Collection

34 Java Collections Interview Questions

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

1) What is the Collection framework in Java?

Collection Framework is a combination of classes and interface, which is used to store and manipulate the data in the form of objects. It provides various classes such as ArrayList, Vector, Stack, and HashSet, etc. and interfaces such as List, Queue, Set, etc. for this purpose.

2) What are the main differences between array and collection?

Array and Collection are somewhat similar regarding storing the references of objects and manipulating the data, but they differ in many ways. The main differences between the array and Collection are defined below:

* Arrays are always of fixed size, i.e., a user can not increase or decrease the length of the array according to their requirement or at runtime, but In Collection, size can be changed dynamically as per need.
* Arrays can only store homogeneous or similar type objects, but in Collection, heterogeneous objects can be stored.
* Arrays cannot provide the ?ready-made? methods for user requirements as sorting, searching, etc. but Collection includes readymade methods to use.

3) Explain various interfaces used in Collection framework?

Collection framework implements various interfaces, Collection interface and Map interface (java.util.Map) are the mainly used interfaces of Java Collection Framework. List of interfaces of Collection Framework is given below:

**1. Collection interface:** Collection (java.util.Collection) is the primary interface, and every collection must implement this interface.

**Syntax:**

1. **public** **interface** Collection<E>**extends** Iterable

Where <E> represents that this interface is of Generic type

**2. List interface:**List interface extends the Collection interface, and it is an ordered collection of objects. It contains duplicate elements. It also allows random access of elements.

**Syntax:**

1. **public** **interface** List<E> **extends** Collection<E>

**3. Set interface:** Set (java.util.Set) interface is a collection which cannot contain duplicate elements. It can only include inherited methods of Collection interface

**Syntax:**

1. **public** **interface** Set<E> **extends** Collection<E>

**Queue interface:**Queue (java.util.Queue) interface defines queue data structure, which stores the elements in the form FIFO (first in first out).

**Syntax:**

1. **public** **interface** Queue<E> **extends** Collection<E>

**4. Dequeue interface:** it is a double-ended-queue. It allows the insertion and removal of elements from both ends. It implants the properties of both Stack and queue so it can perform LIFO (Last in first out) stack and FIFO (first in first out) queue, operations.

**Syntax:**

1. **public** **interface** Dequeue<E> **extends** Queue<E>

**5. Map interface:**A Map (java.util.Map) represents a key, value pair storage of elements. Map interface does not implement the Collection interface. It can only contain a unique key but can have duplicate elements. There are two interfaces which implement Map in java that are Map interface and Sorted Map.

4) 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. |
| 4) | ArrayList is not ?thread-safe? as it is not synchronized. | Vector list is ?thread-safe? as it?s every method is synchronized. |

## **### What are the Legacy Classes in Java?**

Earlier versions of Java did not include the Collections Framework. Only in from version 1.2, you could actually use this Legacy class. In this,  the original classes were reengineered to support the collection interface. These classes are also known as Legacy classes. All legacy classes and interfaces were redesigned by JDK 5 to support [*Generics*](https://www.edureka.co/blog/generics-in-java/)

5) What is the difference between ArrayList and LinkedList?

|  |  |  |
| --- | --- | --- |
| **No.** | **ArrayList** | **LinkedList** |
| 1) | ArrayList uses a dynamic array. | LinkedList uses a doubly linked list. |
| 2) | ArrayList is not efficient for manipulation because too much is required. | LinkedList is efficient for manipulation. |
| 3) | ArrayList is better to store and fetch data. | LinkedList is better to manipulate data. |
| 4) | ArrayList provides random access. | LinkedList does not provide random access. |
| 5) | ArrayList takes less memory overhead as it stores only object | LinkedList takes more memory overhead, as it stores the object as well as the address of that object. |

6) What is the difference between Iterator and ListIterator?

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

|  |  |  |
| --- | --- | --- |
| **No.** | **Iterator** | **ListIterator** |
| 1) | The Iterator traverses the elements in the forward direction only. | ListIterator traverses the elements in backward and forward directions both. |
| 2) | The Iterator can be used in List, Set, and Queue. | ListIterator can be used in List only. |
| 3) | The Iterator can only perform remove operation while traversing the collection. | ListIterator can perform ?add,? ?remove,? and ?set? operation while traversing the collection. |

7) What is the difference between Iterator and Enumeration?

|  |  |  |
| --- | --- | --- |
| **No.** | **Iterator** | **Enumeration** |
| 1) | The Iterator can traverse legacy and non-legacy elements. | Enumeration can traverse only legacy elements. |
| 2) | The Iterator is fail-fast. | Enumeration is not fail-fast. |
| 3) | The Iterator is slower than Enumeration. | Enumeration is faster than Iterator. |
| 4) | The Iterator can perform remove operation while traversing the collection. | The Enumeration can perform only traverse operation on the collection. |

8) What is the difference between List and Set?

