300+ [LATEST] Java Collections Interview Questions and Answers

Q1. How To Sort Arraylist Of Strings In Descending Order?

Collections.sort() method will sort the ArrayList of String in ascending order.

To have a descending order either comparator has to be provided or reverseOrder method of the Collections class can be used.

Collections.sort(cityList, Collections.reverseOrder());

Q2. Which Set Implementation Should Be Used If You Want The Insertion Order To Be Maintained?

LinkedHashSet should be used in this case.

Q3. Difference Between Hashmap And Hashtable In Java?

Though both HashTable and HashMap store elements as a (key, value) pair and use hashing technique to store elements, moreover from Java v1.2, HashTable class was retrofitted to implement the Map interface, making it a member of the Java Collections Framework. But there are certain difference between the two –

- HashMap is not synchronized where as HashTable is synchronized.
- HashMap allows one null value as a key and any number of null values where as HashTable does not allow null values either as key or as value.
- For traversing a HashMap an iterator can be used. For traversing a HashTable either an iterator or Enumerator can be used. The iterator used for both HashMap and HashTable is fail-fast but the enumerator used with HashTable is fail-safe.
- Performance wise HashMap is faster than the HashTable reason being HashMap is not synchronized.

Q4. How Arraylist Works Internally In Java?

Basic data structure used by ArrayList to store objects is an array of Object class, which is defined like –

trient Object[] elementData;

When we add an element to an ArrayList it first verifies whether it has that much capacity in the array to store new element or not, in case there is not then the new capacity is calculated which is 50% more than the old capacity and the array is increased by that much capacity (Actually uses Arrays.copyOf which returns the original array increased to the new length)

Q5. What Is Linkedlist Class In Java? How Is It Implemented?

LinkedList class in Java implements List and Deque interfaces and LinkedList implements it using doubly linkedlist.

Within the LinkedList implementation there is a private class Node which provides the structure for a node in a doubly linked list. It has

item variable for holding the value and two reference to Node class itself for connecting to next and previous nodes.

Q6. What Is Java Collections Framework? What Are The Benefits Of Using Collections Framework?

Java Collections Framework provides a standard way to handle a group of objects. Benefits of the Collections Framework are –

- High performance, implementation of the Collection classes (ArrayList, LinkedList, HashSet etc.) are very efficient and used as-is in most of the cases.
- Reduced effort as framework already provides several implementations for most of the scenarios that can be used as-is.
- Provides interoperability, as exp. if you are using List interface any implementation that implements List interface can be swapped with the existing one.
- If you want to extend any collection that can be easily done using the standard interfaces provided with in the Collections frameworks.

Q7. Which Collection Classes Implement List Interface?

Some of the Collection classes that implement List interface are ArrayList, CopyOnWriteArrayList, LinkedList, Stack, Vector.

Q8. How To Add Elements To An Arraylist?

- List provides a method add(E e) which appends specified element to the end of the list. Using add(E e) method will mean keep adding elements sequentially to the list.
- Another add method – add(int index, E element) inserts the specified element at the specified position in this list.
- Third method addAll(int index, Collection c) inserts all of the elements in the specified collection into this list, starting at the specified position.

Q9. What Is For-each Style Loop?

Another feature which was added to Collection with JDK 5 is for-each style loop. Any collection which wants to be the target of the “for-each loop” statement has to implement iterable interface.

Using for-each loop is easier than constructing the iterator to iterate over the collection and can be used in most of the cases rather than using the iterator loop.

If you have a list of Strings, that can be iterated using for-each loop like this.


```
for(String city : cityList){
   System.out.println(“Name ” + city);
}
```

Q10. Describe The Collection Framework Hierarchy?

At the root of the Collections framework is Collection interface, it must be implemented by any class that defines a collection. This interface declares the core methods that every collection will have, if any class doesn't implement any of the method then it can throw UnsupportedOperationException.

