# JAVA - THE ARRAYLIST CLASS

http://www.tutorialspoint.com/java/java arraylist class.htm

Copyright © tutorialspoint.com

The ArrayList class extends AbstractList and implements the List interface. ArrayList supports dynamic arrays that can grow as needed.

Standard Java arrays are of a fixed length. After arrays are created, they cannot grow or shrink, which means that you must know in advance how many elements an array will hold.

Array lists are created with an initial size. When this size is exceeded, the collection is automatically enlarged. When objects are removed, the array may be shrunk.

Below given is the list of the constructors provided by the ArrayList class.

### **SN** Constructors and Description

### 1 ArrayList

This constructor builds an empty array list.

### 2 **ArrayList**Collectionc

This constructor builds an array list that is initialized with the elements of the collection c.

### 3 **ArrayList**intcapacity

This constructor builds an array list that has the specified initial capacity. The capacity is the size of the underlying array that is used to store the elements. The capacity grows automatically as elements are added to an array list.

Apart from the methods inherited from its parent classes, ArrayList defines following methods:

### **SN** Methods with Description

### 1 **void add**intindex, Objectelement

Inserts the specified element at the specified position index in this list. Throws IndexOutOfBoundsException if the specified index is is out of range  $index < 0 \mid index > size()$ .

### 2 **boolean add**Objecto

Appends the specified element to the end of this list

### 3 **boolean addAll**Collectionc

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. Throws NullPointerException if the specified collection is null.

### 4 **boolean addAll**intindex, Collectionc

Inserts all of the elements in the specified collection into this list, starting at the specified position. Throws NullPointerException if the specified collection is null

.

#### 5 void clear

Removes all of the elements from this list.

### 6 **Object clone**

Returns a shallow copy of this ArrayList.

### 7 **boolean contains***Objecto*

Returns true if this list contains the specified element. More formally, returns true if and only if this list contains at least one element e such that o == null?e == null?o. equals(e).

### 8 **void ensureCapacity** *intminCapacity*

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.

### 9 **Object getintindex**

Returns the element at the specified position in this list. Throws IndexOutOfBoundsException if the specified index is is out of range  $index < 0 \mid index >= size()$ .

### 10 int indexOfObjecto

Returns the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element.

### 11 int lastIndexOfObjecto

Returns the index in this list of the last occurrence of the specified element, or -1 if the list does not contain this element.

### 12 **Object remove**intindex

Removes the element at the specified position in this list. Throws IndexOutOfBoundsException if index out of range  $index < 0 \mid |index> = size()$ .

### protected void removeRangeintfromIndex, inttoIndex

Removes from this List all of the elements whose index is between fromIndex, inclusive and toIndex, exclusive.

### 14 **Object set***intindex*, *Objectelement*

Replaces the element at the specified position in this list with the specified element. Throws IndexOutOfBoundsException if the specified index is is out of range  $index < 0 \mid index >= size()$ .

### 15 int size

Returns the number of elements in this list.

### 16 **Object[] toArray**

Returns an array containing all of the elements in this list in the correct order. Throws NullPointerException if the specified array is null.

### 17 **Object[] toArray**Object[]a

Returns an array containing all of the elements in this list in the correct order; the runtime type of the returned array is that of the specified array.

### 18 void trimToSize

Trims the capacity of this ArrayList instance to be the list's current size.

## **Example:**

The following program illustrates several of the methods supported by ArrayList:

```
import java.util.*;
public class ArrayListDemo {
   public static void main(String args[]) {
      // create an array list
      ArrayList al = new ArrayList();
      System.out.println("Initial size of al: " + al.size());
      // add elements to the array list
      al.add("C");
      al.add("A");
      al.add("E");
      al.add("B");
      al.add("D");
      al.add("F");
      al.add(1, "A2");
      System.out.println("Size of al after additions: " + al.size());
      // display the array list
      System.out.println("Contents of al: " + al);
      // Remove elements from the array list
      al.remove("F");
      al.remove(2);
      System.out.println("Size of al after deletions: " + al.size());
      System.out.println("Contents of al: " + al);
   }
}
```

This would produce the following result:

```
Initial size of al: 0
Size of al after additions: 7
Contents of al: [C, A2, A, E, B, D, F]
Size of al after deletions: 5
Contents of al: [C, A2, F, B, D]
Loading [MathJax]/jax/output/HTML-CSS/jax.js
```