# **Arrays in Java**

# **In Java, Array** is a group of like-typed variables referred to by a common name. Arrays in Java work differently than they do in C/C++.

* In Java, all arrays are dynamically allocated.
* Arrays may be stored in contiguous memory [consecutive memory locations].
* Since arrays are objects in Java, we can find their length using the object property *length*. This is different from C/C++, where we find length using sizeof.
* A Java array variable can also be declared like other variables with [] after the data type.
* The variables in the array are ordered, and each has an index beginning with 0.
* Java arrays can also be used as a static field, a local variable, or a method parameter.

An array can **contain primitives (int, char, etc.)** and **object (or non-primitive) references of a class** depending on the definition of the array. In the case of primitive data types, the actual values might be stored in contiguous memory locations. In the case of class objects, the actual objects are stored in a heap segment.

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.*