

ArrayList in Java With Example Programs

BY CHAITANYA SINGH | FILED UNDER: [JAVA COLLECTIONS](#)

ArrayList class implements List interface. It is widely used because of the functionality and flexibility it offers. Most of the developers **choose ArrayList over Array** as it's a very good alternative of traditional java arrays. ArrayList is a resizable-array implementation of the List interface. It implements all optional list operations, and permits all elements, including null.

The issue with arrays is that they are of fixed length so if it is full you cannot add any more elements to it, likewise if there are number of elements gets removed from it the memory consumption would be the same as it doesn't shrink. On the other ArrayList can dynamically grow and shrink after addition and removal of elements. Apart from these benefits ArrayList class enables us to use predefined methods of it which makes our task easy. Let's see the ArrayList example first then we will discuss it's methods and their usage.

There is a list of several ArrayList tutorials at the end of this guide, refer it to understand ArrayList concept fully.

ArrayList Example in Java

```
import java.util.*;

public class ArrayListExample {
    public static void main(String args[]) {
        /*Creation of ArrayList: I'm going to add String
         *elements so I made it of string type */
        ArrayList<String> obj = new ArrayList<String>();

        /*This is how elements should be added to the array list*/
        obj.add("Ajeet");
        obj.add("Harry");
        obj.add("Chaitanya");
        obj.add("Steve");
        obj.add("Anuj");

        /* Displaying array list elements */
        System.out.println("Currently the array list has following
elements:"+obj);

        /*Add element at the given index*/
        obj.add(0, "Rahul");
        obj.add(1, "Justin");

        /*Remove elements from array list like this*/
        obj.remove("Chaitanya");
        obj.remove("Harry");

        System.out.println("Current array list is:"+obj);
    }
}
```

```
        /*Remove element from the given index*/
        obj.remove(1);

        System.out.println("Current array list is:"+obj);
    }
}
```

Output:

```
Currently the array list has following elements:[Ajeet, Harry, Chaitanya, Steve,
Anuj]
Current array list is:[Rahul, Justin, Ajeet, Steve, Anuj]
Current array list is:[Rahul, Ajeet, Steve, Anuj]
```

Methods of ArrayList class

In the above example we have used methods such as add and remove. However there are number of methods available which can be used directly using object of ArrayList class. Let's discuss few of the important methods.

1) add(Object o): This method adds an object o to the arraylist.

```
obj.add("hello");
```

This statement would add a string hello in the arraylist at last position.

2) add(int index, Object o): It adds the object o to the array list at the given index.

```
obj.add(2, "bye");
```

It will add the string bye to the 2nd index (3rd position as the array list starts with index 0) of array list.

3) remove(Object o): Removes the object o from the ArrayList.

```
obj.remove("Chaitanya");
```

This statement will remove the string “Chaitanya” from the ArrayList.

4) remove(int index): Removes element from a given index.

```
obj.remove(3);
```

It would remove the element of index 3 (4th element of the list – List starts with o).

5) set(int index, Object o): Used for updating an element. It replaces the element present at the specified index with the object o.

```
obj.set(2, "Tom");
```

It would replace the 3rd element (index =2 is 3rd element) with the value Tom.

6) **int indexOf(Object o)**: Gives the index of the object o. If the element is not found in the list then this method returns the value -1.

```
int pos = obj.indexOf("Tom");
```

This would give the index (position) of the string Tom in the list.

7) **Object get(int index)**: It returns the object of list which is present at the specified index.

```
String str= obj.get(2);
```

Function get would return the string stored at 3rd position (index 2) and would be assigned to the string "str". We have stored the returned value in string variable because in our example we have defined the ArrayList is of String type. If you are having integer array list then the returned value should be stored in an integer variable.

8) **int size()**: It gives the size of the ArrayList – Number of elements of the list.

```
int numberofitems = obj.size();
```

9) **boolean contains(Object o)**: It checks whether the given object o is present in the array list if its there then it returns true else it returns false.

```
obj.contains("Steve");
```

It would return true if the string "Steve" is present in the list else we would get false.

