Java Interview Question and Answers

➢ Topic – Java ArrayList:

Java ArrayList Interview Questions Answers

---

1. What is ArrayList in Java?
Answer: ArrayList is a resizable array that can grow or shrink in the memory whenever needed. It is dynamically created with an initial capacity. It uses a dynamic array internally for storing a group of elements or data.

2. Explain the hierarchy of ArrayList class in Java.
Answer: Go to this tutorial: *ArrayList in Java*.
3. Which marker interfaces are implemented by Java ArrayList?
Answer: Java ArrayList class implements three marker interfaces such as RandomAccess, Cloneable, and Serializable.

4. Which collection classes implement List interface in Java?
Answer: The collection classes that implement List interface, are as:

- ArrayList
- LinkedList
- CopyOnWriteArrayList
- Vector
- Stack

5. What are the important features of ArrayList in Java?
Answer: There are several significant features of ArrayList in Java that are as follows:

a) ArrayList in Java uses an index-based structure.

b) The size of ArrayList can increase or decrease at runtime. Once ArrayList is created, we can add any number of elements.

c) An ArrayList allows adding elements into the middle of collection.

d) It allows to delete elements.

e) Duplicate elements are allowed in the array list.

f) Any number of null elements can be added to ArrayList.

g) ArrayList maintains the insertion order in Java. That is insertion order is preserved.

h) ArrayList is not synchronized. That means multiple threads can use the same ArrayList objects simultaneously.

i) Since ArrayList implements random access interface, we can get, set, insert, and remove elements of the array list from any arbitrary position.

j) The performance of ArrayList is slow because if any element is removed from ArrayList, a lot of shifting takes place.

---

6. Why is ArrayList called a dynamically growing array in Java?
Answer: ArrayList is called a dynamically growing array in java because ArrayList uses a dynamic array internally for storing a group of elements.

If the initial capacity of the array is exceeded, a new array with a larger capacity is created automatically and all the elements from the current array are copied to the new array.

When elements are removed from the array list, the size of array list can be shrunk automatically.

7. How to create a generic ArrayList object in Java?
Answer: The general syntax to create a generic ArrayList object is as follows:

```
ArrayList<T> arList = new ArrayList<T>(); // Here, T is a type parameter.

For example:

ArrayList<String> arList = new ArrayList<String>();
```

8. How to add elements to Java ArrayList?
Answer: List interface provides three useful methods to add elements into an ArrayList. They are:

- **add(E element)**: Appends the specified element to the end of the array list.

- **add(int index, E element):** Inserts the specified element at the specified position in the list.
- **addAll(int index, Collection<? extends E > c):** Inserts all of the elements of the specified collection into the list, starting at the specified position.

---

9. Does ArrayList allow to insert duplicate elements?
Answer: Yes, ArrayList allows to insert duplicate elements.

10. Is it possible to add null into ArrayList?
Answer: Yes, we can add any number of nulls into the array list.

11. Is it possible to join two or more ArrayLists in Java?
Answer: Yes, we can join two or more ArrayLists in java. List interface provides a method addAll() to join two or more lists in java.

If we have one list list1 and another list2, we can join them with the help of addAll() like this: list1.addAll(list2);

12. What will be the result of the following code snippet?

```
import java.util.ArrayList;

import java.util.List;

public class ArrayListTest {

public static void main(String[] args)

{

  List<String> list = new ArrayList<String>();

   list.add("Dhanbad");

   list.add(0, "New York");

   list.add("Mumbai");

   list.add(2, "Sydney");

   System.out.println(list);
  }

}
```

Answer: Output: [New York, Dhanbad, Sydney, Mumbai]

---

13. What will be the output of the following program?

```
import java.util.ArrayList;

import java.util.List;

public class ArrayListTest {

public static void main(String[] args)

{

   List<String> list = new ArrayList<String>();

    list.add("Dhanbad");

    list.add(0, "New York");

    list.add("Mumbai");

    list.add(3, "Sydney");

    System.out.println(list);

  }

}
```

Answer: Output: [New York, Dhanbad, Mumbai, Sydney]

14. What will be the output of the below code snippet?

```
import java.util.ArrayList;

import java.util.List;
```

```
public class ArrayListTest {

public static void main(String[] args)

{

  List<String> list = new ArrayList<String>();

    list.add("Dhanbad");

    list.add(0, "New York");

    list.add("Mumbai");

    list.add(1, "Sydney");

    System.out.println(list);

   }

}
```

Answer: Output: [New York, Sydney, Dhanbad, Mumbai]

