

Lesson



ArrayList in Java



Pre-Requisites

- Basics of Java
- Arrays in Java

List of Concepts Involved

- Collections in Java
- The Collection Interface
- The List Interface
- ArrayList Class
- Methods of ArrayList
- Problems based on ArrayList

Topic – Collections in Java

- Until now, we have understood the concept of one dimensional arrays and multi- dimensional arrays and solved some problems based on these concepts. We have also understood the concept of object creation for the non-primitive data types.
- If we know the count of objects to be created. Then, in that case, an array of such objects is created and all the objects are stored in that array. Now, if there is a situation where we do not have information about the count of objects to be stored and in which format, then we have to use the feature of java called “Collections”.
- A group of individual objects which are represented as a single unit is known as the collection of the objects.
- In other words, Java Collection is a **single unit of objects**.
- This collections framework provides a set of interfaces and classes to implement various data structures and algorithms.

What is the interface ?(Will be explained in oops lectures)

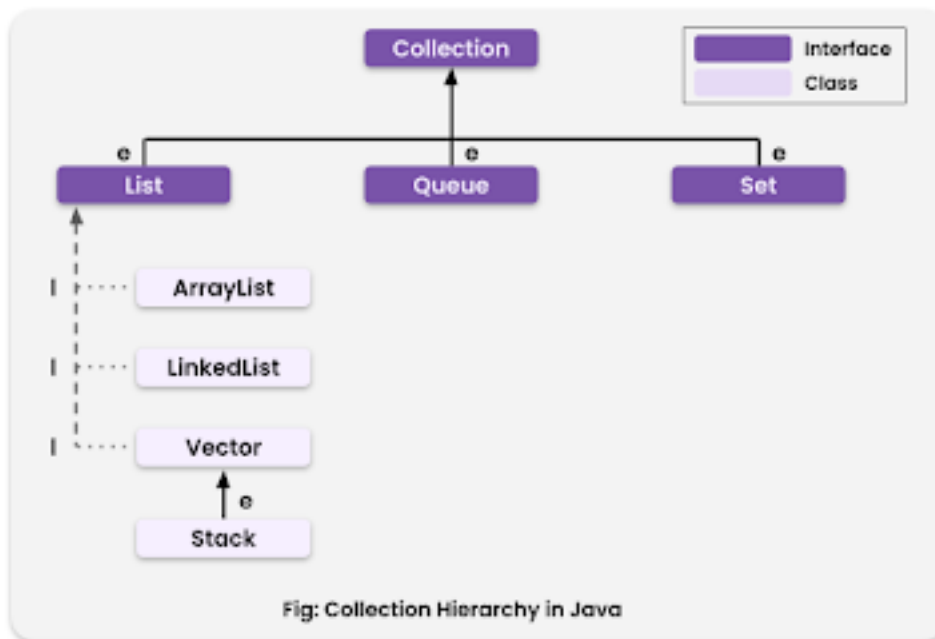
- An interface in Java is a blueprint of a class.
- It has static constants and abstract() methods(The abstract keyword is **a non-access modifier, used for classes and methods**. Class: An abstract class is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class). Method: An abstract method can only be used in an abstract class, and it does not have a body.,).
- It is used to describe a behavior that classes must implement, similar to protocols.
- We can't create object of interfaces.

What is Framework ?

- A framework in programming is a structure of ready-made components or solutions that are customized in order to speed up development.
- Using frameworks saves time and reduces the risk of errors. You don't need to write everything from scratch.

Topic – Collection Interface

- It is the root interface of the Java collections framework.
- It is a member of the Java collections framework.
- It is a part of java.util package.
- This interface cannot be implemented directly, it is implemented through its subinterfaces like List, Queue and Set.
- ArrayList class implements the List interface which is a subinterface of the Collection Interface.



e-> extends & i-> implements

Here, we have not shown the hierarchy of Queue and Set Interface as it will be discussed in coming lectures.

Topic–The List Interface

- It is an ordered collection that allows us to add and remove elements like an array.
- In Java, we must import java.util.List package in order to use List

Java classes which implements List Interface are–

- ArrayList
- LinkedList
- Vector
- Stack

Here, we will focus on ArrayList class only

Topic – ArrayList Class

- Before going towards the ArrayList class, let us recap about Arrays. The Array is a fixed number of groups of Similar kinds of objects placed at a continuous memory location. Similarly, ArrayList is a class which holds the object of the same kind in the order of insertion without any limit to the number of objects to store, i.e. we can say that Array is of fixed size and List is of dynamic sizing.
- ArrayList class extends AbstractList class and implements List interface.
- Syntax to create ArrayList

```
List<AnyClass> list = new ArrayList<AnyClass>();
```
- It uses a dynamic array data structure to store objects and elements.
- It allows duplicate objects and elements.
- It maintains the insertion order.
- It is non-synchronized.
- Its elements/objects can be accessed randomly.

