Multidimensional Arrays in Java

Difficulty Level : Easy Last Updated : 06 May, 2020

Array-Basics 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[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;
```

Example:

```
class GFG {
    public static void main(String[] args)
    {

        int[][] arr = new int[10][20];
        arr[0][0] = 1;

        System.out.println("arr[0][0] = " + arr[0][0]);
    }
}
```

Output:

```
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}};
```

Example:

```java
class GFG {
    public static void main(String[] args)
    {

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

        for (int i = 0; i < 2; i++)
            for (int j = 0; j < 2; j++)
                System.out.println("arr[" + i + "][" + j + "] = "
                                   + arr[i][j]);
    }
}
```

Output:

```
arr[0][0] = 1
arr[0][1] = 2
arr[1][0] = 3
arr[1][1] = 4
```

Accessing Elements of Two-Dimensional Arrays

Elements in two-dimensional arrays are commonly referred 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;
```

The above example represents the element present in first row and first column.

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.

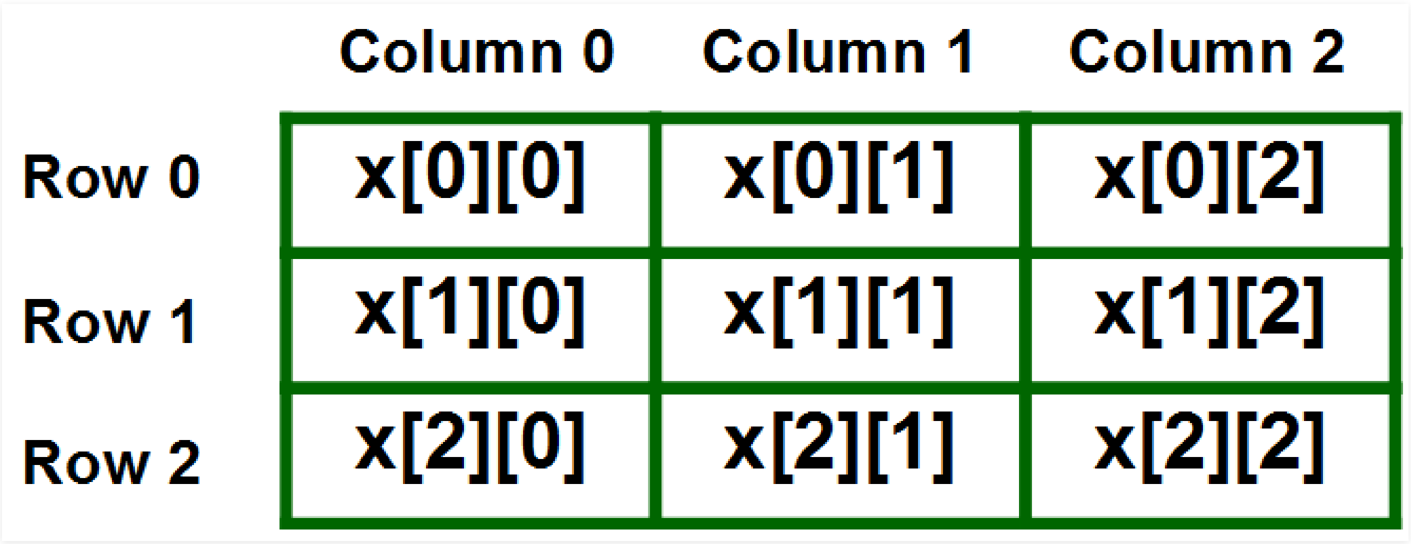
Example:

```
class GFG {
    public static void main(String[] args)
    {

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

        System.out.println("arr[0][0] = " + arr[0][0]);
    }
}
```

Output:

```
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:



	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

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:

```
class GFG {
    public static void main(String[] args)
    {

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

        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print(arr[i][j] + " ");
            }

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

Output:

```
1 2
3 4
```

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;
```

Example:

```
class GFG {
    public static void main(String[] args)
    {

        int[][][] arr = new int[10][20][30];
        arr[0][0][0] = 1;

