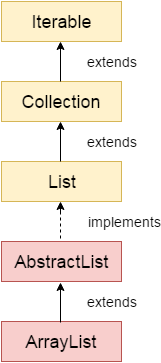
**ARRAYLIST:**

Java ArrayList class uses a dynamic array for storing the elements. It inherits AbstractList class and implements List interface.

The important points about Java ArrayList class are:

* Java ArrayList class can contain duplicate elements.
* Java ArrayList class maintains insertion order.
* Java ArrayList class is non synchronized.
* Java ArrayList allows random access because array works at the index basis.
* In Java ArrayList class, manipulation is slow because a lot of shifting needs to occur if any element is removed from the array list.



### Java ArrayList Example

### **1) Explain the different ways of constructing an ArrayList?**

ArrayList can be created in 3 ways.

1. **ArrayList()** —> It creates an empty ArrayList with initial capacity of 10.

ArrayList<Integer> list1 = new ArrayList<Integer>();

1. **ArrayList(int initialCapacity)** —> It creates an empty ArrayList with supplied initial capacity.

ArrayList<String> list2 = new ArrayList<String>(20);

1. **ArrayList(Collection c)** —> It creates an ArrayList containing the elements of the supplied collection.

ArrayList<Integer> list3 = new ArrayList<Integer>(list1);

### **2)**Creating an array list and adding elements?

**import** java.util.\*;

**class** ArrayList1{

**public** **static** **void** main(String args[]){

ArrayList<String> list=**new** ArrayList<String>();//Creating arraylist

      list.add("Ravi");//Adding object in arraylist

      list.add("Vijay");

      list.add("Ravi");

      list.add("Ajay");

      //Invoking arraylist object

      System.out.println(list);

  }

 }

}

### ****3) Iterating Collection through Iterator interface****

Let's see an example to traverse ArrayList elements using the Iterator interface.

1. **import** java.util.\*;
2. **class** ArrayList2{
3. **public** **static** **void** main(String args[]){
4. ArrayList<String> list=**new** ArrayList<String>();//Creating arraylist
5. list.add("Ravi");//Adding object in arraylist
6. list.add("Vijay");
7. list.add("Ravi");
8. list.add("Ajay");
9. //Traversing list through Iterator
10. Iterator itr=list.iterator();
11. **while**(itr.hasNext()){
12. System.out.println(itr.next());
13. }
14. }
15. }

**4) How do you find the number of elements present in an ArrayList?**

Using **size()** method. size() method returns number of elements present in an ArrayList.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | public class MainClass  {      public static void main(String[] args)      {          ArrayList<Double> list = new ArrayList<Double>();            list.add(1.1);            list.add(2.2);            list.add(3.3);            list.add(4.4);            list.add(5.5);            System.out.println(list);     //Output : [1.1, 2.2, 3.3, 4.4, 5.5]            System.out.println("Size Of ArrayList = "+list.size());   //Output : Size Of ArrayList = 5      }  } |

**5) How do you find out whether the given ArrayList is empty or not?**

**isEmpty()** method of ArrayList is used to check whether the given ArrayList is empty or not. This method returns true if an ArrayList contains no elements otherwise returns false.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public class MainClass  {      public static void main(String[] args)      {          ArrayList<Double> list = new ArrayList<Double>();            System.out.println(list.isEmpty());    //Output : true      }  } |

**6) How do you check whether the given element is present in an ArrayList or not?**

Using **contains()** method of ArrayList, we can examine whether the ArrayList contains the given element or not. This method returns true if ArrayList has that element otherwise returns false.

public class MainClass

{

    public static void main(String[] args)

    {

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

        list.add(1.1);

        list.add(11.11);

        list.add(111.111);

        list.add(1111.1111);

        //Checking whether list conatins '111.1111'

        System.out.println(list.contains(111.1111));    //Output : false

    }

}

**7) How do you get the position of a particular element in an ArrayList?**

We can use **indexOf()** and **lastIndexOf()** methods to find out the position of a given element in an ArrayList.**indexOf()** method returns index of first occurrence of a specified element where as **lastIndexOf()** method returns index of last occurrence of a specified element in an ArrayList. If element is not found, they will return -1.

