Multidimensional Arrays in Java

Multidimensional Arrays can be defined in simple words as array of arrays. Data in multidimensional arrays are stored in tabular form (in row major order).

Syntax:

data_type[1st dimension][2nd dimension][]..[Nth dimension] array_name = new data_type[size1][size2]....[sizeN];

where:

- 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.

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.

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 Method of Declaration:

- Declaration Syntax:

data_type[][] array_name = new data_type[number of rows][number of columns];

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:

- Declaration as well as Initialization Syntax:

data_type[][] array_name = { {valueR1C1, valueR1C2, ...},
{valueR2C1, valueR2C2, ...} };

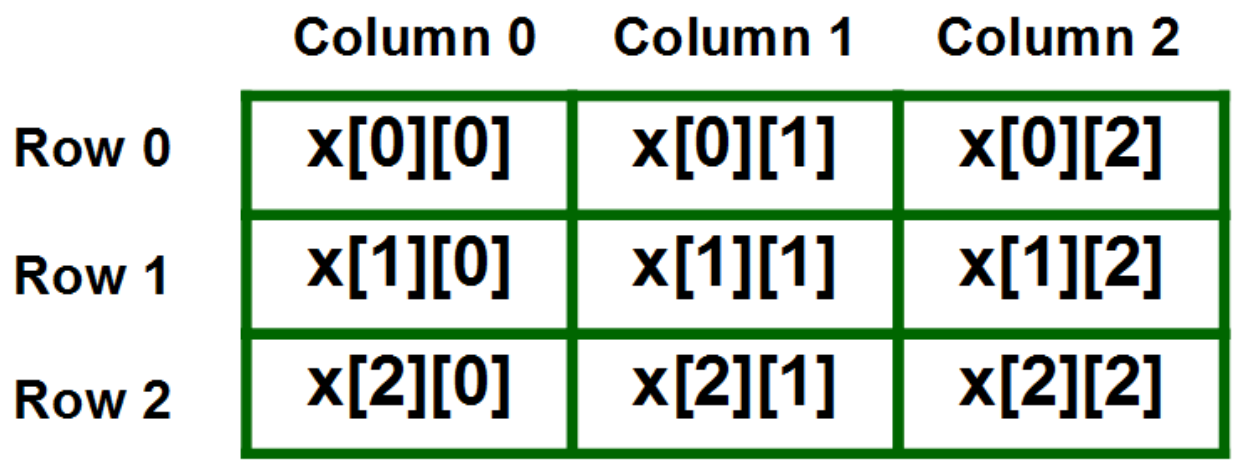
For example:
int[][] arr = {{1, 2}, {3, 4}};

Accessing Elements of Two-Dimensional Arrays

Elements in two-dimensional arrays are commonly referred by arr[i][j] where 'i' is the row number and 'j' is the column number.

Representation of 2D array in Tabular Format:

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



Print 2D array in tabular format: To output all the elements of a Two-Dimensional array, use nested for loops. For this two for loops are required, One to traverse the rows and another to traverse columns.

Example:

```
// Java Code to print a Two-dimensional Array
class GfG {
    public static void main(String[] args) {

        // Initializing a 2D array with 2 rows and 2 columns
```

```
        int[][] arr = {
            { 1, 2 },  // First row
            { 3, 4 }   // Second row
        };

        // Outer loop iterates over the rows of the 2D array
        for (int i = 0; i < 2; i++) {

            // Inner loop iterates over the columns of the current row
            for (int j = 0; j < 2; j++) {

                // Printing each element in the row with a space
                System.out.print(arr[i][j] + " ");
            }

            // New line after printing all columns in the current row
            System.out.println();
        }
    }
}
```