Then there are List and Set interfaces that extend Collection interface and provided some of its own behaviour that will be further implemented by the classes that implement List and Set interfaces respectively.

There is also a Queue interface that extends collection to provide behaviour of a queue. On the other hand there is Map interface which provides core methods for the Map implementations.

Q11. Why Iterator Doesn't Have Add Method Whereas Listiterator Has Add Method?

Iterator can be obtained on any Collection class like List or Set so contract for Iterator makes no guarantees about the order of iteration.

But ListIterator can only be used to traverse a List so it does guarantee the order of the iteration. That is why ListIterator provides an add operation.

Q12. What Is Listiterator In Java?

ListIterator provides the functionality to iterate a list in both directions. The interesting point about list iterator is that it has no current element. Its current cursor position always lies between the element that would be returned by a call to previous() and the element that would be returned by a call to next().

Q13. What Is Hash-collision In Hash Based Collections?

In HashMap, using the key, a Hash is calculated and that hash value decides in which bucket the particular Map.Entry object will reside.

Hash collision me more than one key having the same calculated hash value thus stored in the same bucket. In HashMap, in that case Entry objects are stored as a linked-list with in the same bucket.

Q14. When Do We Need To Override Hashcode() And Equals() Methods?

The default implementation of equals() method in the Object class is a simple reference equality check.

```
public boolean equals(Object obj){
    return (this == obj);
}
```

The default implementation of hashCode() in the Object class just returns integer value of the memory address of the object.

It becomes very important to override these two methods in case we are using a custom object as key in a hash based collection.

In that case we can't rely on the default implementation provided by the Object class and need to provide custom implementation of hashCode() and equals() method.

If two objects Obj1 and Obj2 are equal according to their equals() method then they must have the same hash code too. Though the vice-versa is not true that is if two objects have the same hash code then they do not have to be equal too.

Q15. Difference Between Arraylist And Linkedlist In Java?

In Java collections framework ArrayList and LinkedList are two different implementations of List interface

- LinkedList is implemented using a doubly linked list concept where as ArrayList internally uses an array of Objects which can be resized dynamically
- For LinkedList add(e Element) is always O(1) where as for ArrayList add(e Element) operation runs in amortized constant time, that is, adding n elements requires O(n) time.
- For LinkedList get(int index) is O(n) where as for ArrayList get(int index) is O(1).
- If you are removing using the remove(int index) method then for LinkedList class it will be O(n). In case of ArrayList getting to that index is fast but removing will mean shuffling the remaining elements to fill the gap created by the removed element with in the underlying array.

Q16. How Generics Changed The Collection Framework?

With JDK5 the Collection framework was reengineered to add generics. That was done to add “Type Safety” to the collections. Prior to JDK5 (and generics) elements were stored in collections as Object references, which brought the danger of accidentally storing incompatible types in Collection because for Collections everything was Object reference. Which would have resulted in run time error when trying to retrieve elements and casting them to the desired type.

With the introduction of generics now we can explicitly tell which data type is to be stored in the collections, which helps in avoiding the run time error. As Collection will throw error at compile time itself for incompatible data type. As exp.

Map cityTemperatureMap = new LinkedHashMap();

Here we are explicitly stating that this LinkedHashMap can only store string as both key and value.

Q17. What Is A Weakhashmap?

WeakHashMap is a Hash table based implementation of the Map interface, with weak keys. An entry in a WeakHashMap will automatically be removed when its key is no longer in ordinary use. Which me storing only weak references allows garbage collector to remove the entry (key-value pair) from the map when its key is not referenced outside of the WeakHashMap.

In WeakHashMap both null values and the null key are supported. A WeakHashMap is created as-

Map weakHashMap = new WeakHashMap();

Q18. Difference Between Iterator And Listiterator?

- Iterator can be obtained on any Collection class like List or Set. But ListIterator can only be used to traverse a List.
- Iterator only moves in one direction using next() method. ListIterator can iterate in both directions using next() and previous() methods.
- Iterator always start at the beginning of the collection. ListIterator can be obtained at any point.

