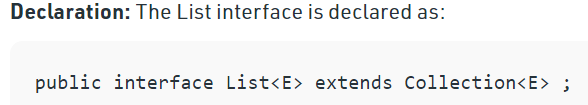
**List(I)**

The List interface in Java provides a way to store the ordered collection. It is a child interface of Collection. It is an ordered collection of objects in which duplicate values can be stored. Since List preserves the insertion order, it allows positional access and insertion of elements.

The List interface is found in **java. util** package and inherits the Collection interface. It is a factory of the **ListIterator interface**. Through the **ListIterator**, we can iterate the list in forward and backward directions. The implementation classes of the List interface are ***ArrayList, LinkedList, Stack, and Vector***. ArrayList and LinkedList are widely used in Java programming.

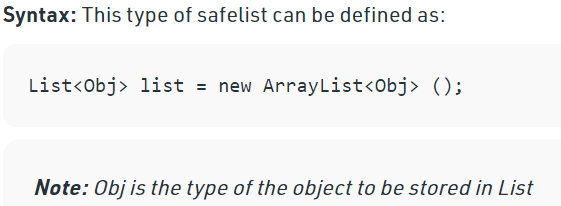
NOTE: **The Vector class is deprecated since Java 5.**





Let us elaborate on creating objects or instances in a List class. Since List is an interface, objects cannot be created of the type list. We always need a class that implements this List to create an object. And also, after the introduction of Generics in Java 1.5, it is possible to restrict the type of object that can be stored in the List.

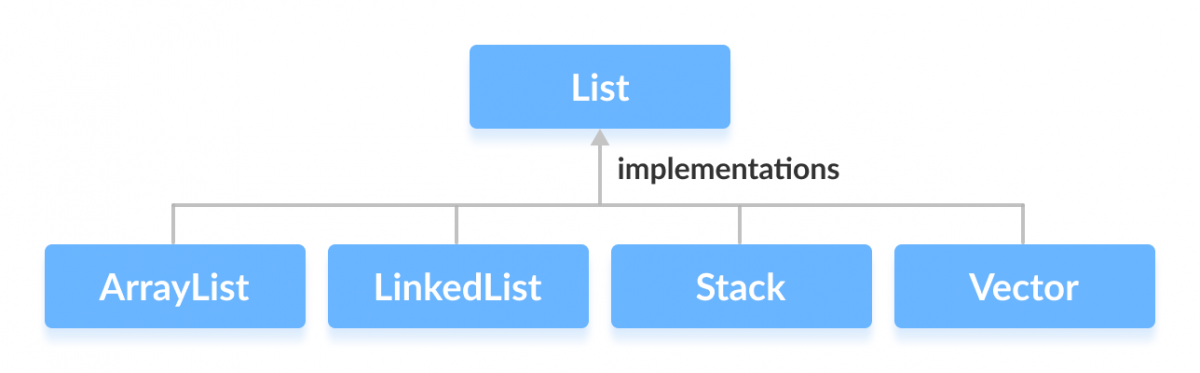
Just like several other user-defined ‘interfaces’ implemented by user-defined ‘classes’, List is an ‘interface’, implemented by the ArrayList class, pre-defined in the java.util package.



