**********

# Array List

**********

## Definition:

- An ArrayList is basically, a special type of the list that does not have fixed size, i.e., it has stretchable size.
- It can increase and decrease the size automatically, while performing the several Operations.
- ArrayList is a part of the Collection Framework and implements the java List Interface. This class provides the methods to resize the list based on needs.
- It allows having null and duplicate values.
- The Array List is like a vector in java except that it is unsynchronized. It is not thread-safe but faster than vectors.
- It is an ordered collection that can insert elements with help of indexes.
- ArrayList can hold only objects but cannot handle primitive types.

### Syntax for using the Array List:

ArrayList <datatype> array-list-name = new ArrayList<datatype> ();

**Example:**

```
Class Solution {
        public static void main (String [] args) {
                ArrayList <String> arList = new ArrayList<String> ();
                // Add elements to the ArrayList
                arList.add("Java");
                arList.add("Python");
                arList.add("Data Structure and Algorithm");
                arList.add("JavaScript");
                arList.add("PHP");
                arList.add ("Compiler Design");
                System.out.println(arList);              // display the array list here

                System.out.println("Does list contains JAVA?"+arList.contains("JAVA"));
                System.out.println("element that present at the given index:"+arList.get (2));

                // Add Element at the specific index
                arList.add ("MySQL");
                System.out.println(arList);              // display the array list

        }
}
```

**Output:**
[Java, Python, Data Structure and Algorithm, JavaScript, PHP, Compiler Design]

There are several functions to perform the operation:

1). add () - insert elements to arrayList.
add(int index, E element) – Insert the specified elements at specified position in list.
2). remove () - remove all elements from the list.
3). contains () - check if the element presents in the list.
4). get () - return the index of the element.
5). size () - return the size of list.
6). isEmpty () – to check list empty or not.
7). indexOf () - return the index of given element.

Advantages of ArrayList:

1). It is very fast because of its asynchronous behavior.
2). It maintains order which means the new element added directly to the end of the list
3). When ArrayList is exceeded, the size is increased by 50% only.
4). It is dynamic in size

Disadvantages of ArrayList:

1). ArrayList only holds objects but not primitive types.
2). If the element is removed from the array, the whole array is shifted to update the list which leads to memory wastage.
3). Data is arranged in a sequential manner, for larger lists, it required a larger contiguous block of memory.

# Why? ArrayList

## 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.

## 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.