20 Java Collections Interview Questions

In java, collection interview questions are mostly asked by the interviewers. Here is the list of mostly asked collections interview questions with answers.

1) What is the difference between ArrayList and Vector?

|  |  |  |
| --- | --- | --- |
| **No.** | **ArrayList** | **Vector** |
| 1) | ArrayList is not synchronized. | Vector is synchronized. |
| 2) | ArrayList is not a legacy class. | Vector is a legacy class. |
| 3) | ArrayList increases its size by 50% of the array size. | Vector increases its size by doubling the array size. |

2) What is the difference between ArrayList and LinkedList?

|  |  |  |
| --- | --- | --- |
| **No.** | **ArrayList** | **LinkedList** |
| 1) | ArrayList uses a dynamic array. | LinkedList uses doubly linked list. |
| 2) | ArrayList is not efficient for manipulation because a lot of shifting is required. | LinkedList is efficient for manipulation. |
| 3) | ArrayList is better to store and fetch data. | LinkedList is better to manipulate data. |

3) What is the difference between Iterator and ListIterator?

Iterator traverses the elements in forward direction only whereas ListIterator traverses the elements in forward and backward direction.

|  |  |  |
| --- | --- | --- |
| **No.** | **Iterator** | **ListIterator** |
| 1) | Iterator traverses the elements in forward direction only. | ListIterator traverses the elements in backward and forward directions both. |
| 2) | Iterator can be used in List, Set and Queue. | ListIterator can be used in List only. |

4) What is the difference between Iterator and Enumeration?

|  |  |  |
| --- | --- | --- |
| **No.** | **Iterator** | **Enumeration** |
| 1) | Iterator can traverse legacy and non-legacy elements. | Enumeration can traverse only legacy elements. |
| 2) | Iterator is fail-fast. | Enumeration is not fail-fast. |
| 3) | Iterator is slower than Enumeration. | Enumeration is faster than Iterator. |

5) What is the difference between List and Set?

List can contain duplicate elements whereas Set contains only unique elements.

6) What is the difference between HashSet and TreeSet?

HashSet maintains **no order** whereas TreeSet maintains **ascending order**.

7) What is the difference between Set and Map?

Set contains values only whereas Map contains key and values both.

8) What is the difference between HashSet and HashMap?

HashSet contains only values whereas HashMap contains entry(key,value). HashSet can be iterated but HashMap need to convert into Set to be iterated.

9) What is the difference between HashMap and TreeMap?

HashMap maintains **no order** but TreeMap maintains **ascending order**.

10) What is the difference between HashMap and Hashtable?

|  |  |  |
| --- | --- | --- |
| **No.** | **HashMap** | **Hashtable** |
| 1) | HashMap is not synchronized. | Hashtable is synchronized. |
| 2) | HashMap can contain one null key and multiple null values. | Hashtable cannot contain any null key or null value. |

11) What is the difference between Collection and Collections?

Collection is an interface whereas Collections is a class. Collection interface provides normal functionality of data structure to List, Set and Queue. But, Collections class is to sort and synchronize collection elements.

12) What is the difference between Comparable and Comparator?

|  |  |  |
| --- | --- | --- |
| **No.** | **Comparable** | **Comparator** |
| 1) | Comparable provides only one sort of sequence. | Comparator provides multiple sort of sequences. |
| 2) | It provides one method named compareTo(). | It provides one method named compare(). |
| 3) | It is found in java.lang package. | it is found in java.util package. |
| 4) | If we implement Comparable interface, actual class is modified. | Actual class is not modified. |

13) What is the advantage of Properties file?

If you change the value in properties file, you don't need to recompile the java class. So, it makes the application easy to manage.

14) What does the hashCode() method?

The hashCode() method returns a hash code value (an integer number).

The hashCode() method returns the same integer number, if two keys (by calling equals() method) are same.

But, it is possible that two hash code numbers can have different or same keys.

15) Why we override equals() method?

