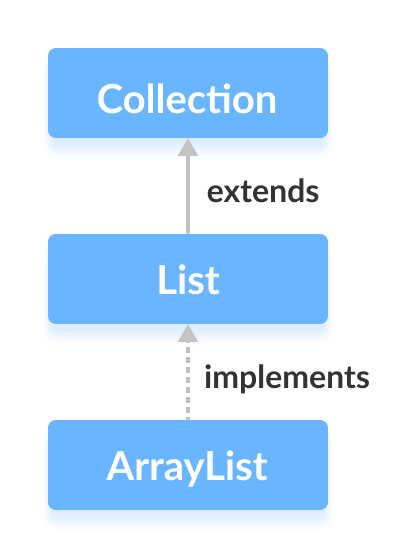
**Java ArrayList Class**

The ArrayList class is an implementation of the List interface that allows us to create resizable-arrays.



## Java Array Vs ArrayList

In Java, we need to declare the size of an array before we can use it. Once the size of an array is declared, it's hard to change it.

To handle this issue, we can use the ArrayList class. The ArrayList class present in the java.util package allows us to create resizable arrays.

Unlike arrays, array lists (objects of the ArrayList class) can automatically adjust its capacity when we add or remove elements from it. Hence, array lists are also known as dynamic arrays.

### Creating an ArrayList

Here is how we can create array lists in Java:

ArrayList<Type> arrayList= new ArrayList<>();

Here, Type indicates the type of an array list. For example,

// create Integer type arraylist

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

// create String type arraylist

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

In the above program, we have used Integer and String. Here, Integer is the corresponding wrapper class of the int type.

**A wrapper class is a class that wraps a primitive data type.** For example, the Integer class wraps the int type, the Float class wraps the float type, etc.

**Note:** We can not create array lists of primitive data types like int, float, char, etc. Instead, we have to use their corresponding wrapper class.

In the case of strings, String is a class and doesn't have a wrapper class. Hence, we have used String as it is.

We can also create array lists using the List interface. It's because the ArrayList class implements the List interface.

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

## **Methods of ArrayList**

### Add Elements to an ArrayList

**1. Using the add() method**

To add a single element to the array list, we use the add() method. For example,

import java.util.ArrayList;

class Main {

public static void main(String[] args){

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

// Add elements

animals.add("Dog");

animals.add("Cat");

animals.add("Horse");

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

}

}

**Output**

[Dog, Cat, Horse]

**2. Using index number**

We can also add elements to an array list using indexes. For example,

import java.util.ArrayList;

class Main {

public static void main(String[] args){

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

// Add elements

animals.add(0,"Dog");

animals.add(1,"Cat");

animals.add(2,"Horse");

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

}

}

**Output**

[Dog, Cat, Horse]

**3. Add elements of an array list to another array list**

To add all the elements of an array list to a new array list, we use the addAll() method. For example,

import java.util.ArrayList;

class Main {

public static void main(String[] args){

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

mammals.add("Dog");

mammals.add("Cat");

mammals.add("Horse");

System.out.println("Mammals: " + mammals);

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

animals.add("Crocodile");

// Add all elements of mammals in animals

animals.addAll(mammals);

System.out.println("Animals: " + animals);

}

}

**Output**

Mammals: [Dog, Cat, Horse]

Animals: [Crocodile, Dog, Cat, Horse]

## Initialize an ArrayList Using asList()

Unlike arrays, we cannot initialize array lists directly. However, we can use the asList() method of the Arrays class to achieve the same effect.

In order to use the asList() method, we must import the java.util.Arrays package first.

For example,

import java.util.ArrayList;

import java.util.Arrays;

class Main {

public static void main(String[] args) {

// Creating an array list

ArrayList<String> animals = new ArrayList<>(Arrays.asList("Cat", "Cow", "Dog"));

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

// Access elements of the array list

String element = animals.get(1);

System.out.println("Accessed Element: " + element);

}

}

**Output**

ArrayList: [Cat, Cow, Dog]

Accessed Elemenet: Cow

In the above example, notice the expression,

new ArrayList<>(Arrays.asList(("Cat", "Cow", "Dog"));

Here, we have first created an array of 3 elements: "Cat", "Cow", and "Dog". **Then, the asList() method is used to convert the array into an array list.**

String arr[]={"kkkk","jjjj"};

Integer arrint[]={2 , 3 , 4};