15. What is the output of the following code snippet?

```
import java.util.ArrayList;

import java.util.List;

public class ArrayListTest {

public static void main(String[] args)

{

   List<String> list = new ArrayList<String>();

    list.add("Orange");

    list.add(0, "Banana");
   ArrayList<String> arList = new ArrayList<>();

    arList.add("Apple");

    list.add("Grapes");

    list.addAll(3, arList);

    System.out.println(list);

   }

}
```

Answer: Output: [Banana, Orange, Grapes, Apple]

16. Will the code compile fine? If yes, what will be the output of the below code?

```
import java.util.ArrayList;

import java.util.Collection;

import java.util.List;

public class ArrayListTest {

public static void main(String[] args)

{

   List<String> list = new ArrayList<String>();

     list.add("A");

     list.add("B");

  Collection<String> c = new ArrayList<>();

     c.add("C");

     list.addAll(1, c);
   System.out.println(list);

  }

}
```

Answer: Yes, the code will compile fine. There is no problem with the above code. Output: [A, C, B].

---

17. How many times null elements will be displayed in the following code?

```
import java.util.ArrayList;

public class ArrayListTest {

public static void main(String[] args)

{

  ArrayList<String> list = new ArrayList<String>();

    list.add(null);

    list.add(1, null);

    list.add(1, null);

    list.add(1, null);

    System.out.println(list);

  }

}
```

Answer: null will be printed 4 times because if any element is already present at the specified position, ArrayList shifts that element to the right side. Therefore, Output: [null, null, null, null].

18. Will the below code compile fine? If yes, what will be the output of the following code snippet?

```
import java.util.ArrayList;

public class ArrayListTest {

public static void main(String[] args)

{

   ArrayList<String> list = new ArrayList<String>();

    list.add(null);

    list.add(0, "A");

    list.add(3, "B");

    list.add(1, "C");

    System.out.println(list);

   }

}
```

Answer: Yes, the above code will compile fine but at runtime, JVM will throw IndexOutOfBoundsException.

---

19. What will be the result of the following program?

```
import java.util.ArrayList;

public class ArrayListTest {

public static void main(String[] args)

{

   ArrayList<String> list = new ArrayList<String>();
```

```
    list.add(null);

    list.add(0, "A");

    list.add(2, "B");

    list.add(1, "C");

    System.out.println(list);

  }

}
```

Answer: Output: [A, C, null, B].

20. What is the output of the following code if there is no error in the code?

```
import java.util.ArrayList;

public class ArrayListTest {

public static void main(String[] args)

{

   ArrayList<String> list = new ArrayList<String>();

    list.add(null);

    list.add(0, "A");

    list.add(null);

    list.add(2, "B");

    list.add("20");

    list.add(1, "C");

   System.out.println(list);
```

```
  }

}
```

Answer: Output: [A, C, null, B, null, 20]

21. Why adding or inserting elements to ArrayList can be slow?
Answer: When the size of ArrayList is unknown then adding elements to ArrayList is slow. When the size of ArrayList grows, a lot of shifting takes place in the memory while adding elements. Due to which the performance of ArrayList becomes slow.

22. What is the difference between the length of an array and size of ArrayList?
Answer: The length of an array can be determined by using property length. But ArrayList does not support the length property. It provides size() method that can be used to find the number of elements in the list.

---

23. Suppose we want to add an element in the middle of list. Which list implementation will provide you better performance? ArrayList or LinkedList?
Answer: For the above scenario, LinkedList is a better choice because in the case of LinkedList, when we add an element at the specified position, internally, a node is created and only two links are changed.

But in the case of ArrayList, a lot of shifting is done in the memory when we add an element in the middle of the list or anywhere, except at the end.

So, LinkedList gives faster performance when we add an element in the middle of list.

24. Both ArrayList and LinkedList provide get() method to retrieve an element at the specified position from the list. Which one is faster, ArrayList or LinkedList?
Answer: ArrayList's get() method is faster than LinkedList's get() because LinkedList does not implement Random Access Interface.

Due to which it will traverse from the beginning or ending over the list until it reaches the index specified.

25. What are the advantages of ArrayList over Arrays?
Answer: The advantages of ArrayList over Arrays are as follows:

- ArrayList can grow or shrink dynamically.
- ArrayList provides a more powerful insertion and search mechanism as compared to arrays.

26. What is the main difference between ArrayList and Arrays?
Answer: The main differences between ArrayList and Arrays are as follows:

a) An array is always of fixed length and it cannot be changed at runtime, while ArrayList's size can be changed dynamically.

b) The array is of static type whereas, ArrayList is of dynamic size.

c) In an array, we can store both primitive data types and objects. In the case of ArrayList, we can store only objects, not primitive data types.

27. Can we delete an element from an ArrayList?
Answer: Yes, we can remove an element from an ArrayList using remove() method.