**Below are the Methods of the List Interface**

|  |  |
| --- | --- |
| Method | Description |
| **add(int index, element)** | This method is used to add an element at a particular index in the list. When a single parameter is passed, it simply adds the element at the end of the list. |
| addAll(int index, Collection collection) | This method is used to add all the elements in the given collection to the list. When a single parameter is passed, it adds all the elements of the given collection at the end of the list. |
| size() | This method is used to return the size of the list. |
| clear() | This method is used to remove all the elements in the list. However, the reference of the list created is still stored. |
| remove(int index) | This method removes an element from the specified index. It shifts subsequent elements (if any) to left and decreases their indexes by 1. |
| remove(element) | This method is used to remove the first occurrence of the given element in the list. |
| get(index) | This method returns elements at the specified index. |
| set(int index,element) | This method replaces elements at a given index with the new element. This function returns the element which was just replaced by a new element. |
| indexOf(elemet) | This method returns the first occurrence of a given element or *-1* if the element is not present in the list. |
| lastIndexOf(elemet) | This method returns the last occurrence of the given element or *-1* if the element is not present in the list. |
| equals(element) | This method is used to compare the equality of the given element with the elements of the list. |
| hashcode() | This method is used to return the hashcode value of the given list. |
| isEmpty() | This method is used to check if the list is empty or not. It returns true if the list is empty, else false. |
| contains(element) | This method is used to check if the list contains the given element or not. It returns true if the list contains the element. |
| containsAll(Collection c) | This method is used to check if the list contains all the collection of elements. |
| sort(Comparator comp) | This method is used to sort the elements of the list on the basis of the given comparator. |

The implementation class of List(I):



1. **Array List**:

The ArrayList class extends AbstractList and implements the List interface. ArrayList is a generic class that has this declaration:

class ArrayList<E>

Here, E specifies the type of objects that the list will hold

* 1. The underlying data structure is a resizable or growable array.
  2. Duplicates are allowed.
  3. Insertion order is preserved.
  4. Heterogeneous objects are allowed (Except tree set and tree map every were heterogeneous are allowed).
  5. Null insertion is possible.
  6. Array list implements serializable cloneable and Random-access interface.
  7. Array List is the best choice if are frequent operation is retrieval.
  8. Array List is the best choice ie frequent operation is insertion and deletion in the middle.

ArrayList list = new ArrayList();

Creates an empty array List object with default initial capacity 10. Once, arraylist reaches its max capacity a new ArrayList object will be created with by using below formula

newCapacity = (currentCapacity \* 3/2) +1; // CurrentCapacity = 10;

ArrayList list = new ArrayList(int initial capacity);

ArrayList list = new ArrayList(20);

Creates an empty array List object with specified initial capacity.

newCapacity = (currentCapacity \* 3/2) +1; // CurrentCapacity = 20;

ArrayList list = new ArrayList(Collection c);

Creates an equivalent array list object for a given collection.

Example of ArrayList:

// Java program to demonstrate the

// creation of list object using the

// Vector class

import java.io.\*;

import java.util.\*;

class GFG {

    public static void main(String[] args)

    {

        // Size of the vector

        int n = 5;

        // Declaring the List with initial size n

        List<Integer> v = new Vector<Integer>(n);

        // Appending the new elements

        // at the end of the list

        for (int i = 1; i <= n; i++)

            v.add(i);

        // Printing elements

        System.out.println(v);

        // Remove element at index 3

        v.remove(3);

        // Displaying the list after deletion

        System.out.println(v);

        // Printing elements one by one

        for (int i = 0; i < v.size(); i++)

            System.out.print(v.get(i) + " ");

    }

}

**Output**

[1, 2, 3, 4, 5]

[1, 2, 3, 5]

1 2 3 5

From v1.5 Generics have been added to maintain Strict Data type (Type safety)

Important points to remember:

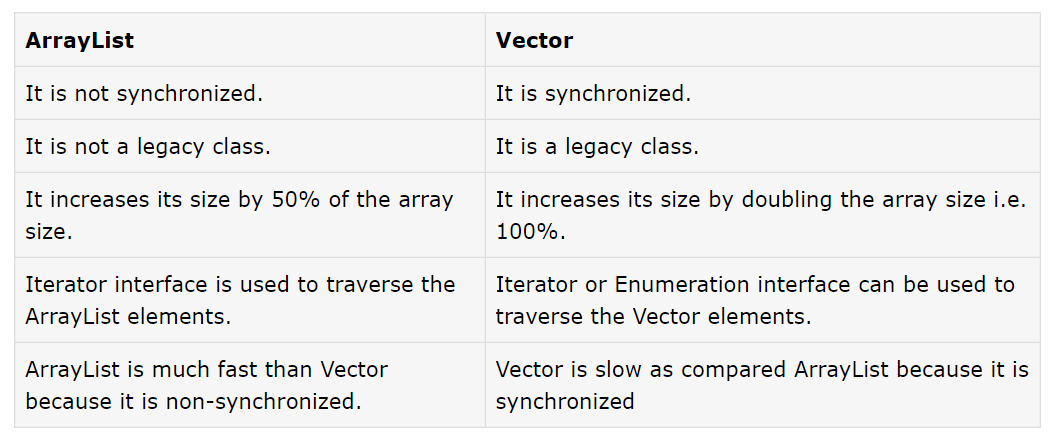
1. Usually, we can use collections to hold and transfer objects from one location to another location (Container), to provide support for this requirement by default every collection class implements Serializable and cloneable interface.
2. Array List and Vector classes implement **randomAccessInterface** 🡪 so that any random element can be accessed with the same speed.
3. RandomAcess Interface present in java.util package and it does not contain any methods it’s a marker interface, where required ability will be provided automatically by the JVM.
4. Array List is the **best choice** if are frequent operation is retrieval operation because ArrayList implements Random access interface.
5. Array List is **a bad choice** if are frequent operation is insertion or deletion at the middle.
6. Synchronized version of Array List:

By default, Array List is non-synchronized, but we can get synchronized version of Array List by using Collections.synchronizedList(List I).