  As Exp. If you have a List of integers numberList, you can obtain a ListIterator from the third index of this List.

ListIterator ltr = numberList.listIterator(3);

- ListIterator provides an add(E e) method which is not there in Iterator. add(E e) inserts the specified element into the list.
- ListItearator also provides set method. void set(E e) replaces the last element returned by next() or previous() with the specified

element

Q19. What Is A Diamond Operator?

Diamond operator let the compiler infer the type arguments for the generic classes. It is added in Java 7.

As Example – Before JDK7 if we had to define a Map using String as both Key and Value we had to write it like this –

Map cityMap = new HashMap();

In Java SE 7, you can substitute the parameterized type of the constructor with an empty set of type parameters (<>) known as diamond operator.

Map cityMap = new HashMap<>();

Q20. How To Join Two Or More Arraylists?

List provides a method addAll to join two or more lists in Java.

If you have one list cityList and another List secondCityList then you can join them using addAll like this –

cityList.addAll(secondCityList);

Q21. Does Arraylist Allow Null?

In ArrayList any number of nulls can be added.

Q22. How To Make A Collection Class Immutable?

Collections class provides static method to make a Collection unmodifiable. Each of the six core collection interfaces -Collection, Set, List, Map, SortedSet, and SortedMap – has one static factory method.

public static Collection unmodifiableCollection(Collection c);

public static Set unmodifiableSet(Set s);

public static List unmodifiableList(List list);

public static Map unmodifiableMap(Map m);

public static SortedSet unmodifiableSortedSet(SortedSet s);

public static SortedMap unmodifiableSortedMap(SortedMap m);

Q23. How To Remove Elements From An Arraylist?

ArrayList provides several methods to remove elements from the List. Since ArrayList internally uses array to store elements, one point to note is that when an element is removed from the List the remaining elements have to be shifted to fill the gap created in the underlying array.

- clear() – Removes all of the elements from this list.
- remove(int index) – Removes the element at the specified position in this list.
- remove(Object o) – Removes the first occurrence of the specified element from this list, if it is present.
- removeAll(Collection c) – Removes from this list all of its elements that are contained in the specified collection.
- removeIf(Predicate filter) – Removes all of the elements of this collection that satisfy the given predicate.

Q24. How To Remove Duplicate Elements From An Arraylist?

We can use a HashSet to do the job of removing the duplicate elements. HashSet only stores unique elements and we'll use that feature of HashSet to remove duplicates.

If you have a List called cityList you can create a HashSet using this list –

```
Set citySet = new HashSet(cityList);
then add this set back to the cityList
cityList.clear();
cityList.addAll(citySet);
```

That will remove all the duplicate elements from the given list. Note that insertion order won't be retained if a HashSet is used. In case insertion order is to be retained use LinkedHashSet.

Q25. What All Collection Classes Are Inherently Thread-safe?

In the initial Collection classes like Vector, HashTable and Stack all the methods were synchronized to make these classes thread safe.

Later implementations starting from Java 1.2 were not synchronized.

Classes in java.util.concurrent package like ConcurrentHashMap, CopyOnWriteArrayList also provides thread safety but with a different implementation that doesn't require making all the methods synchronized.

Q26. Why Arraylist Is Called Dynamically Growing Array? How That Dynamic Behaviour Is Achieved?

In the case of ArrayList you don't have to anticipate in advance, like in the case of array, how many elements you are going to store in the arraylist. As and when elements are added list keeps growing. That is why ArrayList is called dynamically growing array.

For ArrayList, data structure used for storing elements is array itself. When ArrayList is created it initializes an array with an initial capacity (default is array of length 10). When that limit is crossed another array is created which is 1.5 times the original array and the elements from the old array are copied to the new array.

Q27. Which Collection Classes Implement Random Access Interface?