10) **clear()**: It is used for removing all the elements of the array list in one go. The below code will remove all the elements of ArrayList whose object is obj.

```
obj.clear();
```

How to initialize an ArrayList

BY CHAITANYA SINGH | FILED UNDER: [JAVA COLLECTIONS](#)

In the last post we discussed about class [ArrayList in Java](#) and it's important methods. Here we are sharing multiple ways to initialize an ArrayList with examples.

Method 1: Initialization using Arrays.asList

Syntax:

```
ArrayList<Type> obj = new ArrayList<Type>(
    Arrays.asList(Object o1, Object o2, Object o3, ....so on));
```

Example:

```
import java.util.*;
public class InitializationExample1 {
    public static void main(String args[]) {
        ArrayList<String> obj = new ArrayList<String>(
            Arrays.asList("Pratap", "Peter", "Harsh"));
        System.out.println("Elements are:"+obj);
    }
}
```

Output:

```
Elements are:[Pratap, Peter, Harsh]
```

Method 2: Anonymous inner class method to initialize ArrayList

Syntax:

```
ArrayList<T> obj = new ArrayList<T>(){{
    add(Object o1);
    add(Object o2);
    add(Object o3);
    ...
    ...
}};
```

Example:

```
import java.util.*;
public class InitializationExample2 {
    public static void main(String args[]) {
        ArrayList<String> cities = new ArrayList<String>(){{
            add("Delhi");
            add("Agra");
            add("Chennai");
        }};
        System.out.println("Content of Array list cities:"+cities);
    }
}
```

Output:

```
Content of Array list cities:[Delhi, Agra, Chennai]
```

Method3: Normal way of ArrayList initialization

Syntax:

```
ArrayList<T> obj = new ArrayList<T>();
obj.add("Object o1");
```

```
    obj.add("Object o2");
    obj.add("Object o3");
    ...
    ...
```

Example:

```
import java.util.*;

public class Details {
    public static void main(String args[]) {
        ArrayList<String> books = new ArrayList<String>();
        books.add("Java Book1");
        books.add("Java Book2");
        books.add("Java Book3");
        System.out.println("Books stored in array list are: "+books);
    }
}
```

Output:

```
Books stored in array list are: [Java Book1, Java Book2, Java Book3]
```

Method 4: Use Collections.nCopies

Collections.nCopies method can be used when we need to initialize the ArrayList with the same value for all of its elements. **Syntax:** **count** is number of elements and **element** is the item value

```
ArrayList<T> obj = new ArrayList<T>(Collections.nCopies(count, element));
```

Example:

```
import java.util.*;

public class Details {
    public static void main(String args[]) {
        ArrayList<Integer> intlist = new
ArrayList<Integer>(Collections.nCopies(10, 5));
        System.out.println("ArrayList items: "+intlist);
    }
}
```

Output:

```
ArrayList items: [5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
```

Java ArrayList set() Method example

BY CHAITANYA SINGH | FILED UNDER: [JAVA COLLECTIONS](#)

If there is a need to update the list element based on the index then set method of ArrayList class can be used. The method set(int index, Element E) updates the element of specified index with the given element E.

```
public E set(int index, Element E)
```

Example:

In this example I have an ArrayList of Integer Type where I have added few elements and then I'm updating few of elements using set method of java.util.ArrayList class.

```
package beginnersbook.com;
import java.util.ArrayList;
public class SetExample {
    public static void main(String args[]) {
        ArrayList<Integer> arraylist = new ArrayList<Integer>();
        arraylist.add(1);
        arraylist.add(2);
        arraylist.add(3);
        arraylist.add(4);
        arraylist.add(5);
        arraylist.add(6);
        arraylist.add(7);
        System.out.println("ArrayList before update: "+arraylist);
        //Updating 1st element
        arraylist.set(0, 11);
        //Updating 2nd element
        arraylist.set(1, 22);
        //Updating 3rd element
        arraylist.set(2, 33);
        //Updating 4th element
        arraylist.set(3, 44);
        //Updating 5th element
        arraylist.set(4, 55);
        System.out.println("ArrayList after Update: "+arraylist);
    }
}
```