public class MainClass

{

    public static void main(String[] args)

    {

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

        list.add("JAVA");

        list.add("J2EE");

        list.add("JSP");

        list.add("JAVA");

        list.add("SERVLETS");

        list.add("JAVA");

        list.add("STRUTS");

        System.out.println(list);     //Output : [JAVA, J2EE, JSP, JAVA, SERVLETS, JAVA, STRUTS]

        //Getting the index of first occurrence of "JAVA"

        System.out.println(list.indexOf("JAVA"));     //Output : 0

        //Getting the index of last occurrence of "JAVA"

        System.out.println(list.lastIndexOf("JAVA"));    //Output : 5

    }

}

**8) How do you convert an ArrayList to Array?**

Using **toArray()** method of ArrayList class. toArray() method returns an array containing all elements of the ArrayList. This method acts as a bridge between normal arrays and collection framework in java.

public class MainClass

{

    public static void main(String[] args)

    {

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

        list.add("JAVA");

        list.add("J2EE");

        list.add("JSP");

        list.add("SERVLETS");

        list.add("STRUTS");

        System.out.println(list);      //Output : [JAVA, J2EE, JSP, SERVLETS, STRUTS]

        //getting an array containing all elements of the list.

        Object[] array = list.toArray();

        //Printing the elements of the returned array.

        for (Object object : array)

        {

            System.out.println(object);

        }

//      Output :

//      JAVA

//      J2EE

//      JSP

//      SERVLETS

//      STRUTS

    }

}

**9) How do you retrieve an element from a particular position of an ArrayList?**

**get()** method returns an element from a specified position of an ArrayList. This method takes index of the element as an argument.

public class MainClass

{

    public static void main(String[] args)

    {

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

        list.add(111);

        list.add(222);

        list.add(333);

        list.add(444);

        System.out.println(list);     //Output : [111, 222, 333, 444]

        //Getting element at index 3

        System.out.println(list.get(3));    //Output : 444

        //Getting element at index 1

        System.out.println(list.get(1));    //Output : 222

    }

}

### **10) How do you replace a particular element in an ArrayList with the given element?**

**set()** method replaces a particular element in an Arraylist with the given element. This method takes two arguments. One is the index of the element to be replaced and another one is the element to be placed at that position.

public class MainClass

{

    public static void main(String[] args)

    {

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

        list.add(111);

        list.add(222);

        list.add(333);

        list.add(444);

        System.out.println(list);     //Output : [111, 222, 333, 444]

        //Replacing the element at index 1 with '000'

        list.set(1, 000);

        //Replacing the element at index 3 with '000'

        list.set(3, 000);

        System.out.println(list);   //Output : [111, 0, 333, 0]

    }

}

**11) How do you append an element at the end of an ArrayList?**

**add()** method appends an element at the end of an ArrayList.

public class MainClass

{

    public static void main(String[] args)

    {

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

        list.add("ONE");

        list.add("TWO");

        list.add("THREE");

        list.add("FOUR");

        System.out.println(list);     //Output : [ONE, TWO, THREE, FOUR]

    }

}

**12) How do you insert an element at a particular position of an ArrayList?**

**add()** method which takes index and an element as arguments can be used to insert an element at a particular position of an ArrayList. The elements at the right side of that position are shifted one position right i.e indices of right side elements of that position are increased by 1.

public class MainClass

{

    public static void main(String[] args)

    {

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

        list.add("ONE");

        list.add("TWO");

        list.add("THREE");

        list.add("FOUR");

        System.out.println(list);     //Output : [ONE, TWO, THREE, FOUR]

        //Inserting "AAA" at index 1

        list.add(1, "AAA");

        //Inserting "BBB" at index 3

        list.add(3, "BBB");

        System.out.println(list);    //Output : [ONE, AAA, TWO, BBB, THREE, FOUR]

    }

}

**13) How do you remove an element from a particular position of an ArrayList?**

remove() method which takes int type as an argument is used to remove an element from a particular position of an ArrayList.