In Java.util package the classes that implement random access interface are – ArrayList, CopyOnWriteArrayList, Stack, Vector.

Q28. What Happens If Hashmap Has Duplicate Keys?

If an attempt is made to add the same key twice, it won't cause any error but the value which is added later will override the previous value.

As Exp. If you have a Map, cityMap and you add two values with same key in the cityMap, Kolkata will override the New Delhi.

cityMap.put("5", "New Delhi");
cityMap.put("5", "Kolkata");

Q29. What Is Foreach Statement Added In Java 8?

Java 8 has added a functional style lopping to the Collections. Real advantage of this loop is when it is used on a stream with the chain of functional methods. If you have a Map then it can be looped using ForEach statement like this –

Set> valueSet = cityMap.entrySet();

valueSet.forEach((a)->System.out.println("Key is " + a.getKey() + " Value is " + a.getValue()));

Q30. What Is List Interface?

List interface extends the root interface Collection and adds behaviour for the collection that stores a sequence of elements. These elements can be inserted or accessed by their position in the list using a zero-based index.

Q31. How Can We Synchronize A Collection Or How To Make A Collection Thread-safe?

Collections class has several static methods that return synchronized collections. Each of the six core collection interfaces -Collection, Set, List, Map, SortedSet, and SortedMap – has one static factory method.

```
public static Collection synchronizedCollection(Collection c);
public static Set synchronizedSet(Set s);
public static List synchronizedList(List list);
public static Map synchronizedMap(Map m);
public static SortedSet synchronizedSortedSet(SortedSet s);
public static SortedMap synchronizedSortedMap(SortedMap m);
```

Each of these methods returns a synchronized (thread-safe) Collection backed up by the specified collection.

Q32. What Is Linkedhashset?

LinkedHashSet is also one of the implementation of the Set interface. Actually LinkedHashSet class extends the HashSet and has no other methods of its own.

LinkedHashSet also stores unique elements just like other implemetations of the Set interface. How LinkedHashSet differs is that it maintains the insertion-order; that is elements in the LinkedHashSet are stored in the sequence in which they are inserted. Note that insertion order is not affected if an element is re-inserted into the set.

Q33. What Is Linkedhashmap?

LinkedHashMap is also one of the implementation of the Map interface, apart from implementing Map interface LinkedHashMap also extends the HashMap class. So just like HashMap, LinkedHashMap also allows one null key and multiple null values.

How it differs from other implementations of the Map interface like HashMap and TreeMap is thatLinkedHashMap maintains the insertion order of the elements which me if we iterate a

LinkedHashMap we'll get the keys in the order in which they were inserted in the Map.

LinkedHashMap maintains a doubly-linked list running through all of its entries and that's how it maintains the iteration order.

Q34. Hashmap Vs Linkedhashmap Vs Treemap In Java?

- HashMap makes no guarantees as to the order of the map. LinkedHashMap maintains the insertion order of the elements which me if we iterate a LinkedHashMap we'll get the keys in the order in which they were inserted in the Map.

  TreeMap stores objects in sorted order.

- HashMap as well as LinkedHashMap allows one null as key, multiple values may be null though.
  TreeMap does not allow null as key.

- HashMap stores elements in a bucket which actually is an index of the array.
  LinkedHashMap also uses the same internal implementation, it also maintains a doubly-linked list running through all of its entries.

  TreeMap is a Red-Black tree based NavigableMap implementation.

- Performance wise HashMap provides constant time performance O(1) for get() and put() method.
  LinkedHashMap also provides constant time performance O(1) for get() and put() method but in general a little slower than the HashMap as it has to maintain a doubly linked list.

  TreeMap provides guaranteed log(n) time cost for the containsKey, get, put and remove operations.

Q35. What Is Map.entry In Map?

Map.Entry is an interface in java.util, in fact Entry is a nested interface in Map interface. Nested interface must be qualified by the name of the class or interface of which it is a member, that's why it is qualified as Map.Entry.