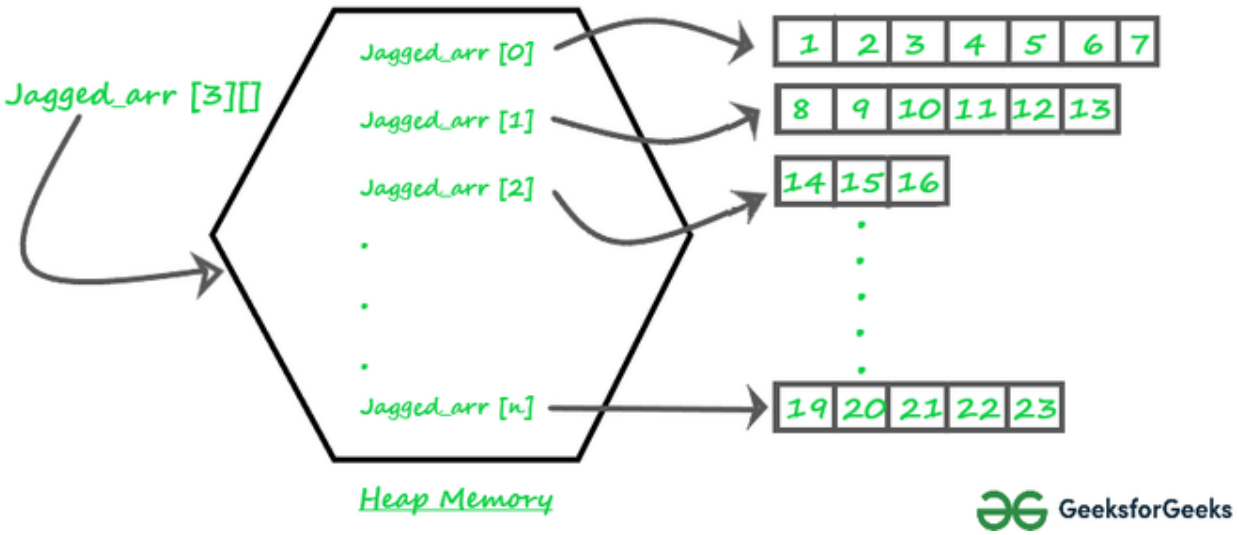
Output

```
1 2
3 4
```

Jagged Array

A jagged array is an array of arrays such that member arrays can be of different sizes, i.e., we can create a 2-D array but with a variable number of columns in each row.

Pictorial representation of Jagged array in Memory:



Declaration and Initialization of Jagged array :

data_type array_name[][] = new data_type[n][];
array_name[] = new data_type[n1]
array_name[] = new data_type[n2]
array_name[] = new data_type[n3]
.
.
.
array_name[] = new data_type[nk]
where n = Total number of rows and ni = number of columns in i-th row

Alternative, ways to Initialize a Jagged array :

int arr_name[][] = new int[][] { new int[] { 10, 20, 30, 40 },
new int[] { 50, 60, 70, 80, 90, 100 },
new int[] { 110, 120 } };

OR

int[][] arr_name = { new int[] { 10, 20, 30, 40 },
new int[] { 50, 60, 70, 80, 90, 100 },
new int[] { 110, 120 } };

OR

int[][] arr_name = { { 10, 20, 30, 40 },
{ 50, 60, 70, 80, 90, 100 },
{ 110, 120 } };

Following is example where i'th row has i columns, i.e., the first row has 1 element, the second row has two elements and so on.

```
// Program to demonstrate 2-D jagged array in Java
class GfG {
    public static void main(String[] args) {
        int m = 3;

        // Declaring 2-D array with 3 rows
        int arr[][] = new int[m][];

        // Creating a 2D array such that first row
        // has 1 element, second row has two
        // elements and so on.
        for (int i = 0; i < arr.length; i++) {
            arr[i] = new int[i + 1];

            for (int j = 0; j < arr[i].length; j++) {
                arr[i][j] = i;

                System.out.print(arr[i][j] + " ");
```

```
            }

            System.out.println();
        }
    }
}
```

Output

```
0
1 1
2 2 2
```

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:

- Declaration Syntax:

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:

- Declaration as well as Initialization Syntax:

data_type[][][] array_name = { { {valueA1R1C1, valueA1R1C2, ...},
{valueA1R2C1, valueA1R2C2, ...} },
{ {valueA2R1C1, valueA2R1C2, ...},
{valueA2R2C1, valueA2R2C2, ...} } };