public class MainClass

{

    public static void main(String[] args)

    {

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

        list.add("AAA");

        list.add("BBB");

        list.add("ccc");

        list.add("DDD");

        list.add("e");

        System.out.println(list);     //Output : [AAA, BBB, ccc, DDD, e]

        //Removing an element from position 2

        list.remove(2);

        System.out.println(list);    //Output : [AAA, BBB, DDD, e]

        //Removing an element from position 3

        list.remove(3);

        System.out.println(list);   //Output : [AAA, BBB, DDD]

    }

}

### **14) How do you remove the given element from an ArrayList?**

**remove(Object obj)** method removes the first occurrence of the specified element ‘**obj**‘. If that element doesn’t exist, ArrayList will be unchanged.

public class MainClass

{

    public static void main(String[] args)

    {

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

        list.add("AAA");

        list.add("BBB");

        list.add("AAA");

        list.add("CCC");

        list.add("BBB");

        System.out.println(list);     //Output : [AAA, BBB, AAA, CCC, BBB]

        //Removing first occurrence of "AAA"

        list.remove("AAA");

        System.out.println(list);    //Output : [BBB, AAA, CCC, BBB]

        //Removing first occurrence of "BBB"

        list.remove("BBB");

        System.out.println(list);   //Output : [AAA, CCC, BBB]

    }

}

### **15) How do you remove all elements of an ArrayList at a time?**

**clear()** method removes all elements of an ArrayList. ArrayList will be empty after this method is executed.

public class MainClass

{

    public static void main(String[] args)

    {

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

        list.add("AAA");

        list.add("BBB");

        list.add("AAA");

        list.add("CCC");

        list.add("BBB");

        System.out.println(list);     //Output : [AAA, BBB, AAA, CCC, BBB]

        //Removing all elements of the list

        list.clear();

        System.out.println(list);    //Output : []

    }

}

**16) How do you retrieve a portion of an ArrayList?**

Using **subList()** method of ArrayList, we can retrieve a portion of an ArrayList. subList() method returns a view of a portion of an ArrayList in the given range. The returned subList is backed by original ArrayList. That means any changes made to subList will be reflected in original ArrayList or Vice-Versa.

public class MainClass

{

    public static void main(String[] args)

    {

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

        list.add(111);

        list.add(222);

        list.add(333);

        list.add(444);

        list.add(555);

        list.add(666);

        System.out.println(list);     //Output : [111, 222, 333, 444, 555, 666]

        //Retrieving a SubList

        List<Integer> subList = list.subList(1, 4);

        System.out.println(subList);    //Output : [222, 333, 444]

        //Modifying the list

        list.set(2, 000);

        //Changes will be reflected in subList

        System.out.println(subList);    //Output : [222, 0, 444]

        //Modifying the subList

        subList.set(2, 000);

        //Changes will be reflected in list

        System.out.println(list);    //Output : [111, 222, 0, 0, 555, 666]

    }

}

### **17) How do you join two ArrayLists?**

We can use addAll() method which takes Collection type as an argument to join two ArrayLists. This method appends all elements of the passed collection to the end of the invoking collection.

public class MainClass

{

    public static void main(String[] args)

    {

        ArrayList<Integer> list1 = new ArrayList<Integer>();

        list1.add(111);

        list1.add(222);

        list1.add(333);

        list1.add(444);

        System.out.println(list1);     //Output : [111, 222, 333, 444]

        ArrayList<Integer> list2 = new ArrayList<Integer>();

        list2.add(555);

        list2.add(666);

        list2.add(777);

        list2.add(888);

        System.out.println(list2);    //Output : [555, 666, 777, 888]

        //Joining list1 and list2

        list1.addAll(list2);

        System.out.println(list1);    //Output : [111, 222, 333, 444, 555, 666, 777, 888]

    }

}

**18) How do you insert more than one element at a particular position of an ArrayList?**

Another version of addAll() method which takes two arguments, one is index and another one is Collection type, can be used for this requirement. This method inserts all of the elements of passed collection at a specified position of an ArrayList.