ArrayList<Integer> intlist = **new** ArrayList<>(Arrays.*asList*(arrint));

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

//List<String> listNames = Arrays.asList("Tom", "John", "Mary", "Peter", "David");

//ArrayList<String> animals = new ArrayList<>(Arrays.asList("Cat", "Cow", "Dog"));

ArrayList<String> str=**new** ArrayList<>(Arrays.*asList*(arr));

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

### Access ArrayList Elements

**1. Using get() Method**

To randomly access elements of an array list, we use the get() method. For example,

import java.util.ArrayList;

class Main {

public static void main(String[] args) {

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

// Add elements in the array list

animals.add("Dog");

animals.add("Horse");

animals.add("Cat");

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

// Get the element from the array list

String str = animals.get(0);

System.out.print("Element at index 0: " + str);

}

}

**Output**

ArrayList: [Dog, Horse, Cat]

Element at index 0: Dog

**2. Using iterator() Method**

To sequentially access elements of an array list, we use the iterator() method. We must import java.util.Iterator package to use this method. For example,

import java.util.ArrayList;

import java.util.Iterator;

class Main {

public static void main(String[] args){

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

// Add elements in the array list

animals.add("Dog");

animals.add("Cat");

animals.add("Horse");

animals.add("Zebra");

// Create an object of Iterator

Iterator<String> iterate = animals.iterator();

System.out.print("ArrayList: ");

// Use methods of Iterator to access elements

while(iterate.hasNext()){

System.out.print(iterate.next());

System.out.print(", ");

}

}

}

**Output**

ArrayList: Dog, Cat, Horse, Zebra,

**Note**:

* hasNext() returns true if there is a next element in the array list.
* next() returns the next element in the array list

### Change ArrayList Elements

To change elements of an array list, we can use the set() method. For example,

import java.util.ArrayList;

class Main {

public static void main(String[] args) {

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

// Add elements in the array list

animals.add("Dog");

animals.add("Cat");

animals.add("Horse");

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

// Change the element of the array list

animals.set(2, "Zebra"); // set method use for replacing

System.out.println("Modified ArrayList: " + animals);

}

}

**Output**

ArrayList: [Dog, Cat, Horse]

Modified ArrayList: [Dog, Cat, Zebra]

### Remove ArrayList Elements

**1. Using remove() Method**

To remove an element from an array list, we can use the remove() method.

import java.util.ArrayList;

class Main {

public static void main(String[] args) {

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

// Add elements in the array list

animals.add("Dog");

animals.add("Cat");

animals.add("Horse");

System.out.println("Initial ArrayList: " + animals);

// Remove element from index 2

String str = animals.remove(2);

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

System. out.println("Removed Element: " + str);

}

}

**Output**

Initial ArrayList: [Dog, Cat, Horse]

Final ArrayList: [Dog, Cat]

Removed Element: Horse

**2. Using removeAll() method**

To remove all elements from an array list, we use the removeAll() method. For example,

import java.util.ArrayList;

class Main {

public static void main(String[] args) {

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

// Add elements in the ArrayList

animals.add("Dog");

animals.add("Cat");

animals.add("Horse");

System.out.println("Initial ArrayList: " + animals);

// Remove all the elements

animals.removeAll(animals);

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

}

}

**Output**

Initial ArrayList: [Dog, Cat, Horse]

Final ArrayList: []

**3. Using clear() Method**

We can also use the clear() method to remove all elements from an array list. For example,

import java.util.ArrayList;

class Main {

public static void main(String[] args) {

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

// Add elements in the array list

animals.add("Dog");

animals.add("Cat");

animals.add("Horse");

System.out.println("Initial ArrayList: " + animals);

// Remove all the elements

animals.clear();

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

}

}

**Output**

Initial ArrayList: [Dog, Cat, Horse]

Final ArrayList: []

**Note:** The clear() method is more efficient than the removeAll() method.

## Loop Through an ArrayList

**1. Using for loop**

import java.util.ArrayList;

class Main {

public static void main(String[] args) {

// Creating an array list

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

animals.add("Cow");

animals.add("Cat");

animals.add("Dog");

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

// Using for loop

System.out.println("Accessing individual elements: ");

for(int i = 0; i < animals.size(); i++) {

System.out.print(animals.get(i));

System.out.print(", ");

}

}

}

**Output**

ArrayList: [Cow, Cat, Dog]

Accessing individual elements:

Cow, Cat, Dog,

**2. Using the forEach loop**

import java.util.ArrayList;

class Main {

public static void main(String[] args) {

// Creating an array list

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

animals.add("Cow");

animals.add("Cat");

animals.add("Dog");

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

// Using forEach loop

System.out.println("Accessing individual elements: ");

for(String animal : animals) {

System.out.print(animal);

System.out.print(", ");

}

}

}

**Output**

ArrayList: [Cow, Cat, Dog]

Accessing individual elements:

Cow, Cat, Dog,

In both examples, we have accessed individual elements of an array list using loops.