Map.Entry represents a key/value pair that forms one element of a Map.

The Map.entrySet method returns a collection-view of the map, whose elements are of this class.

As exp. If you have a Map called cityMap then using entrySet method you can get the set view of the map whose elements are of type Map.Entry, and then using getKey and getValue methods of the Map.Entry you can get the key and value pair of the Map.

```
for(Map.Entry entry:  cityMap.entrySet()){

  System.out.println("Key is " + entry.getKey() + " Value is " + entry.getValue());

}
```

Q36. Difference Between Arraylist And Vector In Java?

- ArrayList is not synchronized whereas Vector is synchronized.
- Performance wise ArrayList is fast in comparison to Vector as ArrayList is not synchronized.
- ArrayList, by default, grows by 50% in case the initial capacity is exhausted. In case of Vector the backing array's size is doubled by default.
- For traversing an ArrayList iterator is used. For traversing Vecor Iterator/Enumerator can be used. Note that Iterator is fail-fast for both Vector and ArrayList. Enumerator which can be used with Vector is not fail-fast.

Q37. How To Sort Arraylist In Java?

Collections class has a static method sort which can be used for sorting an arraylist.

There are two overloaded versions of sort method –

- public static > void sort(List list) – Sorts the specified list into ascending order, according to the natural ordering of its elements.
- public static void sort(List list, Comparator c) – Sorts the specified list according to the order induced by the specified comparator.

If you have a List called cityList it can be sorted like this –

Collections.sort(cityList);

Q38. Difference Between Fail-fast Iterator And Fail-safe Iterator?

- fail-fast iterator throws a ConcurrentModificationException if the underlying collection is structurally modified whereas fail-safe iterator doesn't throw ConcurrentModificationException.
- fail-fast iterator doesn't create a copy of the collection whereas fail-safe iterator makes a copy of the underlying structure and iteration is done over that snapshot.
- fail-fast iterator provides operations like remove, add and set (in case of ListIterator) whereas in case of fail-safe iterators element-changing operations on iterators themselves (remove, set and add) are not supported. These methods throw UnsupportedOperationException.

Q39. Which Map Implementation Should Be Used If You Want Map Values To Be Sorted By Keys?

TreeMap should be used in this case.

Q40. What Is A Fail-safe Iterator?

An iterator is considered fail-safe if it does not throw ConcurrentModificationException. ConcurrentModificationException is not thrown as the fail-safe iterator makes a copy of the underlying structure and iteration is done over that snapshot.

Since iteration is done over a copy of the collection so interference is impossible and the iterator is guaranteed not to throw ConcurrentModificationException.

Q41. What Is A Identityhashmap?

IdentityHashMap class implements the Map interface with a hash table, using reference-equality in place of object-equality when comparing keys (and values). In other words, in an IdentityHashMap, two keys k1 and k2 are considered equal if and only if (k1==k2). Where as In normal Map implementations (like HashMap) two keys k1 and k2 are considered equal if and only if (k1==null ? k2==null : k1.equals(k2)).

Note that This class is not a general-purpose Map implementation. While this class implements the Map interface, it intentionally violates Map's general contract, which mandates the use of the equals method when comparing objects. This class is designed for use only in the rare cases wherein reference-equality semantics are required.

Q42. How To Convert Array To Arraylist In Java?

Arrays class contains a static factory method asList() that allows arrays to be viewed as lists.

public static List asList(T... a)

As Exp.

String cityArray[] = {“Delhi”, “Mumbai”, “Bangalore”, “Hyderabad”, “Chennai”};

//Converting array to List

```
List cityList = Arrays.asList(cityArray);
```

Q43. What Is The Hashcode() Method?

hashCode() method is present in the java.lang.Object class. This method is used to get a unique integer value for a given object. We can see it's use with hash based collections like HashTable or HashMap where hashCode() is used to find the correct bucket location where the particular (key, value) pair is stored.

Q44. How And Why To Synchronize Arraylist In Java?

