**Array basics**

Normally, an array is a collection of similar type of elements which has a contiguous memory location.

Java array is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array. Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index, and so on.



**Why we need array?**

Arrays in Java are essential data structures that allow us to store and manage multiple elements of the same data type in a single variable. They provide efficient memory allocation and fast access to elements, making it easier to work with collections of data and perform various operations like sorting, searching, and iteration.

Creating, initializing, and accessing an Array

The general form of a one-dimensional array declaration is

**Syntax:**

Datatype var-name[];

OR

Datatype [] var-name;

**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 create an array like this: The general form of new as it applies to one-dimensional arrays appears as follows:

var-name = new type [size];

**Example:**

int intArray[]; //declaring array

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

**Array Literal**

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

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

// Declaring array literal

* 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]);

**Arrays of Objects**

An array of objects is created like an array of primitive-type data items in the following way.

Student[] arr = new Student[5]; //student is a user-defined class

**Syntax:**

1) data type[] arrName;

2) datatype arrName[];

3) datatype [] arrName;

**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:**

Multidimensional arrays are arrays of arrays with each element of the array holding the reference of other arrays. These are also known as Jagged Arrays. A multidimensional array is created by appending one set of square brackets ([]) per dimension.

Syntax :

datatype[1st dimension][2nd dimension][]……….[Nth dimension] array name = new datatype[size1][size2]….[size];

* datatype: 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.

**Examples:**

Two-dimensional array:

int[][] twoD\_arr = new int[10][20];

Three-dimensional array:

int[][][] threeD\_arr = new int[10][20][30];

Size of multidimensional arrays: The total number of elements that can be stored in a multidimensional array can be calculated by multiplying the size of all the dimensions.

**For example:**

The array int[][] x = new int[10][20] can store a total of (10\*20) = 200 elements. Similarly, array int[][][] x = new int[5][10][20] can store a total of (5\*10\*20) = 1000 elements.

* Application of Multi-Dimensional Array
* Multidimensional arrays are used to store the data in a tabular form. For example, storing the roll number and marks of a student can be easily done using multidimensional arrays. Another common usage is to store the images in 3D arrays.
* In dynamic programming questions, multidimensional arrays are used which are used to represent the states of the problem.
* Apart from these, they also have applications in many standard algorithmic problems like: Matrix Multiplication, Adjacency matrix representation in graphs, Grid search problems.

**Two – dimensional Array (2D-Array)**

Two – dimensional array is the simplest form of a multidimensional array. A two – dimensional array can be seen as an array of one – dimensional array for easier understanding.

**Indirect 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;

**Direct Method of Declaration: Syntax:**

data\_type[][] array\_name = {

{valueR1C1, valueR1C2, ....},

{valueR2C1, valueR2C2, ....}

};

For example: int[][] arr = {{1, 2}, {3, 4}};

**Accessing Elements of Two-Dimensional Arrays**

Elements in two-dimensional arrays are commonly referred to by x[i][j] where ‘i’ is the row number and ‘j’ is the column number.

**Syntax:**

x[row\_index][column\_index]

**For example:**

int[][] arr = new int[10][20];

arr[0][0] = 1;

**Representation of 2D array in Tabular Format:**

A two – dimensional array can be seen as a table with ‘x’ rows and ‘y’ columns where the row number ranges from 0 to (x-1) and column number ranges from 0 to (y-1). A two – dimensional array ‘x’ with 3 rows and 3 columns is shown below:



**Three – dimensional Array (3D-Array)**

Three – dimensional array is a complex form of a multidimensional array. A three-dimensional array can be seen as an array of two – dimensional array for easier understanding.

**Indirect Method of Declaration:**

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

For example: int[][][] arr = new int[10][20][30];

**Initialization – Syntax:**

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

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

**Direct Method of Declaration:**

data\_type[][][] array\_name = {

{

{valueA1R1C1, valueA1R1C2, ....},

{valueA1R2C1, valueA1R2C2, ....}

},

{

{valueA2R1C1, valueA2R1C2, ....},

{valueA2R2C1, valueA2R2C2, ....}

}

};

For example: int[][][] arr = { {{1, 2}, {3, 4}}, {{5, 6}, {7, 8}} };

Accessing Elements of Three-Dimensional Arrays

Elements in three-dimensional arrays are commonly referred by x[i][j][k] where ‘i’ is the array number, ‘j’ is the row number and ‘k’ is the column number.

**Syntax:**

x[array\_index][row\_index][column\_index]

**For example:**

int[][][] arr = new int[10][20][30];

arr[0][0][0] = 1;

Note: In arrays if size of array is N. Its index will be from 0 to N-1. Therefore, for row\_index 2, actual row number is 2+1 = 3.

**Representation of 3D array in Tabular Format:**

