

| |
|---|
| LinkedList in Java |
| LinkedList clear() Method in Java |
| LinkedList containsKey() Method in Java |
| LinkedList get() Method in Java |
| LinkedList removeEldestEntry() Method in Java |
| LinkedList and LinkedHashSet in Java |
| Design a data structure for LRU Cache |
| Hashing in Java |
| Difference between Soft Computing and Hard Computing |
| Difference between pointer to an array and array of pointers |
| Differences between Procedural and Object Oriented Programming |
| Difference between Process and Thread |
| Difference between Virtual memory and Cache memory |
| Difference Between Programming, Scripting, and Markup Languages |
| Difference between static and non-static variables in Java |
| Greedy approach vs Dynamic programming |
| Difference between concat() and + operator in Java |
| Difference between Primary and Candidate Key |
| Difference between String and Character array in Java |
| Difference between Super Key and Candidate Key |
| Difference between Compile Time Errors and Runtime Errors |
| Difference Between BFS and DFS |
| Difference between regular functions and arrow functions |
| Difference between Schema and Database |
| Difference between Normalization and Denormalization |
| Differences between interface and Class in Java |
| Differences between POP3 and IMAP |
| Difference between var and dynamic in C# |
| Difference between Multiprocessing and Multithreading |
| Difference between while and do-while loop in C, C++, Java |

Classroom program in NODIA Starting from 21st September 2019

Machine Learning

FOUNDATION WITH PYTHON

₹10,999

Register Now

2nd Batch

Mentored by Industry Expert

Works in ADORABLE

Good for foundations Machine Learning

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

Differences between TreeMap, HashMap and LinkedHashMap in Java

Prerequisite : HashMap and TreeMap in Java

TreeMap, HashMap and LinkedHashMap: What's Similar?

- All offer a **key->value** map and a way to iterate through the keys. The most important distinction between these classes is the time guarantees and the ordering of the keys.
- All three classes HashMap, TreeMap and LinkedHashMap implements **java.util.Map** interface, and represents mapping from unique key to values.

Key Points

- HashMap:** HashMap offers O(1) lookup and insertion. If you iterate through the keys, though, the ordering of the keys is essentially arbitrary. It is implemented by an array of linked lists.

Syntax:

```
public class HashMap extends AbstractMap implements Map, Cloneable, Serializable
```

- A HashMap contains values based on the key.
- It contains only unique elements.
- It may have one null key and multiple null values.
- It maintains **no order**.

Insertion order. It is implemented by doubly-linked buckets.

Syntax:

```
public class LinkedHashMap extends HashMap implements Map
```

- A LinkedHashMap contains values based on the key.
- 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:** TreeMap offers O(log N) lookup and insertion. Keys are ordered, so if you need to iterate through the keys in sorted order, you can. This means that keys must implement the Comparable interface. TreeMap is implemented by a Red-Black Tree.

Syntax:

```
public class TreeMap extends AbstractMap implements NavigableMap, Cloneable, Serializable
```

- A TreeMap contains values based on the key. It implements the NavigableMap interface and extends AbstractMap class.
- 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(Sorted using the natural order of its key)**.

- Hashtable:** "Hashtable" is the generic name for hash-based maps.

Syntax:

```
public class Hashtable extends Dictionary implements Map, Cloneable, Serializable
```

- 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 contains only unique elements.
- It may have not have any null key or value.
- It is synchronized.
- It is a legacy class.

HashMap LinkedHashMap TreeMap

```
// Java program to print ordering
// of all elements using HashMap
import java.util.*;
import java.lang.*;
import java.io.*;
class Main
{
    // This function prints ordering of all elements
    static void insertAndPrint(AbstractMap<Integer, String> map)
    {
        int[] array= {1, -1, 0, 2, -2};
        for (int x: array)
        {
            map.put(x, Integer.toString(x));
        }
        for (int k: map.keySet())
        {
            System.out.print(k + " ");
        }
    }

    // Driver method to test above method
    public static void main (String[] args)
    {
        HashMap<Integer, String> map = new HashMap<Integer, String>();
        insertAndPrint(map);
    }
}
```

Output of HashMap:

```
-1, 0, 1, -2, 2,
// ordering of the keys is essentially arbitrary (any ordering)
```

Output of LinkedHashMap:

```
1, -1, 0, 2, -2,
// Keys are ordered by their insertion order
```

