**Java List Interface | Methods, Example**

## List in Java

\* A **list in Java** is a collection for storing elements in sequential order. Sequential order means the first element, followed by the second element, followed by the third element, and so on.

A good realtime example of a list is a line of train bogies on a track: To get to the fourth bogie from the first bogie, we have to go through the second and third bogies in that order.

\* Java list is a sub-interface of the collection interface that is available in java.util package. Sub interface means an interface that extends another interface is called sub interface. Here, the list interface extends the collection interface.

The general declaration of list interface in java is as follows:

List interface in java has four concrete subclasses. They are ArrayList, LinkedList, Vector, and Stack. These four subclasses implements the list interface.

**features of list**

There are the following features of list in java. They are as follows:

1. The list allows storing duplicate elements in Java. JVM differentiates duplicate elements by using ‘index’. Index refers to the position of a certain object in an array. It always starts at zero.

For example, assume that there is a list with size 10. Suppose the first element is ‘a’ at zero index position and the second element is also ‘a’ which is at 9th position. Thus, there are two ‘a’ elements in the list at positions 0 and 9 respectively.

So, JVM will differentiate between both elements in the list based on their numeral position of the index. Therefore, the index is very useful and plays an important role to differentiate objects.

2. In the list, we can add an element at any position.

3. It maintains insertion order. i.e. List can preserve the insertion order by using the index.

4. It allows for storing many null elements.

5. Java list uses a resizable array for its implementation. Resizable means we can increase or decrease the size of the array.

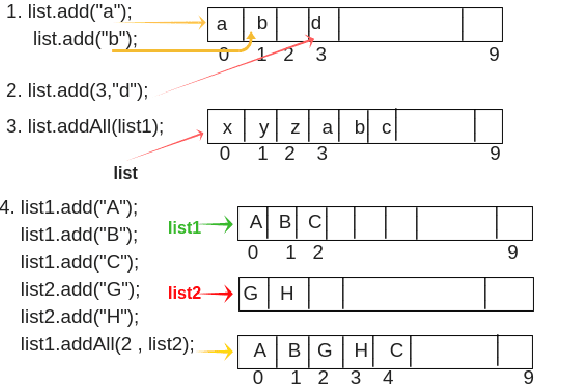
6. Except for LinkedList, ArrayList, and Vector is an indexed-based structure.

7. It provides a special Iterator called a ListIterator that allows accessing the elements in the forward direction using hasNext() and next() methods.

In the reverse direction, it accesses elements using hasPrevious() and previous() methods. We can add, remove elements of the collection, and can also replace the existing elements with the new element using ListIterator.

## Java List Methods

1. **boolean add(Object o):** It starts to add the specified element from zero location. If the element is already present at zero location, it will add the next element in one position.



**2 .void add(int index, Object o):** This method adds/inserts the specified element at a particular position in the list. For example, suppose we want to add element “d” at 3rd position, we will call add(int index, Object o ) method like this:

list.add(3,”d”);

**3.boolean addAll(Collection c):** Here, Collection c represents a group of elements. This method is used to add/insert a group of elements at the end of the last element.

For example, suppose we want to add three elements x, y, and z at positions 0, 1, and 2 respectively in a list.

list1.add("a");

list1.add("b");

list1.add("c");

list.addAll(list1); // It will add group of elements at the end of the last element in the list. The last element is z. So, after z, it will add list1 as shown in above figure.

**4. boolean addAll(int index, Collection c):** This method is used to add/insert a group of elements at a particular position in the list and shift the subsequent elements to the right by increasing their indices.

For example, suppose we want to add three elements at positions 1, 2, and 3 respectively in a list using list1 reference variable.

list1.add("A");

list1.add("B");

list1.add("C");

Now we will create a list of another group of elements using list2 reference variable.

list2.add("G");

list2.add("H");

Assume that we want to add this group of elements at position 2 of the list1. So, we will call addAll(int index, Collection c) method like this:

list1.addAll(2, list2); // You will see that element C is shifted to right at position 4 as shown in the figure.

**5. object remove(int index):** It is used to remove an element at a specified position in the list. For example, consider the above figure.

list1.remove(2); // It will remove the element at 2nd position and element D which is at 3rd position, will be shifted to the left at the 2nd position. The output will be A, B, D.

**6. object get(int index):** This method is used to return element/object stored at a specified position in the list. The return type of get() method is Object and input type is int[index of List].

For example:

list1.get(2); // Output will be 'C'.

**7. int indexOf(Object o):** It is used to return the index of a particular element of the first occurrence in the list. If the element is not present in the list then it will return -1. It takes as an argument as an element and returns as an integer value of that element as it is index value.

For example, suppose an element ‘A’ is present at position 0 and the same element is also present at position 9 in the list.

list.indexOf("A"); // It will return integer value of an element "A" of first occurrence. i.e from zero position, not from position 9. So the output is 0.

**8. int lastIndexOf(Object o):** It returns the index of the last occurrence of a specified element in the list. If the list does not contain that particular element, it will return -1.

For example:

list.lastIndexOf("A"); // Output will be 9.

**9. object set(int index, Object o):** This method replaces the existing element at the specified position in the list with new specified element.

For example:

list1.set(2, "Z");

**10. ListIterator listIterator():** It returns listIterator of the elements in the list in a proper sequence.

Let’s create an example program where we will add both integer and string elements together. So, we will not use generic in this program.

**Program  source  code  1:**

package listPrograms;

import java.util.ArrayList;

import java.util.List;

public class AddEx

{

public static void main(String[] args)

{

// Create a List.

List al = new ArrayList(); // Here, there is no use of generic. So, no type safety. We can add both integer and string elements.

// Adding elements using add() method with reference variable al.

al.add(10);

al.add(20);

al.add(30);

al.add(40);

al.add("Shubh");

// Adding element to 4th position.

al.add(4, 35);

// Adding element to 5th position.

al.add(5, 45);

System.out.println("Elements after adding: " +al);

}

}