public class MainClass

{

    public static void main(String[] args)

    {

        ArrayList<Integer> list1 = new ArrayList<Integer>();

        list1.add(111);

        list1.add(222);

        list1.add(333);

        list1.add(444);

        System.out.println(list1);     //Output : [111, 222, 333, 444]

        ArrayList<Integer> list2 = new ArrayList<Integer>();

        list2.add(555);

        list2.add(666);

        list2.add(777);

        list2.add(888);

        System.out.println(list2);    //Output : [555, 666, 777, 888]

        //Inserting all elements of list2 at index 2 of list1

        list1.addAll(2, list2);

        System.out.println(list1);    //Output : [111, 222, 555, 666, 777, 888, 333, 444]

    }

}

### Java ArrayList example to add elements

Here, we see different ways to add an element.

1. **import** java.util.\*;
2. **class** ArrayList7{
3. **public** **static** **void** main(String args[]){
4. ArrayList<String> al=**new** ArrayList<String>();
5. System.out.println("Initial list of elements: "+al);
6. //Adding elements to the end of the list
7. al.add("Ravi");
8. al.add("Vijay");
9. al.add("Ajay");
10. System.out.println("After invoking add(E e) method: "+al);
11. //Adding an element at the specific position
12. al.add(1, "Gaurav");
13. System.out.println("After invoking add(int index, E element) method: "+al);
14. ArrayList<String> al2=**new** ArrayList<String>();
15. al2.add("Sonoo");
16. al2.add("Hanumat");
17. //Adding second list elements to the first list
18. al.addAll(al2);
19. System.out.println("After invoking addAll(Collection<? extends E> c) method: "+al);
20. ArrayList<String> al3=**new** ArrayList<String>();
21. al3.add("John");
22. al3.add("Rahul");
23. //Adding second list elements to the first list at specific position
24. al.addAll(1, al3);
25. System.out.println("After invoking addAll(int index, Collection<? extends E> c) method: "+al);
27. }
28. }

### Java ArrayList example to remove elements

Here, we see different ways to remove an element.

1. **import** java.util.\*;
2. **class** ArrayList8 {
4. **public** **static** **void** main(String [] args)
5. {
6. ArrayList<String> al=**new** ArrayList<String>();
7. al.add("Ravi");
8. al.add("Vijay");
9. al.add("Ajay");
10. al.add("Anuj");
11. al.add("Gaurav");
12. System.out.println("An initial list of elements: "+al);
13. //Removing specific element from arraylist
14. al.remove("Vijay");
15. System.out.println("After invoking remove(object) method: "+al);
16. //Removing element on the basis of specific position
17. al.remove(0);
18. System.out.println("After invoking remove(index) method: "+al);
20. //Creating another arraylist
21. ArrayList<String> al2=**new** ArrayList<String>();
22. al2.add("Ravi");
23. al2.add("Hanumat");
24. //Adding new elements to arraylist
25. al.addAll(al2);
26. System.out.println("Updated list : "+al);
27. //Removing all the new elements from arraylist
28. al.removeAll(al2);
29. System.out.println("After invoking removeAll() method: "+al);
30. //Removing elements on the basis of specified condition
31. al.removeIf(str -> str.contains("Ajay"));   //Here, we are using Lambda expression
32. System.out.println("After invoking removeIf() method: "+al);
33. //Removing all the elements available in the list
34. al.clear();
35. System.out.println("After invoking clear() method: "+al);
36. }
37. }

**19) How to reverse ArrayList?**

ArrayList aList = new ArrayList();

//Add elements to ArrayList object

aList.add("1");

aList.add("2");

aList.add("3");

aList.add("4");

aList.add("5");

Collections.reverse(aList);

System.out.println("After Reverse Order, ArrayList Contains : " + aList);