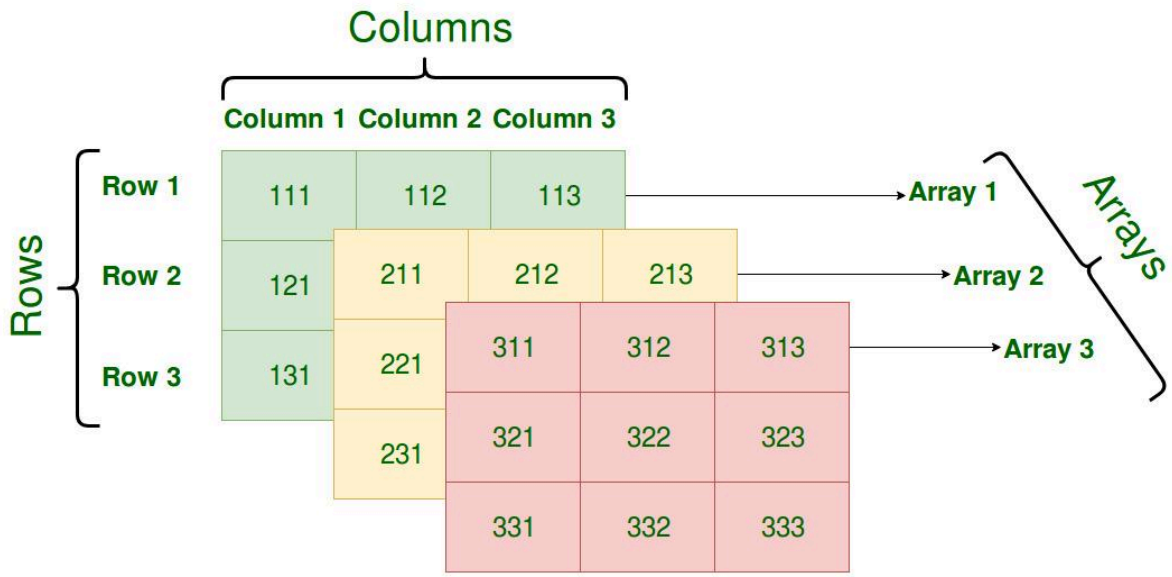
For example:
int[][][] arr = { {{1, 2}, {3, 4}}, {{5, 6}, {7, 8}} };

Accessing Elements of Three-Dimensional Arrays

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

Representation of 3D array in Tabular Format:

A three-dimensional array can be visualized as a table of arrays with n rows, m columns, and l arrays. The row numbers range from `0` to `(n-1)`, the column numbers range from `0` to `(m-1)`, and the array numbers range from `0` to `(l-1)`:
A three-dimensional array with l = 3 arrays, each containing n = 3 rows and m = 3 columns, is shown below:



Print 3D array in tabular format: To output all the elements of a Three-Dimensional array, use nested for loops. For this three for loops are required, One to traverse the arrays, second to traverse the rows and another to traverse columns.

Example:

```
// Java Code to print a Three-dimensional Array
class GfG {
    public static void main(String[] args) {

        // Initializing a 3D array with 2 arrays, each containing 2 rows
and 2 columns
        int[][][] arr = {
            { { 1, 2 }, { 3, 4 } },  // First 2D array
            { { 5, 6 }, { 7, 8 } }   // Second 2D array
        };

        // Outer loop iterates over the 2 arrays
        for (int i = 0; i < 2; i++) {

            // Middle loop iterates over the rows in each 2D array
            for (int j = 0; j < 2; j++) {

                // Inner loop iterates over the columns in each row
```

```
                for (int k = 0; k < 2; k++) {

                    // Printing each element in the 3D array
                    System.out.print(arr[i][j][k] + " ");
                }

                // New line after printing all columns in a row
                System.out.println();
            }

            // New line after printing all rows in one 2D array
            System.out.println();
        }
    }
}
```

Output

```
1 2
3 4

5 6
7 8
```