**Collections\_Interview\_Questions**

**Q1) What is the difference between Comparable and Comparator**

**A)** Comparable provides the single sorting sequence. In other words, we can sort the collection on the basis of a single element such as Id, Name and Price

The Comparator provides multiple sorting sequences. In other words, we can sort the collection on the basis of multiple elements such as Id, Name and Price etc.

Comparable affects the original class

Comparator doesn’t affect the original class

Comparable provides CompareT0() method to sort the elements

Comparator provides the Compare() method to sort the elements

Comparable is present Java.lang package

Comparator is present in Java.util package.

If the sorting of objects needs to be based on natural order then use comparable

Whereas

if you sorting needs to be done on attribute of different objects then use comparator

In comparable:

* Positive Integer :- Current object is greater than specified object
* Negative Integer:- Current object is less then that specified object
* Zero :- Bothe Current object and specified objects are equal

**Q2) Can we use hash map in multithreading**

**A)** It’s not a problem to use hash map inside a multithreading environment as long as our application is not accessing/reading threads which are modifying the created hash map rather than simply accessing the hash map.

OR

It is a bug to have multiple threads use a non-synchronized collection in unprotected manner. If each thread had their own hash map instance then this is not issue. It is a problem if multiple threads are adding to the same hash map instance

If you need to use the same has table object in multithreading then you should consider using the concurrent hash map, because wrapping each of the access to the hash map in a synchronized block.

**Q3) Can we put duplicate elements in array list**

**A)** Yes, we can because it’s not separates the duplicate elements. It will allow duplicate elements to store.

**Q4) Internal working of hash map**

**A)** hash map is a part of the java collection framework. It uses a technic called hashing, it implements the map interface.

It stores the data in pair of key and values, hash map contains the array of nodes and the nodes represents as a class.

It uses an array and linked list data structure internally for sorting key and value. there are three methods in hash map.

1.equals(): It checks the equality of two object it compares the key, whether they are equal or not.

2.hashcode(): This is the method of the object class, it return the reference of the object in integer form. The value received from the method is used as the bucket number.

3.bucket(): Array of the nodes is called bucket each node as a data structure like a linked list, more than one node can share the same bucket.

**Q5) How java-8 hash map different from earlier hash map**

**A)** In java-8 hash map replaces the linked list with another useful data structure called binary tree on breaching a certain threshold.

Once this threshold is reached the linked list of entries is converted to the tree nodes which is reduces the time complexity from o(n) to o(log(n)).

**Q6) What is concurrent hash map and how it is different from hash map**

**A)** Concurrent hash map implements from the concurrent map as well as serializable interface also concurrent hash map is an enhancement of hash map.

The main difference between hash map and concurrent hash map is that concurrent hash map internally synchronized and hence it is thread safe.

Where as hash map is not internally synchronized and it is not thread safe. We can make hash map synchronized externally by using

Collection.syncronizedmap() method.

Hash map allows maximum one null key and any number of null values. Concurrent hash map doesn’t allow even a single null key and a null value.

Hash map is faster and most suitable for single thread applications

Concurrent hasp map is slower and most suitable for multithreaded applications.

**Q7) What is the difference between map() and flatmap() in stream API**

**A)** Map is used for transformation and flatmap is used for both transformation and flattening that’s why it’s called the flatmap.

The key difference between map() and flatmap() functions is that when we uses a map(), it applies a function on each element of the streams and stores the value returned by the function into a new stream

The flatmap() function is the combination of a map and a flat operation. This means you first apply the map function and then flattening the result.

The key difference is the function used by map operation returns a stream of values or a list of values rather then a single value.

**Q8) What is fail-safe and fail-fast iterator**

**A)** Fail-safe and fail-fast are the concepts of concurrent modification. Concurrent modification is a process in which an object is modified concurrently when a different task running on it.

**Fail-fast:** When use the fail-fast, it immediately throws concurrent modification Exception when an element is added or removed from the collection while the thread is iterating over the collection

The Fail-fast iterator use modcount to check whether the collection is structurally modified or not.

**Fail-Safe:** Fail-safe iterator doesn’t throw the concurrent modification exception, and it tried to avoid rising the exception. The fail-safe iterator creates a copy of the original collection or object array and iterates over that copied collection, if any modification made in the iterator affects the copied collection, not the original collection.

**Q9) Which collection is sorted**

**A)** Treeset is a sorted collection

**Q10) What is the difference between array list and linked list, when to use array list over the linked list.**

**A)** Array List and LinkedList both implement the list interface and maintain insertion order, both are non-synchronized classes.

|  |  |
| --- | --- |
| **ArrayList** | **LinkedList** |
| Array list internally uses a dynamic array to store the elements | Linked list internally uses a double linked list to store the elements |
| Manipulation with array list is slow because it internally uses an array, if any element is removed from the array all other elements are shifted in memory | Manipulation with linked list is faster than array List because it uses a double linked list, So no bit shifting is required in memory |
| An array list class can act as a list only because it implements list only | Linked list class can act as a list and queue both because it implements list and D-queue interfaces |
| ArrayList is better for sorting and accessing data | LinkedList better for manipulating data |
| An arraylist is a resizable array | Linked list implements the double linked list of the list interface |

**Q11) What is CopyOnRightArray**

**A)** It is enhanced version of array list in which all modifications (add, set, remove, etc) are implemented by making a fresh copy. It is in java.util.concurrent package.

Copy on right array list creates a cloned copy of underlying array list, for every update operation. Certain point of time both will be synchronized automatically. Which is taken by JVM.

**Q12) Some examples of synchronized and concurrent collections**

**A)** Both synchronized and concurrent collection classes provide thread safety. The difference between them is performance and scalability. Synchronized collections are much slower than concurrent collections.

**Synchronized**  **Concurrent**

Synchronized hashmap Concurrent hashmap

Hashtable Copyonwritearraylist

Vector Copyonwritehashset

Synchronized arrayList

The main reason of slowness is locking. Synchronized collections lock the whole collection e.g: Map or List

While concurrent collection never locks the whole Map or List.

**Q13) what is the difference between hashtable and concurrent hashmap**

**A)**

|  |  |
| --- | --- |
| **Hash Table** | **Hash Map** |
| Hash table is thread safe and synchronized | Concurrent hash map also a thread safe and synchronized |
| It applies the lock on the entire collection | Concurrent hash map applies locks only at bucket level called segment while adding or updating the map |
| It is slower than concurrent hash map | It is better than hash table |
| It doesn’t allow null key and null values | It allows null keys and values |

**Q14) How to convert List into Map**

**A)** We can convert the list into map by using streams and collectors

Ex:

**public** Map<Integer, Animal> **convertListAfterJava8**(List<Animal> list)

{

Map<Integer, Animal> map = list.stream() .collect(Collectors.toMap(Animal::getId, Function.identity())); **return** map;

}