The List and Set both extend the collection interface. However, there are some differences between the both which are listed below.

* The List can contain duplicate elements whereas Set includes unique items.
* The List is an ordered collection which maintains the insertion order whereas Set is an unordered collection which does not preserve the insertion order.
* The List interface contains a single legacy class which is Vector class whereas Set interface does not have any legacy class.
* The List interface can allow n number of null values whereas Set interface only allows a single null value.

9) What is the difference between HashSet and TreeSet?

The HashSet and TreeSet, both classes, implement Set interface. The differences between the both are listed below.

* HashSet maintains no order whereas TreeSet maintains ascending order.
* HashSet implemented by hash table whereas TreeSet implemented by a Tree structure.
* HashSet performs faster than TreeSet.
* HashSet is backed by HashMap whereas TreeSet is backed by TreeMap.

10) What is the difference between Set and Map?

The differences between the Set and Map are given below.

* Set contains values only whereas Map contains key and values both.
* Set contains unique values whereas Map can contain unique Keys with duplicate values.
* Set holds a single number of null value whereas Map can include a single null key with n number of null values.

11) What is the difference between HashSet and HashMap?

The differences between the HashSet and HashMap are listed below.

* HashSet contains only values whereas HashMap includes the entry (key, value). HashSet can be iterated, but HashMap needs to convert into Set to be iterated.
* HashSet implements Set interface whereas HashMap implements the Map interface
* HashSet cannot have any duplicate value whereas HashMap can contain duplicate values with unique keys.
* HashSet contains the only single number of null value whereas HashMap can hold a single null key with n number of null values.

12) What is the difference between HashMap and TreeMap?

The differences between the HashMap and TreeMap are given below.

* HashMap maintains no order, but TreeMap maintains ascending order.
* HashMap is implemented by hash table whereas TreeMap is implemented by a Tree structure.
* HashMap can be sorted by Key or value whereas TreeMap can be sorted by Key.
* HashMap may contain a null key with multiple null values whereas TreeMap cannot hold a null key but can have multiple null values.

13) 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. |
| 3) | HashMap is not ?thread-safe,? so it is useful for non-threaded applications. | Hashtable is thread-safe, and it can be shared between various threads. |
| 4) | 4) HashMap inherits the AbstractMap class | Hashtable inherits the Dictionary class. |

14) What is the difference between Collection and Collections?

The differences between the Collection and Collections are given below.

* The Collection is an interface whereas Collections is a class.
* The Collection interface provides the standard functionality of data structure to List, Set, and Queue. However, Collections class is to sort and synchronize the collection elements.
* The Collection interface provides the methods that can be used for data structure whereas Collections class provides the static methods which can be used for various operation on a collection.

15) What is the difference between Comparable and Comparator?

|  |  |  |
| --- | --- | --- |
| **No.** | **Comparable** | **Comparator** |
| 1) | Comparable provides only one sort of sequence. | The Comparator provides multiple sorts 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 located in java.util package. |
| 4) | If we implement the Comparable interface, The actual class is modified. | The actual class is not changed. |

16) What do you understand by BlockingQueue?

BlockingQueue is an interface which extends the Queue interface. It provides concurrency in the operations like retrieval, insertion, deletion. While retrieval of any element, it waits for the queue to be non-empty. While storing the elements, it waits for the available space. BlockingQueue cannot contain null elements, and implementation of BlockingQueue is thread-safe.

**Syntax:**

1. **public** **interface** BlockingQueue<E> **extends** Queue <E>

17) What is the advantage of Properties file?

If you change the value in the properties file, you don't need to recompile the java class. So, it makes the application easy to manage. It is used to store information which is to be changed frequently. Consider the following example.

1. **import** java.util.\*;
2. **import** java.io.\*;
3. **public** **class** Test {
4. **public** **static** **void** main(String[] args)**throws** Exception{
5. FileReader reader=**new** FileReader("db.properties");
7. Properties p=**new** Properties();
8. p.load(reader);
10. System.out.println(p.getProperty("user"));
11. System.out.println(p.getProperty("password"));
12. }
13. }

**Output**

system

oracle

18) 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 identical.

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

If two objects do not produce an equal result by using the equals() method, then the hashcode() method will provide the different integer result for both the objects.

19) Why we override equals() method?

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

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

20) 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){} |

21) What is the advantage of the generic collection?

There are three main advantages of using the generic collection.

* If we use the generic class, we don't need typecasting.
* It is type-safe and checked at compile time.
* Generic confirms the stability of the code by making it bug detectable at compile time.

