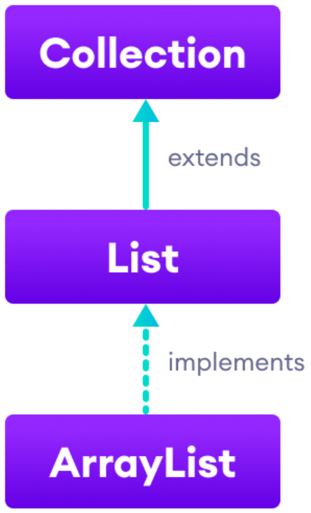
Java ArrayList Class

In this tutorial, we will learn about the Java ArrayList class. We will learn about different ArrayList operations and methods with the help of examples.

The `ArrayList` class of the Java collections framework provides the functionality of **resizable-arrays**.

It implements the `List` interface.



Java ArrayList Implementation

Java ArrayList Vs Array

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. It allows us to create resizable arrays.

Unlike arrays, arraylists can automatically adjust its capacity when we add or remove elements from it. Hence, arraylists are also known as **dynamic arrays**.

Creating an ArrayList

Before using `ArrayList`, we need to import the `java.util.ArrayList` package first. Here is how we can create arraylists in Java:

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

Here, `Type` indicates the type of an arraylist. 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` not int. It is because we cannot use primitive types while creating an arraylist. Instead, we have to use the corresponding wrapper classes.

Here, `Integer` is the corresponding wrapper class of `int`. To learn more, visit the Java wrapper class.

Example: Create ArrayList in Java

```
import java.util.ArrayList;

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

    // create ArrayList
    ArrayList<String> languages = new ArrayList<>();

    // Add elements to ArrayList
    languages.add("Java");
    languages.add("Python");
    languages.add("Swift");
    System.out.println("ArrayList: " + languages);
  }
}
```

Output

```
ArrayList: [Java, Python, Swift]
```

In the above example, we have created an `ArrayList` named `languages`.

Here, we have used the `add()` method to add elements to the arraylist. We will learn more about the `add()` method later in this tutorial.

> **Note**: We can also create an arraylist using the `List` interface. It's because the `ArrayList` class implements the `List` interface.

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

Basic Operations on Java ArrayList

The `ArrayList` class provides various methods to perform different operations on arraylists. We will look at some commonly used arraylist operations in this tutorial:

- Add elements
- Access elements
- Change elements
- Remove elements

---

1. Add Elements to an ArrayList

To add a single element to the arraylist, we use the `add()` method of the `ArrayList` class. For example,

```java
import java.util.ArrayList;

class Main {
  public static void main(String[] args){
    // create ArrayList
    ArrayList<String> languages = new ArrayList<>();

    // add() method without the index parameter
    languages.add("Java");
    languages.add("C");
    languages.add("Python");
    System.out.println("ArrayList: " + languages);

    // add() method with the index parameter
    languages.add(1, "JavaScript");
    System.out.println("Updated ArrayList: " + languages);
  }
}
```

Output

```
ArrayList: [Java, C, Python]
Updated ArrayList: [Java, JavaScript, C, Python]
```

In the above example, we have created an `ArrayList` named `languages`. Here, we have used the `add()` method to add elements to `languages`.

Notice the statement,

```
languages.add(1, "JavaScript");
```

Here, we have used the **index number** parameter. It is an optional parameter that specifies the position where the new element is added.

To learn more, visit the Java ArrayList add().

We can also add elements of a collection to an arraylist using the Java ArrayList addAll() method.

---

2. Access ArrayList Elements

To access an element from the arraylist, we use the `get()` method of the `ArrayList` class. For example,

```java
import java.util.ArrayList;

class Main {
  public static void main(String[] args) {
    ArrayList<String> animals = new ArrayList<>();

    // add elements in the arraylist
    animals.add("Cat");
    animals.add("Dog");
    animals.add("Cow");
    System.out.println("ArrayList: " + animals);

    // get the element from the arraylist
    String str = animals.get(1);
    System.out.print("Element at index 1: " + str);
  }
}
```

Output

```
ArrayList: [Cat, Dog, Cow]
Element at index 1: Dog
```

In the above example, we have used the `get()` method with parameter `1`. Here, the method returns the element at **index 1**.

To learn more, visit the [Java ArrayList get()](#).

We can also access elements of the `ArrayList` using the `iterator()` method. To learn more, visit [Java ArrayList iterator()](#).

3. Change ArrayList Elements

To change element of the arraylist, we use the `set()` method of the `ArrayList` class. For example,

```
import java.util.ArrayList;

class Main {
  public static void main(String[] args) {
    ArrayList<String> languages = new ArrayList<>();

    // add elements in the array list
    languages.add("Java");
    languages.add("Kotlin");
    languages.add("C++");
    System.out.println("ArrayList: " + languages);

    // change the element of the array list
    languages.set(2, "JavaScript");
    System.out.println("Modified ArrayList: " + languages);
  }
}
```

Output

```
ArrayList: [Java, Kotlin, C++]
Modified ArrayList: [Java, Kotlin, JavaScript]
```

In the above example, we have created an `ArrayList` named `languages`. Notice the line,

```
language.set(2, "JavaScript");
```

Here, the `set()` method changes the element at **index 2** to `JavaScript`.