Output:

```
ArrayList before update: [1, 2, 3, 4, 5, 6, 7]
ArrayList after Update: [11, 22, 33, 44, 55, 6, 7]
```

How to compare two ArrayList in Java

BY CHAITANYA SINGH | FILED UNDER: [JAVA COLLECTIONS](#)

In this tutorial we will learn how to compare two ArrayList. We would be using [contains\(\)](#) method for comparing two elements of different ArrayList.

```
public boolean contains(Object o)
```

It returns true if the list contains the Object o else it returns false.

Example:

In this example we have two ArrayList `al1` and `al2` of String type. We have compared these ArrayLists using `contains()` method and stored the comparison result in third ArrayList (`al3` and `al4`).

```
package beginnersbook.com;
import java.util.ArrayList;
public class Details
{
    public static void main(String [] args)
    {
        ArrayList<String> al1= new ArrayList<String>();
        al1.add("hi");
        al1.add("How are you");
        al1.add("Good Morning");
        al1.add("bye");
        al1.add("Good night");

        ArrayList<String> al2= new ArrayList<String>();
        al2.add("Howdy");
        al2.add("Good Evening");
        al2.add("bye");
        al2.add("Good night");

        //Storing the comparison output in ArrayList<String>
        ArrayList<String> al3= new ArrayList<String>();
        for (String temp : al1)
            al3.add(al2.contains(temp) ? "Yes" : "No");
        System.out.println(al3);

        //Storing the comparison output in ArrayList<Integer>
        ArrayList<Integer> al4= new ArrayList<Integer>();
        for (String temp2 : al1)
            al4.add(al2.contains(temp2) ? 1 : 0);
        System.out.println(al4);
    }
}
```

Output:

```
[No, No, No, Yes, Yes]
[0, 0, 0, 1, 1]
```

What is the logic in above code?

If the first element of ArrayList al1 is present in al2 then ArrayList al3 would be having “Yes” and al4 would be having “1” However if the element is not present “No” would be stored in al3 and 0 would be in al4.

How to swap two elements in an ArrayList

BY CHAITANYA SINGH | FILED UNDER: [JAVA COLLECTIONS](#)

This tutorial will help you understand how to swap two elements in an [ArrayList](#). We are using [Collections.swap\(\)](#) method for swapping.

```
public static void swap(List list, int i1, int i2)
```

This method swaps the element of index i1 with the element of index i2. It throws IndexOutOfBoundsException – if either i1 or i2 is less than zero or greater than the size of the list (i1 < 0 || i1 >= list.size() || i2 < 0 || i2 >= list.size()).

Example of swapping two elements in ArrayList

In this example we have a ArrayList<String> and we are swapping 2nd (index =1) and 5th (index=4) element of ArrayList using Collections.swap() method.

```
package beginnersbook.com;
import java.util.ArrayList;
import java.util.Collections;

public class SwappingExample {

    public static void main(String a[]){
        ArrayList<String> al = new ArrayList<String>();
        al.add("Sachin");
        al.add("Rahul");
        al.add("Saurav");
        al.add("Sunil");
        al.add("Kapil");
        al.add("Vinod");

        System.out.println("ArrayList before Swap:");
        for(String temp: al){
            System.out.println(temp);
        }

        //Swapping 2nd(index 1) element with the 5th(index 4) element
        Collections.swap(al, 1, 4);

        System.out.println("ArrayList after swap:");
        for(String temp: al){
            System.out.println(temp);
        }
    }
}
```

```
    }
}
```

Output:

```
ArrayList before Swap:  
Sachin  
Rahul  
Saurav  
Sunil  
Kapil  
Vinod  
ArrayList after swap:  
Sachin  
Kapil  
Saurav  
Sunil  
Rahul  
Vinod
```

How to clone an ArrayList to another ArrayList

BY CHAITANYA SINGH | FILED UNDER: [JAVA COLLECTIONS](#)

In this tutorial we will learn how to clone an `ArrayList` to another one. We would be using [clone\(\) method](#) of `ArrayList` class to serve our purpose.

