1. **What is the purpose of the initial capacity and load factor parameters for each Collection class? What are their default values?**

The **initial capacity** is the number of buckets when the hash table is created, capacity is automatically increased when hash table get full of element.

**Load factor** is a measurement of how full hash table is allowed or number of elements in the hash table. When the hash table gets full of element its capacity automatically gets increased.

The internal data structures are rebuilt when the number of entries in the hash table exceeds the product of LF and IC

In collections, operations like get, put implementation have constant time performance but operation like in Iteration, time is proportional to capacity of hash map  and the size of the hash map instance. Set initial capacity too high or load factor too low for better performance. The default load factor is 0. 75. Increase in the value of LF will decrease the space over-head but increase lookup cost. Default load factor (.75) used by Hash Map is a good value in most situations and set the initial capacity based on how many items it will hold.

1. **What is the difference between HashMap and ConcurrentHashMap ?**

* [HashMap](https://www.geeksforgeeks.org/java-util-hashmap-in-java/) is the class under Traditional Collection and ConcurrentHashMap is a Class under Concurrent Collections.
* HashMap is non-Synchronized in nature i.e. HashMap is not Thread-safe whereas ConcurrentHashMap is thread-safe in nature.
* HashMap performance is relatively high because it is non-synchronized in nature and any number of threads can perform simultaneously. But ConcurrentHashMap performance is low sometimes because sometimes threads are required to wait on ConcurrentHashMap.
* When one thread is iterating the HashMap object, and other thread try to add/modify the contents of object then Run-time exception occurs saying ConcurrentModificationException, whereas in ConcurrentHashMap no exception occurs while performing any modification at the time of Iteration.
* In HashMap, null values are allowed for key and values, whereas in ConcurrentHashMap also null value is not allowed for key and value.
* HashMap is introduced in JDK 1.2 whereas ConcurrentHashMap is introduced by SUN Microsystem in JDK 1.5.

**3. Arrange the following in the ascending order (performance):**

**HashMap , Hashtable , ConcurrentHashMap and SynchronizedMap.**

Hashmap, concurrenthashtable, Hashtable, SynchronizedMap

**4. What is BlockingQueue in Java Collections Framework?**

java.util.concurrent.BlockingQueue is a java Queue that support operations wait for the queue to become non-empty when retrieving and removing an element, and wait for space to become available in the queue when adding an element. Java BlockingQueue doesn’t accept null values and throw NullPointerException if you try to store null value in the queue. Its implementations are thread-safe and all queuing methods are atomic in nature and use internal locks or other forms of [concurrency](https://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers) control. It is part of [java collections framework](https://www.journaldev.com/1260/collections-in-java-tutorial) and it’s primarily used for implementing producer consumer problem. Java provides several BlockingQueue implementations such as ArrayBlockingQueue, LinkedBlockingQueue, PriorityBlockingQueue, SynchronousQueue etc. ArrayBlockingQueue implementation is used for implementing producer consumer problem in BlockingQueue.

* put(E e): This method is used to insert elements to the queue. If the queue is full, it waits for the space to be available.
* E take(): This method retrieves and remove the element from the head of the queue. If queue is empty it waits for the element to be available.

**5. What is IdentityHashMap ?**

IdentityHashMap implements Map, Serializable and Clonable interfaces and extends AbstractMap class. This class is not a general-purpose Map implementation. While this class implements the [Map](https://www.geeksforgeeks.org/map-interface-java-examples/" \t "_blank)interface, it intentionally violates Map’s general contract, which mandates the use of the equals method when comparing objects. This class is used when the user requires the objects to be compared via reference.

**Constructors**

* **IdentityHashMap():**Constructs a new, empty identity hash map with a default expected maximum size.
* **IdentityHashMap(int expectedMaxSize):**Constructs a new, empty map with the specified expected maximum size.
* **IdentityHashMap(Map m):**Constructs a new identity hash map containing the keys-value mappings in the specified map.