Difference between HashMap and ConcurrentHashMap in Java? Example

HashMap vs ConcurrentHashMap in Java

What is the difference between an HashMap and ConcurrentHashMap in Java is one of the common interview questions and knowing the answer is not just important for interviews but also for writing robust and high performance Java cocd. ConcurrentHashMap in Java is introduced as an alternative of Hashtable in Java, which is a synchronized collection class, that makes the main difference between HashMap and ConcurrentHashMap which is one is non-synchronized , non-thread safe and not for use in Concurrent multi-threaded environment while `ConcurrentHashMap` is a thread-safe collection and intended to be used as primary Map implementation especially for multi-threaded and Concurrent environment.

Apart from thread-safety and high performance there are some subtle differences between `HashMap` and `ConcurrentHashMap` which we will see in this article.

By the way, Difference between HashMap and ConcurrentHashMap as well as ConcurrentHashMap vs Hashtable are two popular core Java interview questions, mostly asked on senior level Java programmers.

Difference between HashMap and ConcurrentHashMap in Java

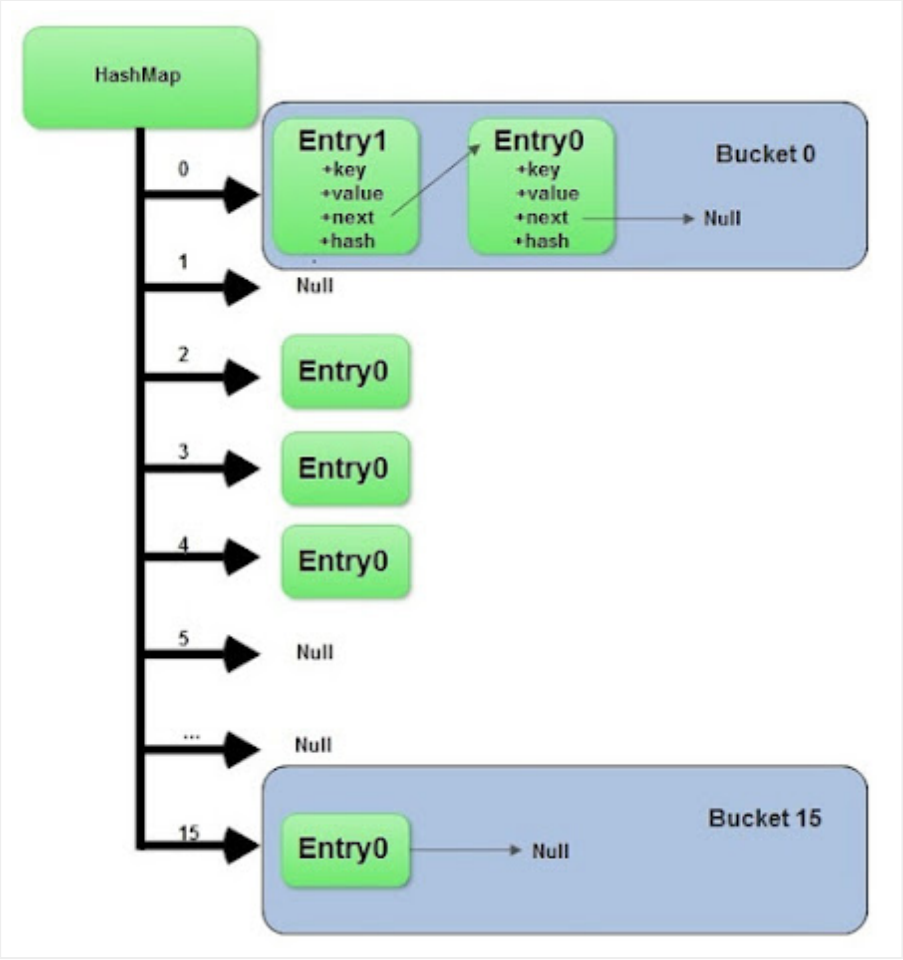
In this section, we will see some more details about HashMap and ConcurrentHashMap and compare them on various parameters like thread-safety, synchronization, performance, ease of use etc.

1. Thread Safety

As I said the earlier first significant difference between `HashMap` and `ConcurrentHashMap` is that later is thread-safe and can be used in a concurrent environment without external synchronization. Though it doesn't provide the same level of synchronization as achieved by using `Hashtable` but it's enough for the most practical purpose.

2. Synchronization

You can make `HashMap` synchronized by wrapping it on `Collections.synchornizedMap(HashMap)` which will return a collection which is almost equivalent to `Hashtable`, where every modification operation on `Map` is locked on Map object while in case of `ConcurrentHashMap`, thread-safety is achieved by dividing whole Map into different partition based upon Concurrency level and only locking particular portion instead of locking the whole Map.



3. Scalability

`ConcurrentHashMap` is more scalable and performs better than Synchronized `HashMap` in the multi-threaded environment while in Single threaded environment both `HashMap` and `ConcurrentHashMap` gives comparable performance, where HashMap only slightly better.

In Summary the main difference between ConcurrentHashMap and HashMap in Java Collection turns out to be thread-safety, Scalability, and Synchronization, and every Java developer should remember these different to use both of them correctly.

I think `ConcurrentHashMap` is a better choice than synchronized `HashMap` if you are using them as cache, which is the most popular use case of a Map in Java application. `ConcurrentHashMap` is more scalable and outperforms when a number of reader threads outnumber the number of writer threads.