## Get ArrayList Length

To get the length of the array list, we use the size() method. For example,

import java.util.ArrayList;

class Main {

public static void main(String[] args) {

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

// Adding elements in the arrayList

animals.add("Dog");

animals.add("Horse");

animals.add("Cat");

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

// getting the size of the arrayList

System.out.println("Size: " + animals.size());

}

}

**Output**

ArrayList: [Dog, Horse, Cat]

Size: 3

## Sort Elements of an ArrayList

To sort elements of an array list, we use the sort() method of the Collections class. In order to use it, we must import the java.util.Collections package first.

By default, the sorting occurs either alphabetically or numerically in ascending order. For example,

import java.util.ArrayList;

import java.util.Collections;

class Main {

public static void main(String[] args){

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

// Add elements in the array list

animals.add("Horse");

animals.add("Zebra");

animals.add("Dog");

animals.add("Cat");

System.out.println("Unsorted ArrayList: " + animals);

// Sort the array list

Collections.sort(animals);

System.out.println("Sorted ArrayList: " + animals);

}

}

**Output**

Unsorted ArrayList: [Horse, Zebra, Dog, Cat]

Sorted ArrayList: [Cat, Dog, Horse, Zebra]

To learn more about sorting array list, visit Java ArrayList sort.

## Java ArrayList To Array

In Java, we can convert array lists into arrays using the toArray() method. For example,

import java.util.ArrayList;

class Main {

public static void main(String[] args) {

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

// Add elements in the array list

animals.add("Dog");

animals.add("Cat");

animals.add("Horse");

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

// Create a new array of String type

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

// Convert ArrayList into an array

animals.toArray(arr);

System.out.print("Array: ");

for(String item:arr) {

System.out.print(item+", ");

}

}

}

**Output**

ArrayList: [Dog, Cat, Horse]

Array: Dog, Cat, Horse,

## Java Array to ArrayList

We can also convert arrays into array lists. For that, we can use the asList() method of the Arrays class.

To use asList(), we must import the java.util.Arrays package first. For example,

import java.util.ArrayList;

import java.util.Arrays;

class Main {

public static void main(String[] args) {

// Create an array of String type

String[] arr = {"Dog", "Cat", "Horse"};

System.out.print("Array: ");

// Print array

for(String str: arr) {

System.out.print(str);

System.out.print(" ");

}

// Create an ArrayList from an array

ArrayList<String> animals = new ArrayList<>(Arrays.asList(arr));

System.out.println("\nArrayList: " + animals);

}

}

**Output**

Array: Dog, Cat, Horse

ArrayList: [Dog, Cat, Horse]

In the above program, we first created an array arr of the String type.

We then converted the array into an array list using the asList() method.

## Java ArrayList to String

To convert an array list into a String, we can use the toString() method. For example,

import java.util.ArrayList;

class Main {

public static void main(String[] args) {

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

// Add elements in the ArrayList

animals.add("Dog");

animals.add("Cat");

animals.add("Horse");

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

// Convert ArrayList into an String

String str = animals.toString();

System.out.println("String: " + str);

}

}

**Output**

ArrayList: [Dog, Cat, Horse]

String: [Dog, Cat, Horse]

**Note:** toString() converts the whole array list into a single String.

## Other ArrayList Methods

|  |  |
| --- | --- |
| Methods | Descriptions |
| clone() | Creates a new array list with the same element, size, and capacity. |
| contains() | Searches the array list for the specified element and returns a boolean result. |
| ensureCapacity() | Specifies the total element the array list can contain. |
| isEmpty() | Checks if the array list is empty. (Returns true and false) |
| indexOf() | Searches a specified element in an array list and returns the index of the element. |
| trimToSize() | Reduces the capacity of an array list to its current size. |