28. What is the output of the below program?

```
import java.util.ArrayList;

public class ArrayListTest {

public static void main(String[] args)

{

ArrayList<String> list = new ArrayList<String>();

  list.add(null);

  list.add(0, "Zero");

  list.add(null);



  list.add(2, "Two");

  list.add("Four");
  list.add(1, "One");

  list.remove(2);

  System.out.println(list);

 }

}
```

Answer: Output: [Zero, One, Two, null, Four]

29. Is it possible to convert an array to ArrayList in java?

Answer: Yes, it is possible to convert an array to ArrayList using asList() method of Arrays class. The asList() method is a static method provided by Arrays class that accepts the List objects.

The sample code is given below:

```
public class ArrayListTest {

public static void main(String[] args)

{

    String[ ] cityArray = {"Dhanbad", "California", "Mumbai", "Paris"};

// Converting an array to ArrayList.

    List<String> cityList = Arrays.asList(cityArray);

    System.out.println(cityList);

  }

}

Output:

        [Dhanbad, California, Mumbai, Paris]
```

30. Can we convert an ArrayList to Array in Java?

Answer: Yes, we can convert an ArrayList to Array in java. ArrayList provides toArray() method that returns an array of elements of the same type as List.

The sample code is given below:

```
public class Test {

public static void main(String[] args)

{

  ArrayList<String> fruitList = new ArrayList<>();

   fruitList.add("Orange");

   fruitList.add("Apple");

   fruitList.add("Banana");



   int size = fruitList.size();

   String[ ] fruitArray = new String[size];

   fruitArray = fruitList.toArray(fruitArray);



// Displaying array of objects.

     for (String str : fruitArray)

          System.out.print(str + " ");

    }

}

Output:

          Orange Apple Banana
```

31. Can we store duplicate elements in an ArrayList? If yes, what are the ways to remove duplicate elements from ArrayList?
Answer: Yes, we can store duplicate elements in ArrayList. There are mainly two ways to remove duplicates from ArrayList. They are as follows:

a) **Using HashSet:** To remove duplicates from ArrayList using HashSet, we first need to convert ArrayList to HashSet. However, in this case, HashSet will not preserve the insertion order.
b) **Using LinkedHashSet:** The use of LinkedHashSet preserves the insertion order while removing duplicates. It can be performed by the following steps:

- First, copy all elements of ArrayList to LinkedHashSet.
- Now, empty the ArrayList by using clear() method. The clear() method will delete all elements from the list and make it empty.
- At last, copy all the elements of LinkedHashSet to ArrayList.

---

32. What is the return type of get() method provided by ArrayList?
Answer: The return type of get() method is Object.

33. What is the output of the following program?

```
import java.util.ArrayList;

import java.util.List;

public class ArrayListTest {

public static void main(String[] args)

{

  List<Integer> list = new ArrayList<>();

    list.add(20);

    list.add(0, 30);

    list.add(2, 40);

    list.add(40);
    Object element = list.get(1);

    System.out.println(element);

  }

}
```

Answer: Output: 20

---

34. Which method of ArrayList is used to replace an element at a particular position with the specified element?

Answer: ArrayList class provides set() method that can be used to replace an element at a specific position with the specified element.

35. Which element will you get when the below code will be executed?

```
import java.util.ArrayList;

import java.util.List;

public class ArrayListTest {

public static void main(String[] args)

{

  List<Integer> list = new ArrayList<>();

   list.add(20);

   list.add(0, 30);

   list.add(2, 40);

   list.add(50);

   list.add(3, 60);

   list.set(1, 80);
  Object element = list.get(4);

  System.out.println(element);

 }

}
```

Answer: While executing the above code, the result will get 50.

36. By default, ArrayList is an ordered collection. Is there any way by which we can sort elements in ArrayList?
Answer: In Java, Collections class provides a static method sort() that can be used to sort elements of ArrayList. The sample code snippet is given below:

```
public class SortingArrayList {

public static void main(String[] args)

{

  List<Integer> list = new ArrayList<>();

   list.add(20);

   list.add(10);

   list.add(5);

   list.add(30);

   list.add(60);

  System.out.println("Original list: " +list);

  Collections.sort(list);

  System.out.println("Sorted list: " +list);

 }

}
```

```
Output:

          Original list: [20, 10, 5, 30, 60]

          Sorted list: [5, 10, 20, 30, 60]
```

37. How to sort elements of ArrayList in descending order?
Answer: Collections.sort() method will sort elements of ArrayList in ascending order. To get elements in descending order, either use comparator or reverseOrder() method of Collections class.