Methods of ArrayList

Method	Description
void add(int index, E element)	This is used to insert the specified element at the specified position in a list.
add(E e)	It is used to append the specified element at the end of a list.
addAll(Collection<? extends E> c)	It is used to append 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.
addAll(int index, Collection<? extends E> c)	It is used to append all the elements in the specified collection, starting at the specified position of the list.
clear()	It is used to remove all of the elements from this list.
E remove(int index)	It is used to remove the element present at the specified position in the list
boolean remove(Object o)	It is used to remove the first occurrence of the specified element

<code>ensureCapacity?(int minCapacity)</code>	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
<code>boolean isEmpty()</code>	It returns true if the list is empty, otherwise false.
<code>listIterator()</code>	Returns a list iterator over the elements in this list
<code>int lastIndexOf(Object o)</code>	It is used to return the index in this list of the last occurrence of the specified element, or -1 if the list does not contain this element.
<code>Object[] toArray()</code>	It is used to return an array containing all of the elements in this list in the correct order.
<code><T> T[] toArray(T[] a)</code>	It is used to return an array containing all of the elements in this list in the correct order.
<code>boolean contains(Object o)</code>	It returns true if the list contains the specified element.
<code>int indexOf(Object o)</code>	It is used to return the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element.
<code>boolean removeAll(Collection<?> c)</code>	It is used to remove all the elements from the list.
<code>boolean removeIf(Predicate<? super E> filter)</code>	It is used to remove all the elements from the list that satisfies the given predicate.
<code>protected void removeRange(int fromIndex, int toIndex)</code>	It is used to remove all the elements lies within the given range.
<code>void replaceAll(UnaryOperator<E> operator)</code>	It is used to replace all the elements from the list with the specified element.

<code>void retainAll(Collection<?> c)</code>	It is used to retain all the elements in the list that are present in the specified collection.
<code>E set(int index, E element)</code>	It is used to replace the specified element in the list, present at the specified position
<code>void sort(Comparator<? super E> c)</code>	It is used to sort the elements of the list on the basis of the specified comparator.
<code>Splitterator<E> spliterator()</code>	It is used to create a spliterator over the elements in a list.
<code>List<E> subList(int fromIndex, int toIndex)</code>	It is used to fetch all the elements that lies within the given range.
<code>int size()</code>	It is used to return the number of elements present in the list.
<code>void trimToSize()</code>	It is used to trim the capacity of this ArrayList instance to be the list's current size.

Examples of ArrayList:

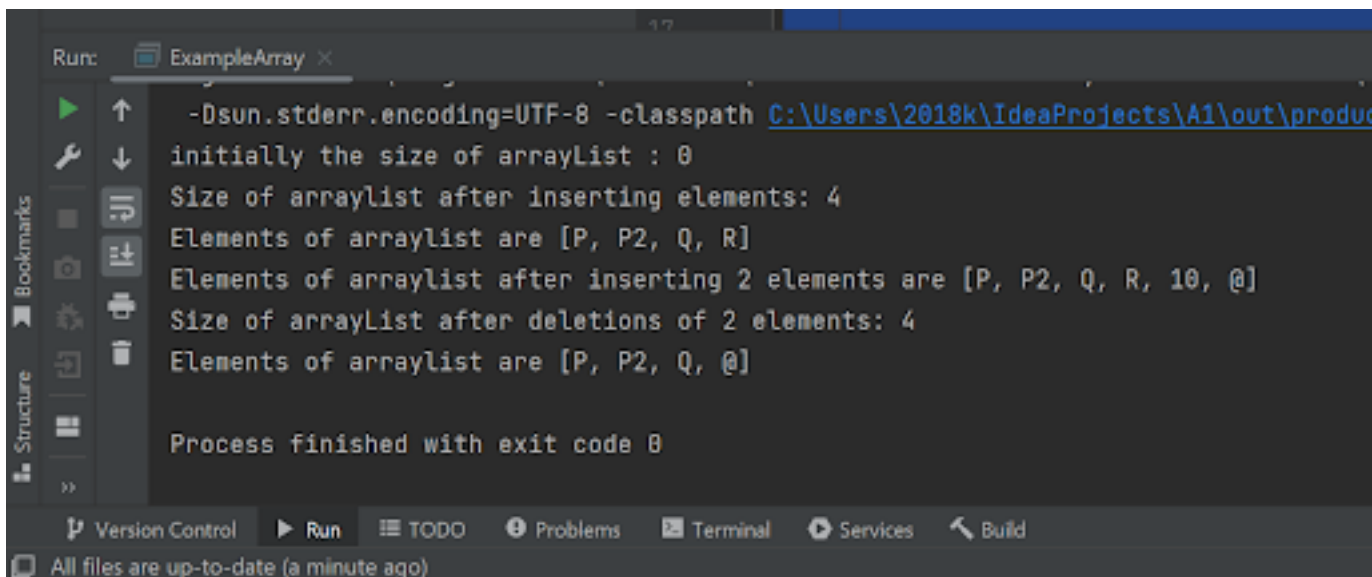
```
import java.util.ArrayList;
import java.io.*;
import java.util.*;
public class ExampleArray{

    public static void main(String args[]) {
        ArrayList arrayList = new ArrayList();
        System.out.println("initially the size of arrayList : " + arrayList.size());

        // add elements to the array list
        arrayList.add("P");
        arrayList.add("Q");
        arrayList.add("R");
        arrayList.add(1, "P2");
        System.out.println("Size of arraylist after inserting elements: " + arrayList.size());

        // display the array list
        System.out.println("Elements of arraylist are " + arrayList);
        arrayList.add("10");
        arrayList.add("@");
        System.out.println("Elements of arraylist after inserting 2 elements are " + arrayList);
        // Remove elements from the array list
        arrayList.remove("10");
        arrayList.remove(3);
        System.out.println("Size of arrayList after deletions of 2 elements: " + arrayList.size());
        System.out.println("Elements of arraylist are " + arrayList);

    }
}
```



```
Run: ExampleArray x
-Dsun.stderr.encoding=UTF-8 -classpath C:\Users\2018k\IdeaProjects\A1\out\product
initially the size of arrayList : 0
Size of arraylist after inserting elements: 4
Elements of arraylist are [P, P2, Q, R]
Elements of arraylist after inserting 2 elements are [P, P2, Q, R, 10, @]
Size of arrayList after deletions of 2 elements: 4
Elements of arraylist are [P, P2, Q, @]

Process finished with exit code 0
```

Topic – Problems based on ArrayList

Problem – Write a program to Reverse the given ArrayList.

Input – [0, 10, 3, 5, 22, 10]

Output – [10, 22, 5, 3, 10, 0]

Approach 1 – We can use the custom function. Here we will reverse the ArrayList by considering it as a simple array and traversing it in reverse order.

Steps –

- 1) Iterate the ArrayList from 0 to n/2 index
- 2) Swap the first and last element in each iteration
- 3) Once the loop ends, it return the reversed ArrayList

```
import java.util.ArrayList;
import java.io.*;
import java.util.*;

public class ExampleArray{
    public static void main(String[] args)
    {
        ArrayList<Integer> list = new ArrayList<Integer>();

        list.add(new Integer(0));
        list.add(new Integer(10));
        list.add(new Integer(3));
        list.add(new Integer(5));
        list.add(new Integer(22));
        list.add(new Integer(10));
        System.out.println("Before reverse: ");
        print(list);
        System.out.println("");
        for (int i = 0; i < list.size() / 2; i++) {
            Integer t = list.get(i);
            list.set(i, list.get(list.size() - i - 1));
            list.set(list.size() - i - 1, t);
        }
        System.out.println("After reverse: ");

        print(list);
    }

    public static void print(ArrayList<Integer> list)
    {
        for (int i = 0; i < list.size(); i++) {
            System.out.print(list.get(i) + " ");
        }
    }
}
```



```
.jar=52194:C:\Program Files\JetBrains\IntelliJ
-Dsun.stderr.encoding=UTF-8 -classpath C:\User

Before reverse:
0 10 3 5 22 10

After reverse:
10 22 5 3 10 0

Process finished with exit code 0
```

Time Complexity –

If there are N numbers in the given ArrayList arr, then complexity will be $O(N/2)$

Space Complexity –

If there are N numbers in arraylist array arr, then complexity will be $O(1)$.