The equals method is used to check whether two objects are same or not. It needs to be overridden if we want to check the objects based on property.

For example, Employee is a class that has 3 data members: id, name and salary. But, we want to check the equality of employee object on the basis of salary. Then, we need to override the equals() method.

16) How to synchronize List, Set and Map elements?

Yes, Collections class provides methods to make List, Set or Map elements as synchronized:

|  |
| --- |
| public static List synchronizedList(List l){} |
| public static Set synchronizedSet(Set s){} |
| public static SortedSet synchronizedSortedSet(SortedSet s){} |
| public static Map synchronizedMap(Map m){} |
| public static SortedMap synchronizedSortedMap(SortedMap m){} |

17) What is the advantage of generic collection?

If we use generic class, we don't need typecasting. It is typesafe and checked at compile time.

18) What is hash-collision in Hashtable and how it is handled in Java?

Two different keys with the same hash value is known as hash-collision. Two different entries will be kept in a single hash bucket to avoid the collision.

19) What is the Dictionary class?

The Dictionary class provides the capability to store key-value pairs.

20) What is the default size of load factor in hashing based collection?

The default size of load factor is **0.75**.

The default capacity is computed as initial capacity \* load factor.

For example, 16 \* 0.75 = 12. So, 12 is the default capacity of Map.

1. HashMap class has one nested interface with name Entry (Node from JDK 1.8) which implements one interface called Map.Entry interface
2. Map interface is also having one method interface called Entry and this interface has got 3 method
3. whenever we create the object of HashMap class using default Constructor and then one Entry (Node from JDK 1.8) class on array will be created of size 16 which is known as capacity of HashMap
4. each index of this array is called a bucket
5. and each bucket is a linked list
6. whenever we add any key value pair in a HashMap class using a put () method then the put () method first convert this key value pair into the single object of Entry (Node from JDK 1.8) class

and this Entry (Node from JDK 1.8) class object is going to be stored into the array of this class which is defined in a HashMap class

How an entry class object is stored in an array: -?

1. one the hashcode of the key class object is first calculated
2. calculating the remainder of this hashcode by dividing it via total number of bucket or capacity
3. this reminder is treated as index of bucket
4. once the bucket is decided then the equals method is called on a key class object and then passes existing key class object into this method one by one
   1. if the match is found from the entry class object, it does not add into the bucket but the value of the key in the bucket will be replaced by the new value
   2. otherwise this entry class object is added in a bucket as first node of the linked list which is maintained by a bucket

Some important points on HashMap

1. the default size of array is 16 (always a power of 2) and the load factors means whenever the size of HashMap reaches to 75% of the current size i.e. 12, it will double its size by re-computing the hash codes of existing Data Structure elements
2. to avoid re-hashing of data structure (as it degrades the performance) as element grow it is best practice to explicitly give the size of HashMap while creating it
   1. it never gives the capacity to high and the load factor too low
3. Since Java is multithreaded it is very possible that more than one thread might be using same HashMap and then they both realize the need for resizing the HashMap at the same time, which lead to race condition

what is race condition with respect to HashMap?

when to said see the need for resizing the same HashMap, they might end up adding the elements of old buckets to the new buckets simultaneously and hence might lead to infinite loops.

in case of Collision i.e. when there are different keys with same hashcode, internally we use singly linked list to store the elements and we store every new element at the head of a linked list to avoid tail traversing and hence at the time of resizing the entire sequence of object in the linked list gets reversed, during which there are chances of infinite loop.

example => let's assume there are three keys with the same hashcode and hence are stored in linked list inside a bucket

which becomes infinite loop for the next iteration and thread hangs here.

The HashMap class uses a Hash table to implement the map interface. The HashMap class in not thread safe and permits only one null key and null value.

the HashMap class is roughly equal to the Hashtable except that it is unsynchronized and permits null. This class makes no guarantee as to the order of the map

HashMap can be synchronized by using a method map m = collection synchronized map