[**https://www.geeksforgeeks.org/array-vs-arraylist-in-java/**](https://www.geeksforgeeks.org/array-vs-arraylist-in-java/)

**Array vs ArrayList in Java**

* Difficulty Level : [Easy](https://www.geeksforgeeks.org/easy/)
* Last Updated : 11 May, 2022

 Read

 Discuss

Let us discuss the concept of the **arrays** and **ArrayList** briefly in the header to incorporate the understanding in java programs later landing onto the conclusive differences between them. As we all are aware of that arrays are linear data structures providing functionality to add elements in a continuous manner in memory address space whereas ArrayList is a class belonging to the Collection framework. Being a good programmer one is already aware of using ArrayList over arrays despite knowing the differences between these two. Now moving ahead even with ArrayList there comes a functionality to pass the type of datatype of elements that are supposed to be stored in the ArrayList be it an object, string, integer, double, float, etc.

**Note:** As a side note, ArrayList in Java can be seen as similar to [vector in C++](https://www.geeksforgeeks.org/vector-in-cpp-stl/).

**Methods of Creating Arrays**

In Java, the following are two different ways to create an array.

1. Simple fixed-sized arrays
2. Dynamically sized arrays

int arr[] = new int[10]

**Syntax:** Declaring a static array

It can be further defined by two types:

* **Type 1**: Declaring and initializing at the same time
* **Type 2:** Declaring than initializing elements later.

**Type 1**

Type array\_name [array\_size] ;

Type array\_name = { Element1, Element2, Element3, Element4,...., ElementN } ;

// It is preferable if we have very limited array elements

**Type 2**

int arr [100] ;

// This does means we are declaring a memory block named 'arr'

// which is containing continuous 100 block associated in it

**Note:** arr(0) returns the first element of the array so it does mean that if we try to print out arr(0) then we will get Element1. It is very important statement and is left unveiliable when it comes to deep understanding of memory storage in arrays.

Now let us dwell on the next concept of ArrayList that is as follows

**Syntax:** Declaring an Arraylist

Arraylist<Type> al = new ArrayList<Type> ;

// Here Type is the type of elements in ArrayList to be created

**Note:** ArrayList in Java (equivalent to vector in C++) having dynamic size. It can be shrunk or expanded based on size. ArrayList is a part of the collection framework and is present in [java.util package](https://www.geeksforgeeks.org/java-util-package-java/).

Now let us illustrate examples with the help of differences between Array and ArrayList

**Base 1:** An array is a basic functionality provided by Java. ArrayList is part of the collection framework in Java. Therefore array members are accessed using [], while ArrayList has a set of methods to access elements and modify them.

**Example:**

|  |
| --- |
| // Java program to demonstrate differences between  // Array and ArrayList    // Importing required classes  import java.util.ArrayList;  import java.util.Arrays;    // Main class  class GFG {        // Main driver method      public static void main(String args[])      {          // Input array          int[] arr = new int[2];          arr[0] = 1;          arr[1] = 2;            // Printing first element of array          System.out.println(arr[0]);            // ArrayList          // Creating an arrayList with initial capacity          // say bi it 2          ArrayList<Integer> arrL = new ArrayList<Integer>(2);            // Adding elements to ArrayList          // using add() method          arrL.add(1);          arrL.add(2);            // Printing alongside accessing          // elements of ArrayList          System.out.println(arrL.get(0));      }  } |

**Output**

1

1

**Base 2:** The array is a fixed-size data structure while ArrayList is not. One need not mention the size of the ArrayList while creating its object. Even if we specify some initial capacity, we can add more elements.

**Example:**

|  |
| --- |
| // Java program to demonstrate differences between  // Array and ArrayList    // Importing required classes  import java.util.ArrayList;  import java.util.Arrays;    // Main class  class GFG {      // Main driver method      public static void main(String args[])      {          // Normal Array          // Need to specify the size for array          int[] arr = new int[3];          arr[0] = 1;          arr[1] = 2;          arr[2] = 3;            // We cannot add more elements to array arr[]            // ArrayList          // Need not to specify size            // Declaring an Arraylist of Integer type          ArrayList<Integer> arrL = new ArrayList<Integer>();            // Adding elements to ArrayList object          arrL.add(1);          arrL.add(2);          arrL.add(3);          arrL.add(4);            // We can add more elements to arrL            // Print and display Arraylist elements          System.out.println(arrL);          // Print and display array elements          System.out.println(Arrays.toString(arr));      }  } |

**Output**

[1, 2, 3, 4]

[1, 2, 3]

**Base 3:** An array can contain both primitive data types as well as objects of a class depending on the definition of the array. However, ArrayList only supports object entries, not the primitive data types.

**Note:** When we do arraylist.add(1) than it converts the primitive int data type into an Integer object which is as illustrated in below example

**Example:**

|  |
| --- |
| import java.util.ArrayList;  class Test  {      public static void main(String args[])      {         // allowed          int[] array = new int[3];            // allowed, however, need to be initialized          Test[] array1 = new Test[3];            // not allowed (Uncommenting below line causes          // compiler error)          // ArrayList<char> arrL = new ArrayList<char>();            // Allowed          ArrayList<Integer> arrL1 = new ArrayList<>();          ArrayList<String> arrL2 = new ArrayList<>();          ArrayList<Object> arrL3 = new ArrayList<>();            System.out.println("Successfully compiled and executed");      }  } |

**Output**

Successfully compiled and executed

**Base 4:** Since ArrayList can’t be created for primitive data types, members of ArrayList are always references to objects at different memory locations (See [this](https://www.geeksforgeeks.org/g-fact-46/) for details). Therefore in ArrayList, the actual objects are never stored at contiguous locations. References of the actual objects are stored at contiguous locations.

On the other hand, in the array, it depends whether the array is of primitive type or object type. In the case of primitive types, actual values are contiguous locations, but in the case of objects, allocation is similar to ArrayList. Java ArrayList supports many additional operations like indexOf(), [remove()](https://www.geeksforgeeks.org/arraylist-linkedlist-remove-methods-java-examples/), etc. These functions are not supported by Arrays.

We have implemented and seen the differences between them as perceived from outputs. Now let us wrap up the article by plotting conclusive differences in a tabular format a shown below as follows:

| **Base** | **Array** | **ArrayList** |
| --- | --- | --- |
| Dimensionality | It can be single-dimensional or multidimensional | It can only be single-dimensional |
| Traversing Elements | For and for each generally is used for iterating over arrays | Here iterator is used to traverse riverArrayList |
| Length | length keyword can give the total size of the array. | size() method is used to compute the size of ArrayList. |
| Size | It is static and of fixed length | It is dynamic and can be increased or decreased in size when required. |
| Speed | It is faster as above we see it of fixed size | It is relatively slower because of its dynamic nature |
| Primitive Datatype Storage | Primitive data types can be stored directly unlikely objects | Primitive data types are not directly added unlikely arrays, they are added indirectly with help of autoboxing and unboxing |
| Generics | They can not be added here hence type unsafe | They can be added here hence makingArrayList type-safe. |
| Adding Elements | Assignment operator only serves the purpose | Here a special method is used known as add() method |

This article is contributed by **Pranjal Mathur**. If you like GeeksforGeeks and would like to contri