### Methods in Java ArrayList

| Method | Description |
| --- | --- |
| [add(int index, Object element)](https://www.geeksforgeeks.org/java-util-arraylist-add-method-java/) | This method is used to insert a specific element at a specific position index in a list. |
| [add(Object o)](https://www.geeksforgeeks.org/java-util-arraylist-add-method-java/) | This method is used to append a specific element to the end of a list. |
| [addAll(Collection C)](https://www.geeksforgeeks.org/java-util-arraylist-addall-method-java/) | This method is used to append all the elements from a specific collection to the end of the mentioned list, in such an order that the values are returned by the specified collection’s iterator. |
| [addAll(int index, Collection C)](https://www.geeksforgeeks.org/java-util-arraylist-addall-method-java/) | Used to insert all of the elements starting at the specified position from a specific collection into the mentioned list. |
| [clear()](https://www.geeksforgeeks.org/arraylist-clear-java-examples/) | This method is used to remove all the elements from any list. |
| [clone()](https://www.geeksforgeeks.org/clone-method-in-java-2/) | This method is used to return a shallow copy of an ArrayList. |
| [contains?(Object o)](https://www.geeksforgeeks.org/arraylist-contains-java/) | Returns true if this list contains the specified element. |
| [ensureCapacity?(int minCapacity)](https://www.geeksforgeeks.org/arraylist-ensurecapacity-method-in-java-with-examples/) | Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument. |
| [forEach?(Consumer<? super E> action)](https://www.geeksforgeeks.org/arraylist-foreach-method-in-java/) | Performs the given action for each element of the Iterable until all elements have been processed or the action throws an exception. |
| [get?(int index)](https://www.geeksforgeeks.org/arraylist-get-method-java-examples/) | Returns the element at the specified position in this list. |
| [indexOf(Object O)](https://www.geeksforgeeks.org/java-util-arraylist-indexof-java/) | The index the first occurrence of a specific element is either returned, or -1 in case the element is not in the list. |
| [isEmpty?()](https://www.geeksforgeeks.org/arraylist-isempty-java-example/) | Returns true if this list contains no elements. |
| [lastIndexOf(Object O)](https://www.geeksforgeeks.org/arraylist-lastindexof-java-example/) | The index of the last occurrence of a specific element is either returned or -1 in case the element is not in the list. |
| [listIterator?()](https://www.geeksforgeeks.org/arraylist-listiterator-method-in-java-with-examples/) | Returns a list iterator over the elements in this list (in proper sequence). |
| [listIterator?(int index)](https://www.geeksforgeeks.org/arraylist-listiterator-method-in-java-with-examples/) | Returns a list iterator over the elements in this list (in proper sequence), starting at the specified position in the list. |
| [remove?(int index)](https://www.geeksforgeeks.org/arraylist-linkedlist-remove-methods-java-examples/) | Removes the element at the specified position in this list. |
| [remove?(Object o)](https://www.geeksforgeeks.org/arraylist-linkedlist-remove-methods-java-examples/) | Removes the first occurrence of the specified element from this list, if it is present. |
| [removeAll?(Collection c)](https://www.geeksforgeeks.org/arraylist-removeall-method-in-java-with-examples/) | Removes from this list all of its elements that are contained in the specified collection. |
| [removeIf?(Predicate filter)](https://www.geeksforgeeks.org/arraylist-removeif-method-in-java/) | Removes all of the elements of this collection that satisfy the given predicate. |
| [removeRange?(int fromIndex, int toIndex)](https://www.geeksforgeeks.org/arraylist-removerange-java-examples/) | Removes from this list all of the elements whose index is between fromIndex, inclusive, and toIndex, exclusive. |
| [retainAll?(Collection<?> c)](https://www.geeksforgeeks.org/arraylist-retainall-method-in-java/) | Retains only the elements in this list that are contained in the specified collection. |
| [set?(int index, E element)](https://www.geeksforgeeks.org/arraylist-set-method-in-java-with-examples/) | Replaces the element at the specified position in this list with the specified element. |
| [size?()](https://www.geeksforgeeks.org/arraylist-size-method-in-java-with-examples/) | Returns the number of elements in this list. |
| [spliterator?()](https://www.geeksforgeeks.org/arraylist-spliterator-method-in-java/) | Creates a late-binding and fail-fast Spliterator over the elements in this list. |
| [subList?(int fromIndex, int toIndex)](https://www.geeksforgeeks.org/arraylist-sublist-method-in-java-with-examples/) | Returns a view of the portion of this list between the specified fromIndex, inclusive, and toIndex, exclusive. |
| [toArray()](https://www.geeksforgeeks.org/arraylist-array-conversion-java-toarray-methods/) | This method is used to return an array containing all of the elements in the list in the correct order. |
| [toArray(Object[] O)](https://www.geeksforgeeks.org/arraylist-array-conversion-java-toarray-methods/) | It is also used to return an array containing all of the elements in this list in the correct order same as the previous method. |
| [trimToSize()](https://www.geeksforgeeks.org/arraylist-trimtosize-java-example/) | This method is used to trim the capacity of the instance of the ArrayList to the list’s current size. |

Difference between Array List and Vector



Other than this there are no much difference and Vectors are deprecated from version java 1.5.

1. Linked List

The LinkedList class extends AbstractSequentialList and implements the List, Deque, and Queue interfaces. It provides a linked-list data structure. This class is an implementation of the LinkedList data structure which is a linear data structure where the elements are not stored in contiguous locations and every element is a separate object with a data part and address part. The elements are linked using pointers and addresses. Each element is known as a node. Due to the dynamicity and ease of insertions and deletions, they are preferred over the arrays. It also has a few disadvantages like the nodes cannot be accessed directly instead we need to start from the head and follow through the link to reach a node we wish to access. LinkedList is a generic class that has this declaration:

class LinkedList<E>

Here, E specifies the type of objects that the list will hold.

Important points about Linked List:

1. The underlying data structure is doubly Linked list.
2. Duplicates are allowed.
3. Insertion order is preserved.
4. Heterogeneous objects are allowed (Except tree set and tree map every were heterogeneous are allowed).
5. Null insertion is possible.
6. Linked list implements serializable cloneable but not Random access.
7. Linked List is a best choice if are frequent operation is insertion and deletion in the middle.
8. Linked List is worst choice if are frequent operation is retrieval.

