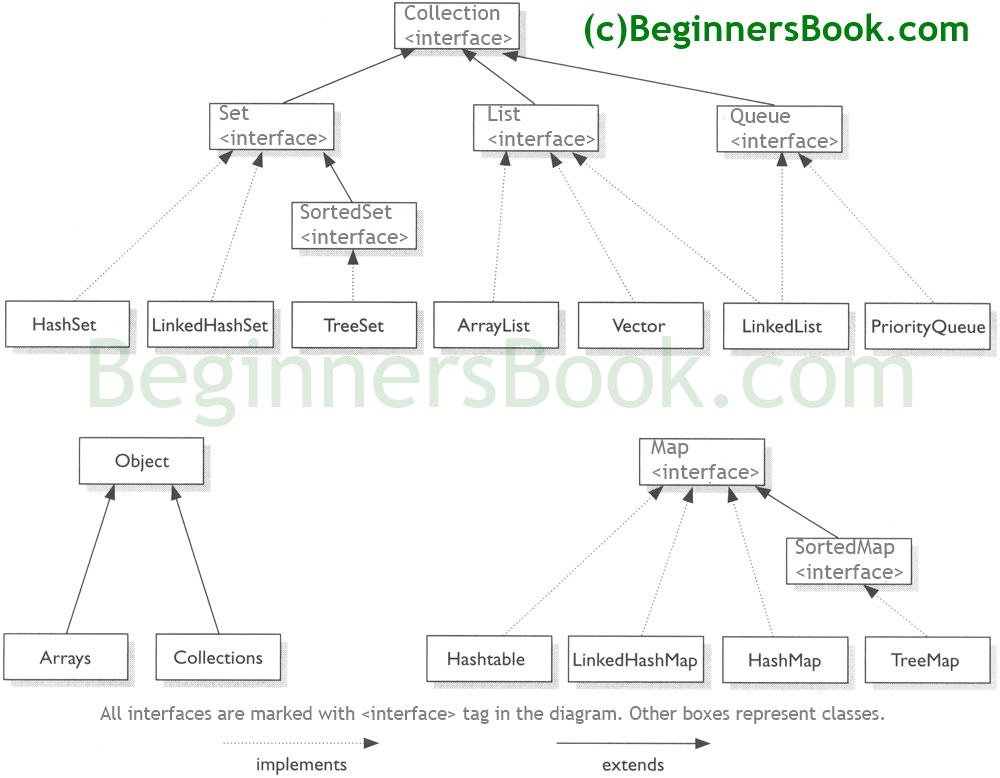
The Java Collections Framework is a collection of interfaces and classes which helps in storing and processing the data efficiently. This framework has several useful classes which have tons of useful functions which makes a programmer task super easy. I have written several tutorials on Collections in Java. All the tutorials are shared with examples and source codes to help you understand better.

**Java Collections – Table of Contents**

1. [ArrayList](https://beginnersbook.com/java-collections-tutorials/#1)  
2. [LinkedList](https://beginnersbook.com/java-collections-tutorials/#2)  
7. [HashMap](https://beginnersbook.com/java-collections-tutorials/#7)  
9. [LinkedHashMap](https://beginnersbook.com/java-collections-tutorials/#9)  
11. [Iterator and ListIterator](https://beginnersbook.com/java-collections-tutorials/#11)  
13. [Java Collections Interview Questions](https://beginnersbook.com/java-collections-tutorials/#13)

**Collections Framework hierarchy**



## Java Collections – List

A List is an ordered Collection (sometimes called a sequence). Lists may contain duplicate elements. Elements can be inserted or accessed by their position in the list, using a zero-based index.

### ArrayList

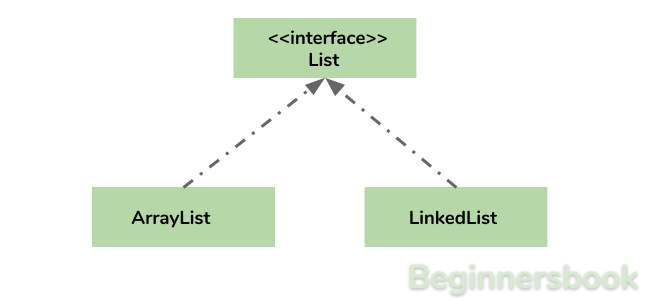
Here is the list of all the tutorials published on the ArrayList.

**ArrayList Basics**

# ArrayList in java with example programs – Collections Framework

BY CHAITANYA SINGH | FILED UNDER: [JAVA COLLECTIONS](https://beginnersbook.com/category/java-collections/)

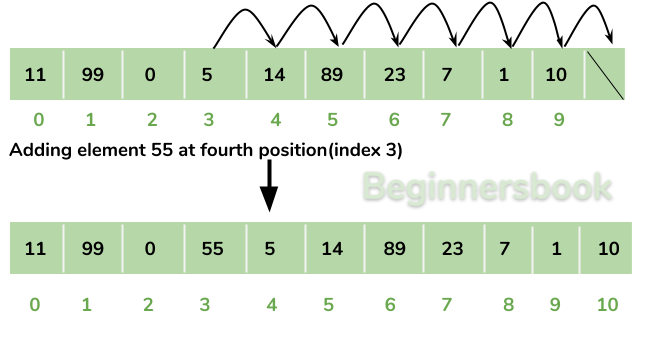
**Arraylist** class implements List interface and it is based on an Array data structure. It is widely used because of the functionality and flexibility it offers. Most of the developers **choose Arraylist over Array** as it’s a very good alternative of traditional java arrays. ArrayList is a resizable-array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



## Why ArrayList is better than Array?

The limitation with array is that it has a 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 (See the images below). Apart from these benefits ArrayList class enables us to use predefined methods of it which makes our task easy. Let’s see the diagrams to understand the addition and removal of elements from ArrayList and then we will see the programs.