Approach 2 – We can use the inbuilt Collections function. Collections is a class in Java which contains different in-built functions for sorting, reversing, etc. We will use the Collections.reverse() function to reverse the ArrayList.

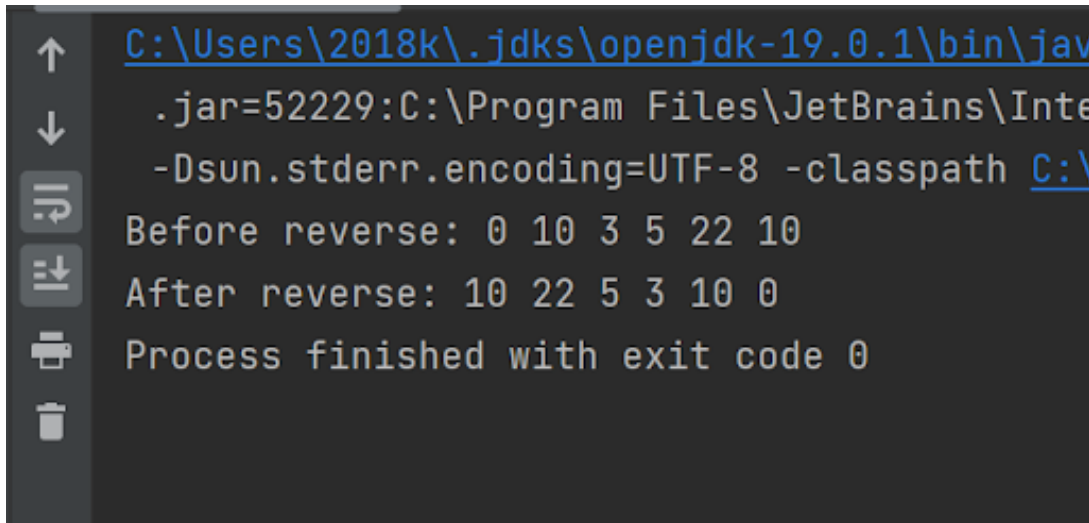
```
import java.io.*;
import java.util.*;

public class ExampleArray{
    public static void main(String[] args)
    {
        ArrayList<Integer> list = new ArrayList<Integer>();

        list.add(new Integer(0));
        list.add(new Integer(10));
        list.add(new Integer(3));
        list.add(new Integer(5));
        list.add(new Integer(22));
        list.add(new Integer(10));
        System.out.print("Before reverse: ");
        print(list);
        System.out.println("");

        Collections.reverse(list);
        System.out.print("After reverse: ");
        print(list);
    }
}
```

```
public static void print(ArrayList<Integer> list)
{
    for (int i = 0; i < list.size(); i++) {
        System.out.print(list.get(i) + " ");
    }
}
}
```



```
C:\Users\2018k\.jdk\openjdk-19.0.1\bin\jav
.jar=52229:C:\Program Files\JetBrains\Inte
-Dsun.stderr.encoding=UTF-8 -classpath C:\
Before reverse: 0 10 3 5 22 10
After reverse: 10 22 5 3 10 0
Process finished with exit code 0
```

Problem 2: Write a program to sort an ArrayList of Strings in descending order.

Solution

```
import java.io.*;
import java.util.*;

public class ExampleArray{
    public static void main(String args[]) {

        ArrayList<String> arraylist = new ArrayList<String>();
        arraylist.add("Akash");
        arraylist.add("Rahul");
        arraylist.add("Pankaj");
        arraylist.add("Manoj");

        /*Unsorted List: ArrayList content before sorting*/
        System.out.println("ArrayList Before Sorting:");
        for(String str: arraylist){
            System.out.println(str);
        }

        /* Sorting in decreasing (descending) order*/
        Collections.sort(arraylist, Collections.reverseOrder());

        /* Sorted List in reverse order*/
        System.out.println("ArrayList in descending order:");
        for(String str: arraylist){
            System.out.println(str);
        }
    }
}
```

```
Run: ExampleArray x
C:\Users\2018k\.jdk\openjdk-19.0.1\bin\
.jar=52234:C:\Program Files\JetBrains\I
-Dsun.stderr.encoding=UTF-8 -classpath
ArrayList Before Sorting:
Akash
Rahul
Pankaj
Manoj
ArrayList in descending order:
Rahul
Pankaj
Manoj
Akash

Process finished with exit code 0
```

Upcoming Class Teasers

- Complexity theory
- Space and time Complexity
- Calculating Time and Space Complexity