**Constructors of Linked List:**

1. **LinkedList():** This constructor is used to create an **empty linked** list. If we wish to create an empty LinkedList with the name ll, then, it can be created as:

LinkedList ll = new LinkedList();

1. **LinkedList(Collection C):** This constructor is used to create an **ordered list that contains all the elements of a specified collection**, as returned by the collection’s iterator. If we wish to create a LinkedList with the name ll, then, it can be created as:

LinkedList ll = new LinkedList(C);

Methods of LinkedList:

| Method | Description |
| --- | --- |
| [add(int index, E element)](https://www.geeksforgeeks.org/java-util-linkedlist-add-method-in-java/) | This method Inserts the specified element at the specified position in this list. |
| [add(E e)](https://www.geeksforgeeks.org/java-util-linkedlist-add-method-in-java/) | This method Appends the specified element to the **end of this list.** |
| [addAll(int index, Collection<E> c)](https://www.geeksforgeeks.org/java-util-linkedlist-addall-method-in-java/) | This method Inserts all of the elements in the specified collection into this list, starting at the specified position. |
| [addAll(Collection<E> c)](https://www.geeksforgeeks.org/java-util-linkedlist-addall-method-in-java/) | This method Appends all of the elements in the specified collection to the **end of this list**, in the order that they are returned by the specified collection’s iterator. |
| [addFirst(E e)](https://www.geeksforgeeks.org/linkedlist-addfirst-method-in-java/) | This method Inserts the specified element at the beginning of this list. |
| [addLast(E e)](https://www.geeksforgeeks.org/linkedlist-addlast-method-in-java/) | This method Appends the specified element to the end of this list. |
| [clear()](https://www.geeksforgeeks.org/linkedlist-clear-method-in-java/) | This method removes all of the elements from this list. |
| [clone()](https://www.geeksforgeeks.org/linkedlist-clone-method-in-java/) | This method returns a **shallow copy** of this LinkedList. |
| [contains(Object o)](https://www.geeksforgeeks.org/linkedlist-contains-method-in-java/) | This method returns true if this list contains the specified element. |
| [descendingIterator()](https://www.geeksforgeeks.org/linkedlist-descendingiterator-method-in-java-with-examples/) | This method returns an iterator over the elements in this deque in reverse sequential order. |
| [element()](https://www.geeksforgeeks.org/linkedlist-element-method-in-java-with-%20examples/) | This method retrieves but does not remove, the head (first element) of this list. |
| [get(int index)](https://www.geeksforgeeks.org/linkedlist-get-method-in-java/) | This method returns the element at the specified position in this list. |
| [getFirst()](https://www.geeksforgeeks.org/java-util-linkedlist-get-getfirst-getlast-java/) | This method returns the first element in this list. |
| [getLast()](https://www.geeksforgeeks.org/linkedlist-getlast-method-in-java/) | This method returns the last element in this list. |
| [indexOf(Object o)](https://www.geeksforgeeks.org/linkedlist-indexof-method-in-java/) | This method returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element. |
| [lastIndexOf(Object o)](https://www.geeksforgeeks.org/linkedlist-lastindexof-method-in-java/) | This method returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element. |
| [listIterator(int index)](https://www.geeksforgeeks.org/linkedlist-listiterator-method-in-java/) | This method returns a list-iterator of the elements in this list (in proper sequence), starting at the specified position in the list. |
| [offer(E e)](https://www.geeksforgeeks.org/java-util-linkedlist-offer-offerfirst-offerlast-java/) | This method Adds the specified element as the tail (last element) of this list. |
| [offerFirst(E e)](https://www.geeksforgeeks.org/java-util-linkedlist-offer-offerfirst-offerlast-java/) | This method Inserts the specified element at the front of this list. |