**Java Program To Reverse An ArrayList :**

**import java.util.ArrayList;**

**import java.util.Collections;**

**public class ReverseArrayListExample**

**{**

**public static void main(String[] args)**

**{**

**//Constructing an ArrayList**

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

**list.add("Gold");**

**list.add("Iron");**

**list.add("Copper");**

**list.add("Silver");**

**list.add("Nickel");**

**list.add("Cobalt");**

**list.add("Zinc");**

**//Printing list before reverse**

**System.out.println("ArrayList Before Reverse :");**

**System.out.println(list);**

**//Reversing the list using Collections.reverse() method**

**Collections.reverse(list);**

**//Printing list after reverse**

**System.out.println("ArrayList After Reverse :");**

**System.out.println(list);**

**}**

**}**

**Output :**

**ArrayList Before Reverse :**

**[Gold, Iron, Copper, Silver, Nickel, Cobalt, Zinc]**

**ArrayList After Reverse :**

**[Zinc, Cobalt, Nickel, Silver, Copper, Iron, Gold]**

# 20) Difference between length of Array and size of ArrayList in Java?

**Array** has length property which provides the length of the Array or Array object. It is the total space allocated in memory during the initialization of the array. Array is static so when we create an array of size n then n block are created of array type and JVM initialize every block by default value.

On the other hand, java **ArrayList** does not have length() method, java ArrayList has size() method for ArrayList which provides the total number of objects available in the collection.

|  |
| --- |
| public static void main(String[] args)      {            /\* creating an array A[] for 10 elements \*/          String A[] = new String[10];            /\* store 2 elements \*/          A[0] = "Hello";          A[1] = "Geeks!";            /\* print length of array A[] \*/          System.out.println(A.length); // 10            /\* Creating an ArrayList \*/          ArrayList<String> al = new ArrayList<String>();            /\* add 3 elements \*/          al.add("G");          al.add("F");          al.add("G");            /\* print size of ArrayList \*/          System.out.println(al.size()); // 3      }  } |

**Output:**

10

3

# 21) Convert an ArrayList of String to a String array in Java?

public class GFG {

    // Function to convert ArrayList<String> to String[]

    public static String[] GetStringArray(ArrayList<String> arr)

    {

        // declaration and initialise String Array

        String str[] = new String[arr.size()];

        // ArrayList to Array Conversion

        for (int j = 0; j < arr.size(); j++) {

            // Assign each value to String array

            str[j] = arr.get(j);

        }

        return str;

    }

    // Driver code

    public static void main(String[] args)

# {

        // declaration and initialise ArrayList

        ArrayList<String>

            a1 = new ArrayList<String>();

        a1.add("Geeks");

        a1.add("for");

        a1.add("Geeks");

        // print ArrayList

        System.out.println("ArrayList: " + a1);

        // Get String Array

        String[] str = GetStringArray(a1);

        // Print Array elements

        System.out.print("String Array[]: "

                         + Arrays.toString(str));

    }

}

# 23) How to make an ArrayList read only in Java?

An ArrayList can be made read-only easily with the help of Collections.unmodifiableList() method. This method takes the modifiable ArrayList as a parameter and returns the read-only unmodifiable view of this ArrayList.

readOnlyArrayList = Collections.unmodifiableList(ArrayList);

|  |
| --- |
| public class GFG1 {      public static void main(String[] argv)          throws Exception      {          try {                // creating object of ArrayList<Character>              List<Character> list = new ArrayList<Character>();                // populate the list              list.add('X');              list.add('Y');              list.add('Z');                // printing the list              System.out.println("Initial list: "                                 + list);                // getting readonly list              // using unmodifiableList() method              List<Character>                  immutablelist = Collections                                      .unmodifiableList(list);                // printing the list              System.out.println("ReadOnly ArrayList: "                                 + immutablelist);                // Adding element to new Collection              System.out.println("\nTrying to modify"                                 + " the ReadOnly ArrayList.");              immutablelist.add('A');          }            catch (UnsupportedOperationException e) {              System.out.println("Exception thrown : " + e);          }      }  } |

**Output:**

Initial list: [X, Y, Z]

ReadOnly ArrayList: [X, Y, Z]

Trying to modify the ReadOnly ArrayList.