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

Two different keys with the same hash value are known as hash-collision. Two separate entries will be kept in a single hash bucket to avoid the collision. There are two ways to avoid hash-collision.

* Separate Chaining
* Open Addressing

23) What is the Dictionary class?

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

24) 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.

25) What do you understand by fail-fast?

The Iterator in java which immediately throws ConcurrentmodificationException, if any structural modification occurs in, is called as a Fail-fast iterator. Fail-fats iterator does not require any extra space in memory.

26) [What is the difference between Array and ArrayList?](https://www.javatpoint.com/array-vs-arraylist-in-java)

The main differences between the Array and ArrayList are given below.

|  |  |  |
| --- | --- | --- |
| **SN** | **Array** | **ArrayList** |
| 1 | The Array is of fixed size, means we cannot resize the array as per need. | ArrayList is not of the fixed size we can change the size dynamically. |
| 2 | Arrays are of the static type. | ArrayList is of dynamic size. |
| 3 | Arrays can store primitive data types as well as objects. | ArrayList cannot store the primitive data types it can only store the objects. |

27) [What is the difference between the length of an Array and size of ArrayList?](https://www.javatpoint.com/difference-between-length-of-array-and-size-of-arraylist-in-java)

The length of an array can be obtained using the property of length whereas ArrayList does not support length property, but we can use size() method to get the number of objects in the list.

**Finding the length of the array**

1. Int [] array = **new** **int**[4];
2. System.out.println("The size of the array is " + array.length);

**Finding the size of the ArrayList**

1. ArrayList<String> list=**new** ArrayList<String>();
2. list.add("ankit");
3. list.add("nippun");
4. System.out.println(list.size());

28) [How to convert ArrayList to Array and Array to ArrayList?](https://www.javatpoint.com/how-to-convert-arraylist-to-array-and-array-to-arraylist-in-java)

We can convert an Array to ArrayList by using the asList() method of Arrays class. asList() method is the static method of Arrays class and accepts the List object. Consider the following syntax:

1. Arrays.asList(item)

We can convert an ArrayList to Array using toArray() method of the ArrayList class. Consider the following syntax to convert the ArrayList to the List object.

1. List\_object.toArray(**new** String[List\_object.size()])

29) [How to make Java ArrayList Read-Only?](https://www.javatpoint.com/how-to-make-java-arraylist-read-only)

We can obtain java ArrayList Read-only by calling the Collections.unmodifiableCollection() method. When we define an ArrayList as Read-only then we cannot perform any modification in the collection through  add(), remove() or set() method.

30) [How to remove duplicates from ArrayList?](https://www.javatpoint.com/how-to-remove-duplicates-from-arraylist-in-java)

There are two ways to remove duplicates from the ArrayList.

* **Using HashSet:** By using HashSet we can remove the duplicate element from the ArrayList, but it will not then preserve the insertion order.
* **Using LinkedHashSet:** We can also maintain the insertion order by using LinkedHashSet instead of HashSet.

The Process to remove duplicate elements from ArrayList using the LinkedHashSet:

* Copy all the elements of ArrayList to LinkedHashSet.
* Empty the ArrayList using clear() method, which will remove all the elements from the list.
* Now copy all the elements of LinkedHashset to ArrayList.

31) [How to reverse ArrayList?](https://www.javatpoint.com/how-to-reverse-arraylist-in-java)

To reverse an ArrayList, we can use reverse() method of Collections class. Consider the following example.

1. **import** java.util.ArrayList;
2. **import** java.util.Collection;
3. **import** java.util.Collections;
4. **import** java.util.Iterator;
5. **import** java.util.List;
6. **public** **class** ReverseArrayList {
7. **public** **static** **void** main(String[] args) {
8. List list = **new** ArrayList<>();
9. list.add(10);
10. list.add(50);
11. list.add(30);
12. Iterator i = list.iterator();
13. System.out.println("printing the list....");
14. **while**(i.hasNext())
15. {
16. System.out.println(i.next());
17. }
18. Iterator i2 = list.iterator();
19. Collections.reverse(list);
20. System.out.println("printing list in reverse order....");
21. **while**(i2.hasNext())
22. {
23. System.out.println(i2.next());
24. }
25. }
26. }

**Output**

printing the list....

10

50

30

printing list in reverse order....

30

50

10

32) [How to sort ArrayList in descending order?](https://www.javatpoint.com/how-to-sort-java-arraylist-in-descending-order)

To sort the ArrayList in descending order, we can use the reverseOrder method of Collections class. Consider the following example.

