ArrayList in Java

**//1) Creating an ArrayList**

**ArrayList**<**String**> myStringList = **new** **ArrayList**<**String**>();

*//Generic ArrayList to Store only String objects*

*//2) Putting an Item into ArrayList*

myStringList.add("Item"); *//no error because we are storing String*

**//3) Checking size of ArrayList**

**int** size = myStringList.size();

**//4) Checking Index of an Item in Java ArrayList**

//You can use indexOf() method of ArrayList in Java to find out index of a particular object.

**int** index = myStringList.indexOf("Item"); *//location of Item object in List*

//**5) Retrieving Item from ArrayList in a loop**

/\*Many a times we need to **traverse on Java ArrayList** and perform some operations on each retrieved item. Here are two ways of doing it without using Iterator. We will see use of Iterator in next section.\*/

for (**int** i = 0; i < myStringList.size(); i++)  
   **String** item = myStringList.get(i);  
   **System**.out.println("Item " + i + " : " + item);  
}  
  
//From Java 5 onwards you can use foreach loop as well  
  
**for**(**String** item: myStringList){  
[**System**](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/System.html).out.println("retrieved element: " + item);  
}

//**6) Checking ArrayList for an Item**

**/\***

Sometimes we need to *check whether an element exists in ArrayList in Java* or not for this purpose we can use contains() method of Java. contains() method takes type of object defined in ArrayList creation and returns true if this list contains the specified element. Alternatively you can also use Collections.binarySearch() method to see if an object is present inside List or not. ArrayList, Vector, CopyOnWriteArrayList and Stack implements RandomAccess interface, they can be used for performing binary search. To see which approach is better, see [this](http://javarevisited.blogspot.sg/2014/03/binary-search-vs-contains-performance.html) article.

\*/

**//7) Checking if ArrayList is Empty**

/\*We can use **isEmpty() method of Java ArrayList to check whether ArrayList is empty**. isEmpty() method returns true if this ArrayList contains no elements. You can also use size() method of List to check if List is empty

\*/

**boolean** result = myStringList.isEmpty(); *//isEmpty() will return true if List is empty*  
  
**if**(myStringList.size() == 0){  
   **System**.out.println("ArrayList is empty");  
}

**//8) Removing an Item from ArrayList**

/\*

There are two ways to **remove any elements from ArrayList in Java**. You can either remove an element based on its index or by providing object itself. Remove remove (int index) and remove (Object o) method is used to remove any element from ArrayList in Java. Since ArrayList allows duplicate its worth noting that remove (Object o) removes the first occurrence of the specified element from this list, if it is present. In below code first call will remove first element from ArrayList while second call will remove first occurrence of item from ArrayList in Java.

\*/

myStringList.remove(0);

myStringList.remove(item);

**//9) Copying data from one ArrayList to another ArrayList in Java**

**/\***

Many a times you need to **create a copy of ArrayList** for this purpose you can use addAll(Collection c) method of ArrayList in Java to copy all elements from on ArrayList to another ArrayList in Java. Below code will add all elements of myStringList to newly created copyOfMyStringList.

\*/

**ArrayList**<**String**> copyOfMyStringList = **new** **ArrayList**<**String**>();  
copyOfMyStringList.addAll(myStringList);

**//10) Replacing an element at a particular index**

/\*

You can use set (int index, E element) method of Java ArrayList to replace any element from a particular index. Below code will replace first element of myStringList from "Item" to "Item2".

\*/

myStringList.set(0,"Item2");

**//11) Clearing all data from ArrayList**

/\*

ArrayList in Java provides clear() method which removes all of the elements from this list. Below code will remote all elements from our myStringList and make the list empty. **You can reuse Java ArrayList** after clearing it.

\*/

myStringList.clear();

**//12) Converting from ArrayList to Array in Java**

**/\***

**Java ArrayList** provides you facility to *get the array back from your ArrayList*. You can use toArray(T[] a) method returns an array containing all of the elements in this list in proper sequence (from first to last element). "a" is the array into which the elements of the list are to be stored, if it is big enough; otherwise, a new array of the same runtime type is allocated for this purpose.

\*/

**String**[] itemArray = **new** **String**[mySringList.size()];  
**String**[] returnedArray = myStringList.toArray(itemArray);

/\*

If you want to convert ArrayList back to Array than see 3 ways to convert array into arraylist in Java

\*/

**//13) Creating Synchronized ArrayList**

Sometimes you need to synchronize your ArrayList in java to make it shareable between multiple threads you can use Collections utility class for this purpose as shown below.

**List** list = **Collections**.synchronizedList(**new** **ArrayList**(...));

**//14) Creating ArrayList from Array in Java**

ArrayList in Java is amazing you can create even an ArrayList full of your element from an already existing array. You need to use Arrays.asList(T... a)  method for this purpose which returns a fixed-size list backed by the specified array.

**ArrayList** myStringList = **Arrays**.asList(**new** **String**[]{"One", "Two", "Three"); *//this is not read only List you can still update value of existing elements*

**//15) Traversing in ArrayList in Java**

You can use either [Iterator](http://javarevisited.blogspot.com/2010/10/what-is-difference-between-enumeration.html) or ListIterator for **traversing on Java ArrayList**. ListIterator will allow you to traverse in both directions while both Iterator and ListIterator will allow you to r*emove elements from ArrayList in Java while traversing*.

**Iterator** itr = myStringList.iterator();  
**while**(itr.hasNext()){  
**System**.out.println(itr.next());  
}  
  
**ListIterator** listItr = stringList.listIterator();  
**while**(listItr.hasNext()){  
**System**.out.println(itr.next());  
}

**//16) Sorting elements of ArrayList in Java**

You can use Collections.sort(List list) method to sort a Java ArrayList in natural order defined by Comparable interface and can use Collections.sort(List list, Comparator c) method to sort your Java ArrayList based upon custom Comparator.

**//17) ArrayList to HashSet conversion**  
Most of Collection class provides a constructor which accepts a Collection object as argument. Which can be used to copy all elements of one Collection into another. HashSet also provide such constructors which can be used to copy all object from ArrayList to HashSet. But be careful since HashSet doesn't allow duplicates some of the objects will not be included which result in less number of objects.

<http://java-tips.org/java-se-tips/java.lang/use-of-arraylist-class.html>