Exception thrown : java.lang.UnsupportedOperationException

# 24) How to sort an ArrayList in Ascending Order in Java?

Collections.sort(ArrayList);

public static void main(String args[])

    {

        // Get the ArrayList

        ArrayList<String>

            list = new ArrayList<String>();

        // Populate the ArrayList

        list.add("Geeks");

        list.add("For");

        list.add("ForGeeks");

        list.add("GeeksForGeeks");

        list.add("A computer portal");

        // Print the unsorted ArrayList

        System.out.println("Unsorted ArrayList: "

                           + list);

        // Sorting ArrayList in ascending Order

        // using Collection.sort() method

        Collections.sort(list);

        // Print the sorted ArrayList

        System.out.println("Sorted ArrayList "

                           + "in Ascending order : "

                           + list);

    }

}

# 25) How to sort an ArrayList in Descending Order in Java?

There are two ways:

1. First sort in ascending order and then reverse.

Collections.sort(list);  
Collections.reverse(list);

1. Use Collection.reverseorder

Collections.sort(ArrayList, Collections.reverseOrder());

public class GFG {

    public static void main(String args[])

    {

        // Get the ArrayList

        ArrayList<String>

            list = new ArrayList<String>();

        // Populate the ArrayList

        list.add("Geeks");

        list.add("For");

        list.add("ForGeeks");

        list.add("GeeksForGeeks");

        list.add("A computer portal");

        // Print the unsorted ArrayList

        System.out.println("Unsorted ArrayList: "

                           + list);

        // Sorting ArrayList in descending Order

        // using Collection.sort() method

        // by passing Collections.reverseOrder() as comparator

        Collections.sort(list, Collections.reverseOrder());

        // Print the sorted ArrayList

        System.out.println("Sorted ArrayList "

                           + "in Descending order : "

                           + list);

    }

}

# 26)What is subList() method in ArrayList?

public List subList(int fromIndex, int toIndex)

**Parameters:** This method takes the following argument as a parameter.

* **fromIndex –** low endpoint (inclusive) of the subList
* **toIndex –** high endpoint (exclusive) of the subList

|  |
| --- |
| public class GFG1 {      public static void main(String[] argv)          throws Exception      {            try {                // Creating object of ArrayList<Integer>              ArrayList<String>                  arrlist = new ArrayList<String>();                // Populating arrlist1              arrlist.add("A");              arrlist.add("B");              arrlist.add("C");              arrlist.add("D");              arrlist.add("E");                // print arrlist              System.out.println("Orignal arrlist: "                                 + arrlist);                // getting the subList              // using subList() method              List<String> arrlist2 = arrlist.subList(2, 4);                // print the subList              System.out.println("Sublist of arrlist: "                                 + arrlist2);          }            catch (IndexOutOfBoundsException e) {              System.out.println("Exception thrown : " + e);          }            catch (IllegalArgumentException e) {              System.out.println("Exception thrown : " + e);          }      }  } |

* **Output:**
* Orignal arrlist: [A, B, C, D, E]
* Sublist of arrlist: [C, D]

# 28) Program to convert ArrayList to LinkedList in Java?

**Algorithm**:

1. Get the ArrayList to be converted.
2. Create an empty LinkedList
3. Iterate through the items in the ArrayList.
4. For each item, add it to the LinkedList
5. Return the formed LinkedList

**29) How to compare two ArrayList in Java**

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);

**}**

**30) swapping two elements in ArrayList**

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);

}

}

}

**31) How to serialize ArrayList in java**

[ArrayList](https://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html) is [serializable](https://docs.oracle.com/javase/6/docs/api/java/io/Serializable.html) by **default**. This means you need not to implement Serializable interface explicitly in order to serialize an ArrayList. In this tutorial we will learn how to serialize and de-serialize an ArrayList.

package beginnersbook.com;

import java.util.ArrayList;

import java.io.\*;

public class ArrayListSerialization

{

public static void main(String [] args)

{

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

al.add("Hello");

al.add("Hi");

al.add("Howdy");

try{

FileOutputStream fos= new FileOutputStream("myfile");

ObjectOutputStream oos= new ObjectOutputStream(fos);

oos.writeObject(al);

oos.close();

fos.close();

}catch(IOException ioe){

ioe.printStackTrace();

