**Concurrent Collections**

Traditional Collections:

1. Not Thread-safe
2. Performance wise very slow
3. Will through Concurrent Modification Exception when you are trying to modify the Collection when any operation is happening

Concurrent Collections:

1. Thread Safe
2. Related to Traditional Performance is high
3. Will not through Concurrent Modification Exception when you are trying to modify the Collection when any operation is happening.
4. **ConcurrentHashMap implements ConcurrentMap**

**ConcurrentMap** Interface has 3 methods:

1. Object putIfAbsent -- Put into map only if Key doesn’t exit
2. Boolean remove – map.remove(101, “Phani”) 🡪 If map contains 101 key with value “Phani”, then only it will remove. Else it won’t delete.
3. Boolean replace(Object key, Object oldValue, Object newValue) 🡪 If key and oldValue is matched, then only we can replace with new value.

* Underlying is Hashtable.
* CHM allows concurrent read
* To perform read operation thread won’t require any lock. But to perform update operation thread requires lock. But it is lock of only particular part of map (Segment lock/ Bucket level lock) instead of total map
* Concurrency is achieved by internally dividing map into smaller portions which is defined by concurrency level
* The default concurrency level is 16
* CHM allows any no. of read operations but only 16 update operations at a time by default
* Null is not allowed for keys and values
* While one thread iterating, the other thread can perform update operation and CHM never through ConcurrentModificationException

**Difference between HashMap and ConcurrentHashMap:**

|  |  |  |
| --- | --- | --- |
| Sl No: | HashMap | ConcurrentHashMap |
| 1 | Not Thread Safe | Thread Safe |
| 2 | Relatively Performance is high because threads are not required to wait to operate in HashMap | Relatively performance is low because sometimes threads are required to wait to operate on ConcurrentHashMap |
| 3 | Null keys and Null values are allowed | Neither null key or value is allowed |
| 4 | Iterator is Fail-Fast | Iterator is Fail-Safe |
| 5 | While one thread iterates the HashMap, then other threads are not allowed to modify map object otherwise we will ConcurrentModificationException | While one thread iterate ConcurrentHashMap, then other threads are allowed to modify map object. It doesn’t thrown ConcurrentModificationException |
| 6 | Introduced in 1.2 Version | Introduced in 1.5 version |
|  |  |  |

**Difference between ConcurrentHashMap, SynchronizedMap and HashTable**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl No | ConcurrentHashMap | SynchronizedMap | HashTable |
| 1 | We will get thread safety without locking total map object just with BucketLevel Lock. | We will get thread safety by locking Whole Map Object | We will get thread safety by locking Whole Map Object |
| 2 | At a time multiple threads are allowed to operate on Map object in Safe Manner | At a time only one Thread is allowed to perform any operation on Map object | At a time only one Thread is allowed to operation on Map object |
| 3 | Read Operation can be performed without any Lock but write Operation can be performed with Bucket Level lock | Every read and write operations require Total Map ObjectLock | Every read and write operations require Total Map ObjectLock |
| 4 | While One thread iterating Map Object, the other Threads are allowed to modify and won’t get ConcurrentModificationException | While One Thread iterating Map Object, the other threads are not allowed to Modify Map. Otherwise we will get ConcurrentModificationException | While One Thread iterating Map Object, the other threads are not allowed to Modify Map. Otherwise we will get ConcurrentModificationException |
| 5 | It is Fail-safe iterator | It is Fail-fast iterator | It is Fail-fast iterator |
| 6 | Null is not allowed for both Key and Values | Null is allowed for both Key and Values | Null is not allowed for both Key and Values |
| 7 | Introduced in 1.5 version | Introduced in 1.2 version | Legacy |
|  |  |  |  |

1. **CopyOnWriteArrayList**

* CopyOnWriteArrayList implements List Interface and List Interface extends Collection Interface
* It is a ThreadSafe version of ArrayList as the name indicates CopyOnWriteArrayList, creates a Cloned Copy of Underlying ArrayList for Every Update operation. At certain point both of them will Synchronized automatically which is taken care by JVM
* As Update operation will be performed on cloned copy there is no effect for the Threads which performs Read Operation
* It is costly to Use because for every Update Operation a cloned copy will be Created. Hence CopyOnWriteArrayList is the Best choice if Several Read Operations and Less number of wirte Operations
* Insertion order is Preserved
* Duplicate objects are allowed
* Heterogeneous objects are allowed
* Null insertion is possible
* It implements Serializable, Clonable and RandomAccess Interfaces
* While One Thread iterating CopyOnWriteArrayList, the Other threads are allowed to Modify and We Won’t get Concurrent Modification Exception
* This is Fail-safe iterator
* Iterator on ArrayList can perform Remove Operation but iterator on CopyOnWriteArrayList can’t perform Remove Operation. Otherwise we will get Runtime Exception saying UnsupportedOperationException

1. **CopyOnWriteArraySet**

* CopyOnWriteArraySet implements Set Interface and Set extends Collection Interface
* It is a Thread Safe version of Set
* Internally implement by CopyOnWriteArrayList
* Insertion Order is preserved
* Duplicate orders are not allowed
* Multiple threads are able to perform read operation simultaneously but for Every Update operation a separate cloned copy will be created
* As for every update operation a separate cloned copy will be created which is costly. Hence if multiple Update operation are required then it is not recommended to use CopyOnWriteArraySet
* While one thread is iterating Set the other Threads are allowed to Modify Set and we won’t get ConcurrentModificationException
* Iterator of CopyOnWriteArraySet can perform only read Operations and won’t perform Remove operation. Otherwise we will get Runtime Exception: UnsupportedOperationException.