# **Arrays in Java**

# **Arrays** **in Java** are one of the most fundamental data structures that allow us to store multiple values of the same type in a single variable. They are useful for storing and managing collections of data. Arrays in Java are objects.

* **Contiguous Memory Allocation (for Primitives):** Java array elements are stored in contiguous memory locations, which means that the elements are placed next to each other in memory.
* **Zero-based Indexing:** The first element of the array is at index 0.
* **Fixed Length:** Once an array is created, its size is fixed and cannot be changed.
* **Can Store Primitives & Objects:** Java arrays can hold both primitive types (like int, char, boolean, etc.) and objects (like String, Integer, etc.)
* Since arrays are objects in Java, we can find their length using the object property *length*.
* A Java array variable can also be declared like other variables with [ ] after the data type.

To declare an array in Java, use the following syntax:

*type[] arrayName;*

* **type**: The data type of the array elements (e.g., int, String).
* **arrayName**: The name of the array.

***int[] nums = new int[3];***

Memory:

**Stack:**

**nums** → reference to the array

**Heap:**

Actual array object: **[0, 0, 0]**

The **Arrays** class in the java.util **package** is a part of the **Java Collection Framework**. This class provides static methods to dynamically create and access **Java arrays**. It consists of only static methods and the methods of Object class. The methods of this class can be used by the class name itself.

## **Instantiating an Array in Java**

When an array is declared, only a reference of an array is created. To create or give memory to the array, you can allocate memory to the array using ***new*** keyword

var-name = new type [size];

//declaring array

**int intArray[];**

// allocating memory to array

**intArray = new int[20];**

// combining both statements in one

**int[] intArray = new int[20];**

## **Array Literal in Java**

**In a situation where the size of the array and variables of the array are already known, array literals can be used.**

**// Declaring array literal**

**int[] intArray = new int[]{ 1,2,3,4,5,6,7,8,9,10 };**

* **The length of this array determines the length of the created array.**
* **There is no need to write the new int[] part in the latest versions of Java.**

### **Accessing Java Array Elements using for Loop**

**Each element in the array is accessed via its index. The index begins with 0 and ends at (total array size)-1. All the elements of array can be accessed using Java for Loop.**

**// accessing the elements of the specified array**

**for (int i = 0; i < arr.length; i++)**

**System.out.println("Element at index " + i + " : "+ arr[i]);**

### **What happens if we try to access elements outside the array size?**

JVM throws **ArrayIndexOutOfBoundsException** to indicate that the array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of an array

**Multidimensional Arrays** can be defined in simple words as an array of arrays. Data in multidimensional arrays are stored in tabular form

* **data\_type**: Type of data to be stored in the array. For example: int, char, etc.
* **dimension**: The dimension of the array created. For example: 1D, 2D, etc.
* **array\_name**: Name of the array
* **size1, size2, …, sizeN**: Sizes of the dimensions respectively.
* **Declaration – Syntax:**

**data\_type[][] array\_name = new data\_type[x][y];**

For example: int[][] arr = new int[10][20];

* **Initialization – Syntax:**

**array\_name[row\_index][column\_index] = value;**

For example: arr[0][0] = 1;

// declaring **and** initializing 2D array

**int arr[][] = { { 2, 7, 9 }, { 3, 6, 1 }, { 7, 4, 2 } };**

### **1. Which is the direct superclass of an array in Java?**

*An Object is a direct superclass of an array in Java.*

### **2. Which Interfaces are implemented by Arrays in Java?**

*Every array type implements the interfaces Cloneable and java.io.Serializable.*

### **3. Can we alter the size of Array?**

*The size of the array cannot be altered(once initialized). However, an array reference can be made to point to another array.*

### ***1. Declaring an Array***

*The general form of array declaration is*

***Method 1:****type var-name[];*

***Method 2:****type[] var-name;*

*The element type determines the data type of each element that comprises the array. Like an array of integers, we can also create an array of other primitive data types like char, float, double, etc., or user-defined data types (objects of a class).*

***Note:*** *It is just how we can create an array variable,* ***no actual array exists****. It merely tells the compiler that this variable (int Array) will hold an array of the integer type.*

*Now, Let us provide memory storage to this created array.*

### ***2. Initialization an Array in Java***

*When an array is declared, only a reference of an array is created. The general form of new as it applies to one-dimensional arrays appears as follows:*

*var-name = new type [size];*

*Here, type specifies the type of data being allocated, size determines the number of elements in the array, and var-name is the name of the array variable that is linked to the array. To use new to allocate an array,* ***you must specify the type and number of elements to allocate.***

#### ***Example:***

*// declaring array  
 int intArray[];*

*// allocating memory to array  
 intArray = new int[20];*

*// combining both statements in one  
 int[] intArray = new int[20];*

***Note:*** *The elements in the array allocated by new will automatically be initialized to* ***zero*** *(for numeric types),* ***false*** *(for boolean), or* ***null*** *(for reference types). Do refer to default array values in Java.*

*Obtaining an array is a two-step process. First, you must declare a variable of the desired array type. Second, you must allocate the memory to hold the array, using new, and assign it to the array variable. Thus,* ***in Java****,* ***all arrays are dynamically allocated.***

### ***Array Literal in Java***

***In a situation where the size of the array and variables of the array are already known, array literals can be used.***

***// Declaring array literal   
 int[] intArray = new int[]{ 1,2,3,4,5,6,7,8,9,10 };***

* ***The length of this array determines the length of the created array.***
* ***There is no need to write the new int[] part in the latest versions of Java.***

### ***3. Accessing Java Array Elements using for Loop***

***Now , we have created an Array with or without the values stored in it. Access becomes an important part to operate over the values mentioned within the array indexes using the points mentioned below:***

* ***Each element in the array is accessed via its index.***
* ***The index begins with 0 and ends at (total array size)-1.***
* ***All the elements of the array can be accessed using Java for Loop.***

***Let us check the syntax of basic for loop to traverse an array:***

***// Accessing the elements of the specified array  
 for (int i = 0; i < arr.length; i++)  
 System.out.println(“Element at index ” + i + ” : “+ arr[i]);***

| ***Operation*** | ***Example*** |
| --- | --- |
| ***Sort*** | ***Arrays.sort(arr);*** |
| ***Search*** | ***Arrays.binarySearch(arr, key);*** |
| ***Copy*** | ***int[] copy = Arrays.copyOf(arr, len);*** |
| ***Fill*** | ***Arrays.fill(arr, 0);*** |