**Concurrent Collection– Part\_04**

* **HashMap vs Hashtable vs ConcurrentHashMap:**

In HashMap, multiple threads can access simultaneously but there may be a chance of data inconsistency, hence it is not thread safe.

Hashtable is synchronized so there is no chance for data inconsistency but it will increase the waiting time of the thread as it allows only one thread to access at a time. This will impact the performance.

ConncurentHashMap allows multiple threads at a time with good performance and there won’t be data inconsistency.

Note: ConcurrentHashMap uses different locking mechanism to improve the performance.

* **Difference related to Locking Hashtable vs ConcurrentHashMap:**

In Hashtable, for read and write operation we will lock the entire map object.

But in ConcurrentHashMap, only for write and update operation the lock is required and for read operation lock is not required. For write or update operation the lock is done at bucket level.

The default capacity of the ConcurrentHashMap is 16, so 16 locks will be available.

In Hastable at a time only one Thread can operate.

In ConcurrentHashMap at a time 16 Threads can perform write/update operation.

Note:

We can call the lock as Bucket level locking or segment level locking.

Concurrency Level:

At a time how, many threads can perform a write or update operation with the lock.

default capacity = number of buckets = concurrency level

Note:

Most of the cases the bucket level and concurrency level will be same, but need not be all the times.

Let us say

Initial capacity = 16

Concurrency level = 8

Then for every two buckets one lock will be available.

Initial capacity = 16

Concurrency level = 32

Then for every bucket two locks will be available.

* **ConcurrentHashMap Conclusions:**

1. The underlying DS is HashTable.
2. ConcurrentHashMap allows concurrent read and thread-safe update operations.
3. To perform read operation thread won’t require any lock. But to perform update operation thread requires lock. But it is the lock of only a particular part of the map (Segment lock/ bucket level lock) instead of total map.
4. Concurrent update achieved by internally dividing map into smaller portions, which is defined by concurrency level.
5. The default concurrency level is 16.
6. ConcurrentHashMap allows any number of read operations but 16 update operations at a time by default.
7. Null is not allowed for both keys and values.
8. While one thread iterating, the other thread can perform update operation and ConcurrentHashMap never throw ConcurrentModificationException.

* **Constructors of ConcurrentHashMap:**

ConcurrentHashMap m = new ConcurrentHashMap();

Creates an empty ConcurrentHashMap with following default values:

Initial capacity = 16

Fill ratio = 0.75

Concurrency level = 16

ConcurrentHashMap m = new ConcurrentHashMap(int initialCapacity);

ConcurrentHashMap m = new ConcurrentHashMap(int initialCapacity, float fillRatio);

ConcurrentHashMap m = new ConcurrentHashMap(int initialCapacity, float fillRatio, int concurrencyLevel);

ConcurrentHashMap m = new ConcurrentHashMap(map);