}

}

}

**De-Serialization**:

In this class we are retrieving the stream of bytes from myfile which we have stored using the above class. We are type casting the returned object to ArrayList and displaying the elements of ArrayList.

package beginnersbook.com;

import java.io.\*;

import java.util.ArrayList;

public class DeSerializationClass

{

public static void main(String [] args)

{

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

try

{

FileInputStream fis = new FileInputStream("myfile");

ObjectInputStream ois = new ObjectInputStream(fis);

arraylist = (ArrayList) ois.readObject();

ois.close();

fis.close();

}catch(IOException ioe){

ioe.printStackTrace();

return;

}catch(ClassNotFoundException c){

System.out.println("Class not found");

c.printStackTrace();

return;

}

for(String tmp: arraylist){

System.out.println(tmp);

}

}

}

**32) How to join/combine two ArrayLists in java**

import java.util.ArrayList;

public class Details

{

public static void main(String [] args)

{

//First ArrayList

ArrayList<String> arraylist1=new ArrayList<String>();

arraylist1.add("AL1: E1");

arraylist1.add("AL1: E2");

arraylist1.add("AL1: E3");

//Second ArrayList

ArrayList<String> arraylist2=new ArrayList<String>();

arraylist2.add("AL2: E1");

arraylist2.add("AL2: E2");

arraylist2.add("AL2: E3");

//New ArrayList

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

al.addAll(arraylist1);

al.addAll(arraylist2);

//Displaying elements of the joined ArrayList

for(String temp: al){

System.out.println(temp);

}

}

}

**33) How to clone an ArrayList to another ArrayList**

In this tutorial we will learn how to clone an ArrayList to another one. We would be using [clone() method](https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html#clone()) of ArrayList class to serve our purpose.

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]

34) [**Increase the capacity(size) of ArrayList**](https://beginnersbook.com/2013/12/java-arraylist-ensurecapacity-method-example/)

ArrayList internally implements growable dynamic array which means it can increase and decrease its size automatically. If we try to add an element to a already full ArrayList then it automatically re-sized internally to accommodate the new element however sometimes its not a good approach.

Consider a scenario when there is a need to add huge number of elements to an already full ArrayList, in such case ArrayList has to be resized several number of times which would result in a poor performance. For such scenarios ensureCapacity() method of Java.util.ArrayList class is very useful as it increases the size of the ArrayList by a specified capacity.

import java.util.ArrayList;

public class EnsureCapacityExample {

public static void main(String args[]) {

// ArrayList with Capacity 4

ArrayList<String> al = new ArrayList<String>(4);

//Added 4 elements

al.add("Hi");

al.add("Hello");

al.add("Bye");

al.add("GM");

//Increase capacity to 5

al.ensureCapacity(5);

al.add("GE");

// let us print all the elements available in list

for (String temp: al) {

System.out.println(temp);

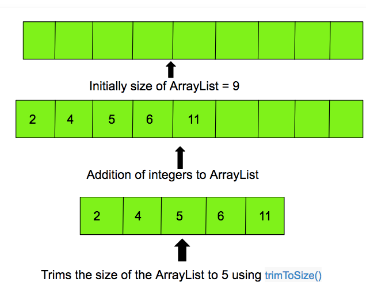
}

}

}

# 35) What is ArrayList trimToSize() in ArrayList?

The **trimToSize()** method of [**ArrayList**](https://www.geeksforgeeks.org/arraylist-in-java/) in Java trims the capacity of an ArrayList instance to be the list’s current size.



public class GFG {

    public static void main(String[] args)

    {

        // creating an Empty Integer ArrayList

        ArrayList<Integer> arr = new ArrayList<Integer>(9);

        // using add(), add 5 values

        arr.add(2);

        arr.add(4);

        arr.add(5);

        arr.add(6);

        arr.add(11);

        // trims the size to the number of elements

        arr.trimToSize();

        System.out.println("The List elements are:");

        // prints all the elements

        for (Integer number : arr) {

            System.out.println("Number = " + number);