To learn more, visit the Java ArrayList set().

---

4. Remove ArrayList Elements

To remove an element from the arraylist, we can use the `remove()` method of the `ArrayList` class. For example,

```java
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);

    // aemove element from index 2
    String str = animals.remove(2);
    System.out.println("Updated ArrayList: " + animals);
    System.out.println("Removed Element: " + str);
  }
}
```

Output

```
ArrayList: [Dog, Cat, Horse]
Updated ArrayList: [Dog, Cat]
Removed Element: Horse
```

Here, the `remove()` method takes the **index number** as the parameter. And, removes the element specified by the **index number**.

To learn more, visit the [Java ArrayList remove()](#).

We can also remove all the elements from the arraylist at once. To learn more, visit

- Java ArrayList removeAll()
- Java ArrayList clear()

Methods of ArrayList Class

In previous section, we have learned about the `add()`, `get()`, `set()`, and `remove()` method of the `ArrayList` class.

Besides those basic methods, here are some more `ArrayList` methods that are commonly used.

Methods	Descriptions
size()	Returns the length of the arraylist.
sort()	Sort the arraylist elements.
clone()	Creates a new arraylist with the same element, size, and capacity.
contains()	Searches the arraylist for the specified element and returns a boolean result.
ensureCapacity()	Specifies the total element the arraylist can contain.
isEmpty()	Checks if the arraylist is empty.
indexOf()	Searches a specified element in an arraylist and returns the index of the element.

If you want to learn about all the different methods of arraylist, visit Java ArrayList methods.

Iterate through an ArrayList

We can use the Java for-each loop to loop through each element of the arraylist. For example,

```
import java.util.ArrayList;

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

    // creating an array list
    ArrayList<String> languages = new ArrayList<>();
    languages.add("Cow");
    languages.add("Cat");
    languages.add("Dog");
    System.out.println("ArrayList: " + languages);

    // iterate using for-each loop
    System.out.println("Accessing individual elements:  ");

    for (String language : languages) {
      System.out.print(language);
      System.out.print(", ");
    }
  }
}
```

Output

```
ArrayList: [Cow, Cat, Dog]
Accessing individual elements:
Cow, Cat, Dog,
```

Java ArrayList To Array Conversion

We can convert the `ArrayList` into an array using the `toArray()` method. For example,

```
import java.util.ArrayList;

class Main {
  public static void main(String[] args) {
    ArrayList<String> languages = new ArrayList<>();

    // add elements in the array list
    languages.add("Java");
    languages.add("Python");
    languages.add("C++");
    System.out.println("ArrayList: " + languages);

    // create a new array of String type
    String[] arr = new String[languages.size()];

    // convert ArrayList into an array
    languages.toArray(arr);
    System.out.print("Array: ");

    // access elements of the array
    for (String item : arr) {
      System.out.print(item + ", ");
    }
  }
}
```

Output

```
ArrayList: [Java, Python, C++]
Array: Java, Python, C++,
```

In the above example, we have created an arraylist named `languages`. Notice the statement,

```
languages.toArray(arr);
```

Here, the `toArray()` method converts the arraylist into an array and stores it in `arr`. To learn more, visit Java ArrayList toArray().

Java Array to ArrayList Conversion

We can also convert the array into an arraylist. For that, we 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 = { "Java", "Python", "C++" };
    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> languages = new ArrayList<>(Arrays.asList(arr));
    System.out.println("\nArrayList: " + languages);
  }
}
```

Output

```
Array: Java Python C++
ArrayList: [Java, Python, C++]
```

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

```
Arrays.asList(arr)
```

Here, the `asList()` method converts the array into an arraylist.

> **Note**: We can also use the `Arrays.asList()` method to create and initialize the arraylist in a single line. For example,
>
> ```
>  // create and initialize arraylist
> ArrayList<String> animals = new ArrayList<>(Arrays.asList("Cat", "Cow", "Dog"));
> ```

Java ArrayList to String Conversion

We can use the `toString()` method of the `ArrayList` class to convert an arraylist into a string. For example,

```
import java.util.ArrayList;

class Main {
  public static void main(String[] args) {
    ArrayList<String> languages = new ArrayList<>();

    // add elements in the ArrayList
    languages.add("Java");
    languages.add("Python");
    languages.add("Kotlin");
    System.out.println("ArrayList: " + languages);

    // convert ArrayList into a String
    String str = languages.toString();
    System.out.println("String: " + str);
  }
}
```

Output

```
ArrayList: [Java, Python, Kotlin]
String: [Java, Python, Kotlin]
```

Here, the `toString()` method converts the whole arraylist into a single string.