**Adding Element in ArrayList at specified position:**  


**Removing Element from ArrayList:**  


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

## How to create an ArrayList?

We can create an ArrayList by writing a simple statement like this:

This statement creates an ArrayList with the name alist with type “String”. The type determines which type of elements the list will have. Since this list is of “String” type, the elements that are going to be added to this list will be of type “String”.

ArrayList<String> alist=new ArrayList<String>();

Similarly we can create ArrayList that accepts int elements.

ArrayList<Integer> list=new ArrayList<Integer>();

## How to add elements to an ArrayList?

We add elements to an ArrayList by using add() method, this method has couple of variations, which we can use based on the requirement. For example: If we want to add the element at the end of the List then simply do it like this:

alist.add("Steve"); //This will add "Steve" at the end of List

To add the element at the specified location in ArrayList, we can specify the index in the add method like this:

alist.add(3, "Steve"); //This will add "Steve" at the fourth position

Lets write the complete code:

import java.util.\*;

class JavaExample{

public static void main(String args[]){

ArrayList<String> alist=new ArrayList<String>();

alist.add("Steve");

alist.add("Tim");

alist.add("Lucy");

alist.add("Pat");

alist.add("Angela");

alist.add("Tom");

//displaying elements

System.out.println(alist);

//Adding "Steve" at the fourth position

alist.add(3, "Steve");

//displaying elements

System.out.println(alist);

}

}

**Output:**

[Steve, Tim, Lucy, Pat, Angela, Tom]

[Steve, Tim, Lucy, Steve, Pat, Angela, Tom]

**Note:** Since the index starts with 0, index 3 would represent fourth position not 3.

## How to remove elements from ArrayList?

We use remove() method to remove elements from an ArrayList, Same as add() method, this method also has few variations.

For example:

import java.util.\*;

class JavaExample{

public static void main(String args[]){

ArrayList<String> alist=new ArrayList<String>();

alist.add("Steve");

alist.add("Tim");

alist.add("Lucy");

alist.add("Pat");

alist.add("Angela");

alist.add("Tom");

//displaying elements

System.out.println(alist);

//Removing "Steve" and "Angela"

alist.remove("Steve");

alist.remove("Angela");

//displaying elements

System.out.println(alist);

//Removing 3rd element

alist.remove(2);

//displaying elements

System.out.println(alist);

}

}

**Output:**

[Steve, Tim, Lucy, Pat, Angela, Tom]

[Tim, Lucy, Pat, Tom]

[Tim, Lucy, Tom]

## Iterating ArrayList

In the above examples, we have displayed the ArrayList elements just by referring the ArrayList instance, which is definitely not the right way to displays the elements. The correct way of displaying the elements is by using an advanced for loop like this.

import java.util.\*;

class JavaExample{

public static void main(String args[]){

ArrayList<String> alist=new ArrayList<String>();

alist.add("Gregor Clegane");

alist.add("Khal Drogo");

alist.add("Cersei Lannister");

alist.add("Sandor Clegane");

alist.add("Tyrion Lannister");

//iterating ArrayList

for(String str:alist)

System.out.println(str);

}

}

**Output:**

Gregor Clegane

Khal Drogo

Cersei Lannister

Sandor Clegane

Tyrion Lannister

## ArrayList Example in Java

This example demonstrates how to create, initialize, add and remove elements from ArrayList. In this example we have an ArrayList of type “String”. We have added 5 String element in the ArrayList using the method add(String E), this method adds the element at the end of the ArrayList.

We are then adding two more elements in the ArrayList using method add(int index, String E), this method adds the specified element at the specified index, index 0 indicates first position and 1 indicates second position.

We are then removing the elements “Chaitanya” and “Harry” from the ArrayList and then we are removing the second element of the ArrayList using method remove(int index). Since we have specified the index as 1 (remove(1)), it would remove the second element.

import java.util.\*;

public class JavaExample {

public static void main(String args[]) {

/\* Creating ArrayList of type "String" which means

\* we can only add "String" elements

\*/

ArrayList<String> obj = new ArrayList<String>();

/\*This is how we add elements to an ArrayList\*/

obj.add("Ajeet");

obj.add("Harry");

obj.add("Chaitanya");

obj.add("Steve");

obj.add("Anuj");

// Displaying elements

System.out.println("Original ArrayList:");

for(String str:obj)

System.out.println(str);

/\* Add element at the given index

\* obj.add(0, "Rahul") - Adding element "Rahul" at first position

\* obj.add(1, "Justin") - Adding element "Justin" at second position

\*/

obj.add(0, "Rahul");

obj.add(1, "Justin");

// Displaying elements

System.out.println("ArrayList after add operation:");

for(String str:obj)

System.out.println(str);

//Remove elements from ArrayList like this

obj.remove("Chaitanya"); //Removes "Chaitanya" from ArrayList

obj.remove("Harry"); //Removes "Harry" from ArrayList

// Displaying elements

System.out.println("ArrayList after remove operation:");

for(String str:obj)

System.out.println(str);

//Remove element from the specified index

obj.remove(1); //Removes Second element from the List

// Displaying elements

System.out.println("Final ArrayList:");

for(String str:obj)

System.out.println(str);

}

}

**Output:**

Original ArrayList:

Ajeet

Harry

Chaitanya

Steve

Anuj

ArrayList after add operation:

Rahul

Justin

Ajeet

Harry

Chaitanya

Steve

Anuj

ArrayList after remove operation:

Rahul

Justin

Ajeet

Steve

Anuj

Final ArrayList:

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 important methods of ArrayList class.

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();