        }

    }

}

# 36) What is removeRange() in ArrayList Java?

The **removeRange()** method of [**ArrayList**](https://www.geeksforgeeks.org/arraylist-in-java/) in Java is used to remove all elements within the specified range from an ArrayList object.

removeRange(int fromIndex, int toIndex)

* **fromIndex** − This is the index of first element to be removed.
* **toIndex** − This is the index after last element to be removed.
* public class ArrayListDemo extends ArrayList {
* public static void main(String[] args) {
* // create an empty array list
* ArrayListDemo arrlist = new ArrayListDemo();
* // use add() method to add values in the list
* arrlist.add(10);
* arrlist.add(12);
* arrlist.add(31);
* // print the list
* System.out.println("The list:" + arrlist);
* // removing range of 1st 2 elements
* arrlist.removeRange(0,2);
* System.out.println("The list after using removeRange:" + arrlist);
* }
* }
* Let us compile and run the above program, this will produce the following result −
* The list:[10, 12, 31]
* The list after using removeRange:[31]

37) Given an arraylist, convert it into a  HashSet of strings in Java?

**Method 1 (Simple)**

class Test {

    public static void main(String[] args)

    {

        // Creating a list of strings

        List<String> aList = Arrays.asList("Geeks", "for",

                     "GeeksQuiz", "GeeksforGeeks", "GFG");

        Set<String> hSet = new HashSet<String>();

        for (String x : aList)

            hSet.add(x);

        System.out.println("Created HashSet is");

        for (String x : hSet)

            System.out.println(x);

        // We can created TreeSet same way

    }

}

**Method 2 (Using addAll method)**

class Test {

    public static void main(String[] args)

    {

        // Creating a list of strings

        List<String> aList = Arrays.asList("Geeks", "for",

                    "GeeksQuiz", "GeeksforGeeks", "GFG");

        Set<String> hSet = new HashSet<String>(aList);

        hSet.addAll(aList);

        System.out.println("Created HashSet is");

        for (String x : hSet)

            System.out.println(x);

        Set<String> tSet = new TreeSet<String>(aList);

        tSet.addAll(aList);

        System.out.println("Created TreeSet is");

        for (String x : tSet)

            System.out.println(x);

    }

}

# 37) Synchronization of ArrayList in Java?

Implementation of arrayList is not synchronized is by default. It means if a thread modifies it structurally and multiple threads access it concurrently, it must be synchronized externally.

Structural modification means addition or deletion of element(s) from the list or explicitly resizes the backing array. Changing the value of existing element is not structural modification.

**Using Collections.synchronizedList() method**

1. **import** java.util.\*;
2. **public** **class** SyncronizeArrayList {
3. **public** **static** **void** main(String args[]) {
4. // Non Synchronized ArrayList
5. List<String> fruitList = **new** ArrayList<String>();
7. fruitList.add("Mango");
8. fruitList.add("Banana");
9. fruitList.add("Apple");
10. fruitList.add("Strawberry");
11. fruitList.add("Pineapple");
13. // Synchronizing ArrayList in Java
14. furitList = Collections.synchronizedList(fruitList);
16. // we must use synchronize block to avoid non-deterministic behavior
17. **synchronized** (fruitList) {
18. Iterator<String> itr = fruitList.iterator();
19. **while** (itr.hasNext()) {
20. System.out.println(itr.next());
21. }
22. }
23. }
24. }

38) What is time complexity of ArrayList for different operations?

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Array | ArrayList | Singly Linked List |
| Read (any where) | O(1) | O(1) | O(n) |
| Add/Remove at end | O(1) | O(1) | O(n) |
| Add/Remove in the interior | O(n) | O(n) | O(n) |
| Resize | O(n) | N/A | N/A |
| Find By position | O(1) | O(1) | O(n) |
| Find By target (value) | O(n) | O(n) | O(n) |

38) How to find largest and smallest number in unsorted array?  
  
This is a rather simple array interview question. You have given an unsorted integer array and you need to find the largest and smallest element in the array. Of course, you can sort the array and then pick the top and bottom element but that would cost you O(NLogN) because of sorting, getting element in array with index is O(1) operation.