak);

        // getting value of key 2
        System.out.println(weak1.get(2));

        // size of map
        System.out.println("Size of map is: " + weak1.size());

        // removing 2nd element
        weak1.remove(2);

        // size after removing key and value pair
        System.out.println("Size after removing: " + weak1.size());
    }
}
```

**Output:**

```
for
Size of map is: 3
Size after removing: 2
```

This article is contributed by **Abhishek Verma**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](https://contribute.geeksforgeeks.org) or mail your article to [contribute@geeksforgeeks.org](mailto: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.

### Recommended Posts:

- WeakHashMap get() Method in Java
- WeakHashMap put() Method in Java
- HashMap vs WeakHashMap in Java
- WeakHashMap clear() Method in Java
- WeakHashMap keySet() method in Java
- WeakHashMap containsValue() Method in Java
- WeakHashMap size() method in Java
- WeakHashMap remove() method in Java
- WeakHashMap values() Method in Java
- WeakHashMap entrySet() Method in Java
- WeakHashMap containsKey() Method in Java
- WeakHashMap putAll() Method in Java
- WeakHashMap isEmpty() Method in Java with Examples
- Implement Septet Class from Quintet Class in Java using JavaTuples
- Implement Septet Class from Sextet Class in Java using JavaTuples

Article Tags : [Java](#) [Java - util package](#) [Java-HashMap](#)

Practice Tags : [Java](#)

👍

2

☐ To-do

☐ Done

0

No votes yet.

Feedback/ Suggest Improvement

Add Notes


Improve Article

Please write to us at [contribute@geeksforgeeks.org](mailto:contribute@geeksforgeeks.org) to report any issue with the above content.

Previous [For Loop in Java](#) | Important points [Java.io.PushbackInputStream class in Java](#) Next

Java

Writing code in comment? Please use [ide.geeksforgeeks.org](https://ide.geeksforgeeks.org), generate link and share the link here.




Boost your career with Simplilearn's

## Artificial Intelligence Engineer

Master's Program and Job Assist.

₹44,999

GET STARTED



1 in every 15 CS STUDENTS OPTED FOR

DSA

SELF-PACED COURSE

WHAT ARE YOU WAITING FOR?

₹2,499

Register Now

2000+ ENROLLMENTS SO FAR

Most popular in Java
Print characters and their frequencies in order of occurrence using a Linked-HashMap in Java
Difference between the Constructors and Methods
Lambda Expressions   Concurrent Programming Approach 4
Difference between an Iterator and ListIterator in Java
NumberFormat.getCurrencyInstance() method in Java with Examples





LENOVO V130 (81HN00FRH) (Core i3-7020U ...

★★★★★ (1)

Rs. 29,000.00 (details + delivery)

More related articles in Java
Count occurrence of a given character in a string using Stream API in Java
How to Compile and Run C/C++/Java Programs in Linux
Count of words ending at the given suffix in Java
Android   How to Request permissions in Android Application?
Integer.valueOf() vs Integer.parseInt() with Examples

Starting from 1<sup>st</sup> September 2019

SDE

TEST SERIES

₹599

₹199

Register Now

ONE TIME OFFER

Test Series Features

5 Contest

Video Solution

Job Opportunities