Output of TreeMap:

```
-2, -1, 0, 1, 2,
// Keys are in sorted order
```

Comparison Table

| Property | HashMap | LinkedHashMap | TreeMap |
|---|---|---|--|
| Time Complexity(Big O notation) Get, Put, ContainsKey and Remove method | O(1) | O(1) | O(1) |
| Iteration Order | Random | Sorted according to either Insertion Order or Access Order (as specified during construction) | Sorted according to either natural Order of keys or comparator(as specified during construction) |
| Null Keys | allowed | allowed | Not allowed if keys uses Natural Ordering or Comparator does not support comparison on null Keys. |
| Interface | Map | Map | Map, SortedMap and NavigableMap |
| Synchronization | None, use Collections.synchronizedMap() | None, use Collections.synchronizedMap() | None, use Collections.synchronizedMap() |
| Data Structure | List of buckets, if more than 8 entries in bucket then Java 8 will switch to balanced tree from linked list | Doubly Linked List of Buckets | Red Black(a kind of self-balancing binary search tree) implementation of Binary Tree. This data structure offers O(log n) for insert, Delete and Search operations and O(n) space complexity. |
| Applications | General Purpose, fast retrieval, non-synchronized. ConcurrentHashMap can be used where concurrency is involved. | Can be used for LRU cache, other places where insertion or access order matters | Algorithms where Sorted or Navigable features are required. For example, find among the list of employees whose salary is next to given employee, Range Search, etc. |
| Requirements for keys | Equals() and hashCode() needs to be overwritten. | Equals() and hashCode() needs to be overwritten. | Comparator needs to be supplied for key implementation, otherwise natural order will be used to sort the keys. |

Real Life Applications

- Suppose you were creating a mapping of names to Person objects. You might want to periodically output the people in alphabetical order by name. A TreeMap lets you do this.
- A TreeMap also offers a way to, given a name, output the next 10 people. This could be useful for a "More" function in many applications.
- A LinkedHashMap is useful whenever you need the ordering of keys to match the ordering of insertion. This might be useful in a caching situation, when you want to delete the oldest item.
- Generally, unless there is a reason not to, you would use HashMap. That is, if you need to get the keys back in insertion order, then use LinkedHashMap. If you need to get the keys back in their true/natural order, then use TreeMap. Otherwise, HashMap is probably best. It is typically faster and requires less overhead.

This article is contributed by **Mr. Somesh Awasthi**. 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.

Recommended Posts:

[HashMap and TreeMap in Java](#)

[Program to Convert HashMap to TreeMap in Java](#)

[Differences between HashMap and Hashtable in Java](#)

[LinkedList in Java](#)

[LinkedList get\(\) Method in Java](#)

[LinkedList and LinkedHashSet in Java](#)

[LinkedList removeEldestEntry\(\) Method in Java](#)

[LinkedList clear\(\) Method in Java](#)

[LinkedList containsKey\(\) Method in Java](#)

[Print characters and their frequencies in order of occurrence using a LinkedHashMap in Java](#)

[TreeMap in Java](#)

[Java.util.TreeMap.firstEntry\(\) and firstKey\(\) in Java](#)

[Java.util.TreeMap.containsKey\(\) and containsValue\(\) in Java](#)

[Java.util.TreeMap.pollFirstEntry\(\) and pollLastEntry\(\) in Java](#)

[Java.util.TreeMap.floorEntry\(\) and floorKey\(\) in Java](#)

Article Tags :

[Difference Between](#) [Java](#) [Java-HashMap](#) [Java-Linked-HashMap](#) [Java-Map-Programs](#) [Java-TreeMap](#)

Practice Tags :

[Java](#)



9

☐ To-do ☐ Done

3.5

Based on 14 vote(s)

[Feedback/ Suggest Improvement](#)

[Add Notes](#)

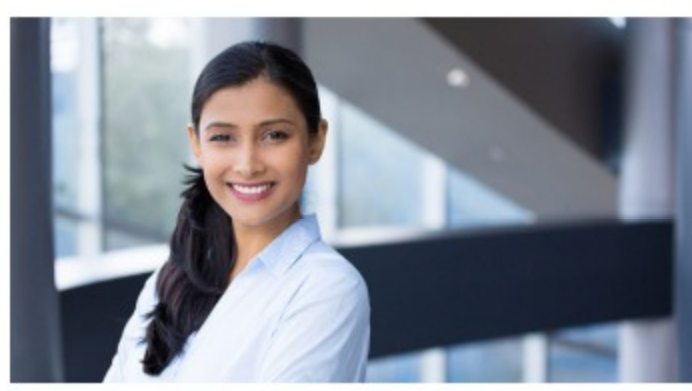
[Improve Article](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

[Previous](#)
[Java.net.HttpCookie in Java](#)

[Next](#)
[Flow control in try catch finally in Java](#)

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.



AMCAT Job Assessment Test

2000+ Top Companies
Recognise AMCAT Test