`Object clone()`

This method returns a shallow copy of the `ArrayList` instance.

Complete example of ArrayList Cloning

In this example we have an `ArrayList` of `String` type and we are cloning it to another `ArrayList` using `clone()` method. The interesting point to see here is when we added and removed few elements from original `ArrayList` after the `clone()` method, the cloned `ArrayList` didn't get affected. It shows that `clone()` method just returns a shallow copy of `ArrayList`.

```
package beginnersbook.com;  
import java.util.ArrayList;  
public class Details {  
  
    public static void main(String a[]){  
        ArrayList<String> al = new ArrayList<String>();
```

```

//Adding elements to the ArrayList
al.add("Apple");
al.add("Orange");
al.add("Mango");
al.add("Grapes");
System.out.println("ArrayList: "+al);

ArrayList<String> al2 = (ArrayList<String>)al.clone();
System.out.println("Shallow copy of ArrayList: "+ al2);

//add and remove on original ArrayList
al.add("Fig");
al.remove("Orange");

//Display of both ArrayLists after add & remove
System.out.println("Original ArrayList:"+al);
System.out.println("Cloned ArrayList:"+al2);
}
}

```

Output:

```

ArrayList: [Apple, Orange, Mango, Grapes]
Shallow copy of ArrayList: [Apple, Orange, Mango, Grapes]
Original ArrayList:[Apple, Mango, Grapes, Fig]
Cloned ArrayList:[Apple, Orange, Mango, Grapes]

```

Java ArrayList isEmpty() Method example

BY CHAITANYA SINGH | FILED UNDER: [JAVA COLLECTIONS](#)

`isEmpty()` method of `java.util.ArrayList` class is used for checking whether the list is empty or not. This method returns a boolean value.

```
public boolean isEmpty()
```

It returns true if the list is empty otherwise it gives false.

Example

```

package beginnersbook.com;
import java.util.ArrayList;
public class IsEmptyExample {
    public static void main(String args[]) {
        //ArrayList of Integer Type
        ArrayList<Integer> al = new ArrayList<Integer>();
        //Checking whether the list is empty
        System.out.println("Is ArrayList Empty: "+al.isEmpty());

        //Adding Integer elements
        al.add(1);
    }
}

```

```

    al.add(88);
    al.add(9);
    al.add(17);

    //Again checking for isEmpty
    System.out.println("Is ArrayList Empty: "+al.isEmpty());

    //Displaying elements of the list
    for (Integer num: al) {
        System.out.println(num);
    }
}
}

```

Output:

```

Is ArrayList Empty: true
Is ArrayList Empty: false
1
88
9
17

```

Java ArrayList trimToSize() Method example

BY CHAITANYA SINGH | FILED UNDER: [JAVA COLLECTIONS](#)

`trimToSize()` method is used for memory optimization. It trims the capacity of `ArrayList` to the current list size. For e.g. An `ArrayList` is having capacity of 15 but there are only 5 elements in it, calling `trimToSize()` method on this `ArrayList` would change the capacity from 15 to 5.

```
public void trimToSize()
```

Example

Here I have defined the `ArrayList` of capacity 50. After adding 10 elements I called `trimToSize()` method which would have reduced the capacity from 50 to 10 (current size of arraylist).

```

package beginnersbook.com;
import java.util.ArrayList;
public class TrimExample {
    public static void main(String args[]) {
        ArrayList<Integer> arraylist = new ArrayList<Integer>(50);
        arraylist.add(1);
        arraylist.add(2);
        arraylist.add(3);
        arraylist.add(4);
        arraylist.add(5);
    }
}

```

```
        arraylist.add(6);
        arraylist.add(7);
        arraylist.add(1);
        arraylist.add(1);
        arraylist.add(1);
        arraylist.trimToSize();
        System.out.println(arraylist);
    }
}
```

Output:

```
[1, 2, 3, 4, 5, 6, 7, 1, 1, 1]
```