| [offerLast(E e)](https://www.geeksforgeeks.org/java-util-linkedlist-offer-offerfirst-offerlast-java/) | This method Inserts the specified element at the end of this list. |
| [peek()](https://www.geeksforgeeks.org/java-util-linkedlist-peek-peekfirst-peeklast-java/) | This method retrieves but does not remove, the head (first element) of this list. |
| [peekFirst()](https://www.geeksforgeeks.org/java-util-linkedlist-peek-peekfirst-peeklast-java/) | This **method retrieves, but does not remove**, the first element of this list, or returns null if this list is empty. |
| [peekLast()](https://www.geeksforgeeks.org/java-util-linkedlist-peek-peekfirst-peeklast-java/) | This **method retrieves, but does not remove**, the last element of this list, or returns null if this list is empty. |
| [poll()](https://www.geeksforgeeks.org/java-util-linkedlist-poll-pollfirst-polllast-%20examples-java/) | This **method retrieves and removes the head** (first element) of this list. |
| [pollFirst()](https://www.geeksforgeeks.org/java-util-linkedlist-poll-pollfirst-polllast-%20examples-java/) | This **method retrieves and removes the first** element of this list, or returns null if this list is empty. |
| [pollLast()](https://www.geeksforgeeks.org/java-util-linkedlist-poll-pollfirst-polllast-%20examples-java/) | This **method retrieves and removes the last element** of this list, or returns null if this list is empty. |
| [pop()](https://www.geeksforgeeks.org/linkedlist-pop-method-in-java/) | This method Pops an element from the stack represented by this list. |
| [push(E e)](https://www.geeksforgeeks.org/linkedlist-push-method-in-java/) | This method pushes an element onto the stack represented by this list. |
| [remove()](https://www.geeksforgeeks.org/linkedlist-remove-method-in-java/) | This method retrieves and removes the head (first element) of this list. |
| [remove(int index)](https://www.geeksforgeeks.org/linkedlist-remove-method-in-java/) | This method removes the element at the specified position in this list. |
| [remove(Object o)](https://www.geeksforgeeks.org/linkedlist-remove-method-in-java/) | This method removes the first occurrence of the specified element from this list if it is present. |
| [removeFirst()](https://www.geeksforgeeks.org/linkedlist-removefirst-method-in-java/) | This method removes and returns the first element from this list. |
| [removeFirstOccurrence(Object o)](https://www.geeksforgeeks.org/linkedlist-removefirstoccurrence-method-in-%20java/) | This method removes the first occurrence of the specified element in this list (when traversing the list from head to tail). |
| [removeLast()](https://www.geeksforgeeks.org/linkedlist-removelast-method-in-java/) | This method removes and returns the last element from this list. |
| [removeLastOccurrence(Object o)](https://www.geeksforgeeks.org/linkedlist-removelastoccurrence-method-in-java-with-example/) | This method removes the last occurrence of the specified element in this list (when traversing the list from head to tail). |
| [set(int index, E element)](https://www.geeksforgeeks.org/linkedlist-set-method-in-java/) | This method replaces the element at the specified position in this list with the specified element. |
| [size()](https://www.geeksforgeeks.org/linkedlist-size-method-in-java/) | This method returns the number of elements in this list. |
| [spliterator()](https://www.geeksforgeeks.org/linkedlist-spliterator-method-in-java/) | This method creates a late-binding and fail-fast Spliterator over the elements in this list. |
| toArray() | This method returns an array containing all of the elements in this list in proper sequence (from first to last element). |
| toArray(T[] a) | This method returns an array containing all of the elements in this list in proper sequence (from first to last element); the runtime type of the returned array is that of the specified array. |
| toString() | This method returns a string containing all of the elements in this list in proper sequence (from first to the last element), each element is separated by commas and the String is enclosed in square brackets. |