In order to have better performance most of the Collections are not synchronized. Which me sharing an instance of arrayList among many threads where those threads are modifying (by adding or removing the values) the collection may result in unpredictable behaviour.

To synchronize a List Collections.synchronizedList() method can be used.

Q45. What Is Randomaccess Interface?

RandomAccess interface is a marker interface used by List implementations to indicate that they support fast (generally constant time) random access.

Q46. Can We Iterate Through A Map?

Though we can't iterate a Map as such but Map interface has methods which provide a set view of the Map. That set can be iterated. Those two methods are –

- Set>entrySet() – This method returns a set that contains the entries in the map. The entries in the set are actually object of type Map.Entry.
- SetkeySet() – This method returns a set that contains the keys of the map.

There is also a values() method in the Map that returns a Collection view of the values contained in this map.

Q47. How To Loop/iterate An Arraylist In Java?

There are many ways to loop/iterate an arrayList in Java. Options are –

- for loop
- for-each loop
- iterator
- list iterator
- Java 8 forEach loop

for-each loop is the best way to iterate a list if you just need to traverse sequentially through a list.

Q48. Are Maps Actually Collections Or Why Doesn't Map Extend Collection?

Maps themselves are not collections because they don't implement Collection interface.

Q49. Does Hashmap Allow Null Keys And Null Values?

HashMap allows one null key and any number of null values. If another null key is added it won't cause any error. But the value with the new null key will override the value with old null key.

As exp. If you have a Map, cityMap and you add two values with null keys in the cityMap, Kolkata will override the New Delhi as only one null key is allowed.

cityMap.put(null, "New Delhi");
cityMap.put(null, "Kolkata");

Q50. Difference Between Array And Arraylist In Java?

- Array is fixed in size which is provided at the time of creating an Array. ArrayList grows dynamically and also known as dynamically growing array.
- Array can store primitive types as well as objects. In ArrayList only Objects can be stored.
- Arrays can be multi-dimensional whereas ArrayList is always unidimensional.
  As Exp. a two dimensional array can be created as –

  Integer myArray[][] = new Integer[4][3];

- Performance wise both Array and ArrayList are almost same as internally ArrayList also uses an Array. But there is overhead of resizing the array and copying the elements to the new Array in case of ArrayList.

---- >> Related Posts Of Above Questions :::

------>>[MOST IMPORTANT]<<------

1. **300+ [LATEST] Java Collections Framework Interview Questions and Answers**
2. **300+ [LATEST] Java.util Interview Questions and Answers**
3. **250+ TOP MCQs on Collections Interface and Answers**
4. **250+ TOP MCQs on Java.util – Array Class and Answers**
5. **300+ [LATEST] Java Inheritance Interview Questions and Answers**
6. **300+ [LATEST] Java Equals And Hashcode Interview Questions and Answers**
7. **300+ [LATEST] Java Generics Interview Questions and Answers**
8. **250+ TOP MCQs on Java.util – LinkedList, HashSet & TreeSet Class and Answers**

9. 250+ TOP MCQs on Java.util – ArrayList Class and Answers
10. 300+ TOP Play Framework Interview Questions & Answers
11. 300+ [LATEST] Java Webdynpro Interview Questions and Answers
12. 250+ TOP MCQs on Data Structures-List and Answers
13. 300+ [LATEST] Java Concurrency Interview Questions and Answers
14. 300+ [LATEST] Java Native Interface (JNI) Interview Questions and Answers
15. 250+ TOP MCQs on Java.util – Dictionary, Hashtable & Properties and Answers
16. 300+ [LATEST] Java Developer Interview Questions and Answers
17. 250+ TOP MCQs on Java.util – Vectors & Stack and Answers
18. 300+ [LATEST] Java Serialization Interview Questions and Answers
19. 300+ [LATEST] Java 9 Interview Questions and Answers
20. 250+ TOP MCQs on Java.util – Maps and Answers