1. **import** java.util.ArrayList;
2. **import** java.util.Collection;
3. **import** java.util.Collections;
4. **import** java.util.Comparator;
5. **import** java.util.Iterator;
6. **import** java.util.List;
8. **public** **class** ReverseArrayList {
9. **public** **static** **void** main(String[] args) {
10. List list = **new** ArrayList<>();
11. list.add(10);
12. list.add(50);
13. list.add(30);
14. list.add(60);
15. list.add(20);
16. list.add(90);
18. Iterator i = list.iterator();
19. System.out.println("printing the list....");
20. **while**(i.hasNext())
21. {
22. System.out.println(i.next());
23. }
25. Comparator cmp = Collections.reverseOrder();
26. Collections.sort(list,cmp);
27. System.out.println("printing list in descending order....");
28. Iterator i2 = list.iterator();
29. **while**(i2.hasNext())
30. {
31. System.out.println(i2.next());
32. }
34. }
35. }

**Output**

printing the list....

10

50

30

60

20

90

printing list in descending order....

90

60

50

30

20

10

33) [How to synchronize ArrayList?](https://www.javatpoint.com/how-to-synchronize-arraylist-in-java)

We can synchronize ArrayList in two ways.

* Using Collections.synchronizedList() method
* Using CopyOnWriteArrayList<T>

\*\*\*\*\*\*\* What is in insertion order?

Insertion do not follow ascending and descending order that how you add the values is called Insertion / as need.

What is No order?

No order does not follow ascending, descending and insertion order that its resaved randomly values.

#### Order is another name is sorting.

\*\*\* Tri or Tri set or anything is allowed ascending order.

\*\*\*\* Anything List is followed insertion order.

\*\*\*\* Anything has + is no follow any order

Q.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Points | Array | Array list | Linked list | Vector | Stack |
| Null value | Yes | Yes | Yes | Yes | Yes |
| Duplicate | Yes | Yes | Yes | Yes | Yes |
| Advantage | Multiple value | 1. Dynamic 2. Search faster | Add or remove faster than Array list | Synchronize | Last in fast out (LIFO) |
| Disadvantage | Length fixed | 1. Slower than array 2. Add/remove slower than linked list | 1. Search slower than array list | 1. Slow 2. Legacy | Slow |
| Performance | Faster | 1. Slower than array 2. Search faster 3. Add/remove slower | 1. Search slower 2. Add/remove faster | Slow | Slow |
| Add value | Assign | Add() | Add() | Add() | Push() |
| Remove value | Assign | Remove() | Remove() | Remove() | Pop() |
| Total number | Length() | Size() | Size() | Size() | Size() |
| Synchronize (Thread safe) | No | No | No | Yes | Yes |
| Order/Sorting | insertion | insertion | insertion | insertion | Last in fast out (LIFO) |

Q. Deferent between set and map ?

|  |  |  |
| --- | --- | --- |
| Set | Map | Points |
| HashSet | HashMap | Duplicate :  HashSet=No  Hash Map[K(D=no),V(D=yes)]  Null :  Hash Set= One  HashMap[K(null=one),V(D=yes)]  Order :  No order |
| Linked HashSet | Linked HashMap | Duplicate :  Linked HashSet=No  Linked HashMap[K(D=no),V(D=yes)]  Null :  Linked HashSet=One  Linked HashMap[K(null=one),V(D=yes)]  Order :  Insertion order |
| Tree Set | Tree Map | Duplicate :  Tree Set=No  Tree Map[K(D=no),V(D=yes)]  Null :  Tree Set=No  Tree Map[K(null=one),V(D=yes)]    Order :  Ascending order |

Q. Deferent between HashMap and Hash Table ?

|  |  |  |
| --- | --- | --- |
| Point | HashMap | Hash Table |
| Synchronize | No | Yes |
| Thread safe | No | Yes |
| Duplicate | K = No, V = Yes | K = No , V = value |
| Null | K = One , V = Yes | K = No, V = No (Tree) |

Q.

| **Sr. No.** | **Key** | **Stack** | **Heap Memory** |
| --- | --- | --- | --- |
| 1 | Basic | Stack memory is used to store items which have a very short life like local variables, a reference variable of objects | Heap memory is allocated to store objects and JRE classes. |
| 2 | Ordering | The stack is always reserved in a LIFO (last in first out) order | Heap memory is dynamic allocation there is no fixed pattern for allocating and deallocating blocks in memory |
| 3 | Size | We can increase stack memory size by using JVM parameter -XSS | We can increase or decrease heap memory size by using JVM option -Xms and -Xmx |
| 4 | Visibility | Variables are visible to only to owner thread | It is visible to all threads |
| 5 | Exception | JVM will throw java.lang.StackOverFlowError | JVM will throw java.lang.OutOfMemoryError |