The sample code is given below:

```
public class DescOrderArrayList {

public static void main(String[] args)

{

  List<Integer> list = new ArrayList<>();

    list.add(20);

    list.add(10);

    list.add(5);

    list.add(30);

    list.add(60);

   System.out.println("Original list: " +list);

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

   System.out.println("Sorted list in descending order: " +list);

  }

}

Output:
```

```
Original list: [20, 10, 5, 30, 60]

Sorted list in descending order: [60, 30, 20, 10, 5]
```

38. Is there any way by which we can reverse the elements in ArrayList?
Answer: Collections class provides reverse() method to reverse elements in ArrayList. The sample code is as follows:

```
List<Integer> list = new ArrayList<>();

  list.add(20);

  list.add(10);

  list.add(5);

  list.add(30);

System.out.println("Original list: " +list); // 20, 10, 5, 30

Collections.reverse(list);

System.out.println("Reversed list: " +list); // 30, 5, 10, 20
```

39. In which scenario ArrayList is useful in Java?
Answer: There are the following scenarios where ArrayList can be used. They are as:

- When we want to store duplicate elements.
- When we want to store null elements.
- It is more preferred when getting (retrieval) of the element is more as compared to adding and removing elements.
- When we are not working in the multi-threading environment because ArrayList is non-synchronized.

---

40. What are the ways by which we can iterate over elements of an ArrayList?
Answer: There are five ways by which we can retrieve (or iterate) elements of an array list. They are as:

- Using for loop
- Using Enhanced for loop or Advanced for loop
- Using while Loop
- By using Iterator
- By ListIterator

For more detail, go to this tutorial: *How to iterate ArrayList in Java*.

41. Will the below code snippet compile successfully? If yes, what will be the result of the following program?

a)

```
import java.util.ArrayList;

import java.util.List;

public class ArrayListTest {

public static void main(String[] args)

{

   List<Integer> list = new ArrayList<>();

     list.add(20);

     list.add(10);

     list.add(5);

     list.add(30);

   for(Integer i : list) {

      System.out.println(i);

      list.remove(3);

    }

  }

}
```

Answer: Yes, the above code will compile successfully but at runtime, JVM will throw ConcurrentModificationException because we can not add or remove an element in the ArrayList during Iteration.

Iteration over elements in the ArrayList using Enhanced for loop is fail-fast.

b)

```
import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

public class ArrayListTest {

public static void main(String[] args)

{

   List<String> list = new ArrayList<String>();

    list.add("John");

    list.add("Herry");

    list.add("Ivaan");

    list.add("Deep");

  Iterator<String> itr = list.iterator();

  while(itr.hasNext())

  {

    String str = itr.next();

     if(str.equals("Herry")) {

        itr.remove();

        System.out.println(list);
    }

}}}
```

Answer: Output: [John, Ivaan, Deep]

---

42. Can we iterate over elements of ArrayList in both forward and backward directions?

Answer: Using ListIterator.

43. What is the output of following program after execution?

```
public class ArrayListTest {

public static void main(String[] args)

{

   List<String> list = new ArrayList<String>();

    list.add("John");

    list.add("Herry");

    list.add("Ivaan");

    list.add("Deep");

  ListIterator<String> litr = list.listIterator();

  while(litr.hasNext()) {

     System.out.println(litr.next());

   }

  }

}
```

Answer: Output: John, Herry, Ivaan, Deep.

44. What are the ways by which we can synchronize ArrayList objects in Java?
Answer: There are two ways by which we can get the synchronized version of ArrayList. They are as follows:

- By using Collections.synchronizedList() method
- By using CopyOnWriteArrayList

Test Your Knowledge

---

45. Suppose we have a Student class and have to add its objects in an ArrayList. Now, the list has to be stored on the basis of the studentID of Student class. What will be the following steps involved?
Answer: Go to this tutorial: How to store user-defined class objects in ArrayList.
46. What are concrete classes implementing List interface in Java?
47. Why is it preferred to declare: List list = new ArrayList(); instead of ArrayList arList = new ArrayList();?
48. How to make an ArrayList read-only?
49. How to sort an ArrayList or a list in reverse order?
50. What are collections classes that implement Random Access Interface?
51. How to avoid ConcurrentModificationException during the iteration of a collection?
52. Can we pass List<String> to a method that accepts List<Object>?
53. What is the difference between size and capacity of ArrayList in Java?
Answer: The number of elements present in an ArrayList is called size of ArrayList.

The capability to store elements in an ArrayList is called capacity of ArrayList. The initial default capacity of an ArrayList is 10.

Hope that this tutorial has covered all the important **ArrayList interview questions in Java** with answers. I hope that you will have understood all

answers to ArrayList interview questions and practiced coding problems.
All the best!!!