        System.out.println("arr[0][0][0] = " + arr[0][0][0]);
    }
}
```

Output:

```
arr[0][0][0] = 1
```

Direct Method of Declaration:

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}} };
```

Example:

```java
class GFG {
    public static void main(String[] args)
    {

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

        for (int i = 0; i < 2; i++)
            for (int j = 0; j < 2; j++)
                for (int z = 0; z < 2; z++)
                    System.out.println("arr[" + i
                                       + "]["
                                       + j + "]["
                                       + z + "] = "
                                       + arr[i][j][z]);
    }
}
```

Output:

```
arr[0][0][0] = 1
arr[0][0][1] = 2
arr[0][1][0] = 3
arr[0][1][1] = 4
arr[1][0][0] = 5
arr[1][0][1] = 6
arr[1][1][0] = 7
arr[1][1][1] = 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;
```

The above example represents the element present in the first row and first column of the first array in the declared 3D array.

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.

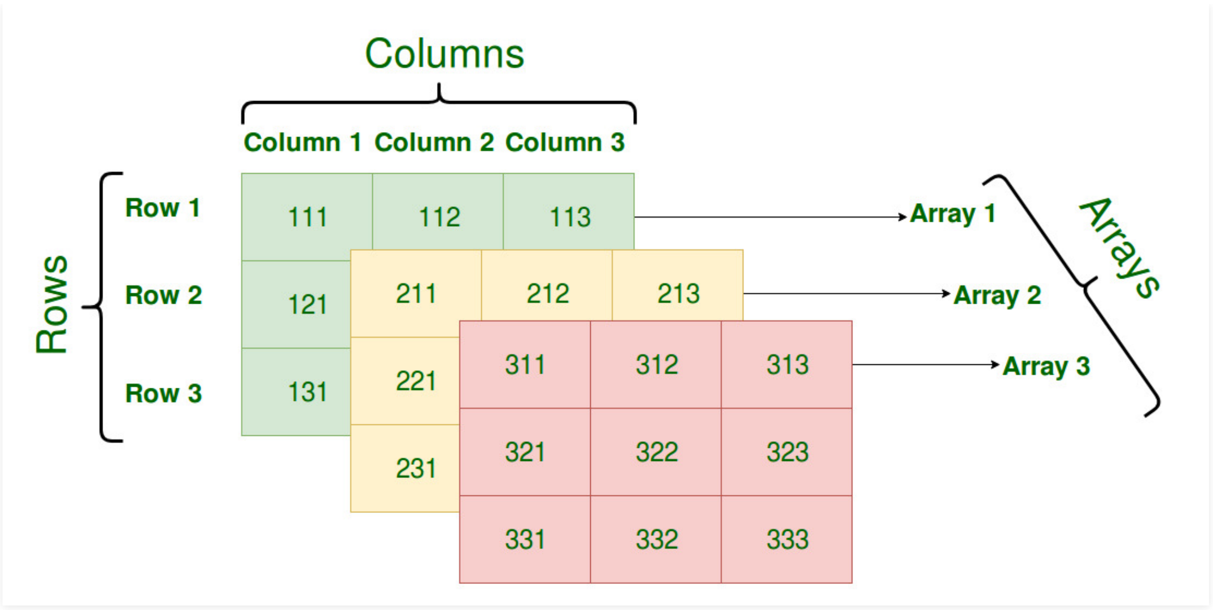
Example:

```
class GFG {
    public static void main(String[] args)
    {

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

        System.out.println("arr[0][0][0] = " + arr[0][0][0]);
    }
}
```

Output:

```
arr[0][0][0] = 1
```

Representation of 3D array in Tabular Format: A three – dimensional array can be seen as a tables of arrays 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 three – dimensional array with 3 array containing 3 rows and 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:

```
class GFG {
    public static void main(String[] args)
    {

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

        for (int i = 0; i < 2; i++) {

            for (int j = 0; j < 2; j++) {

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

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

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

Output:

```
1 2
3 4


5 6
7 8
```

Inserting a Multi-dimensional Array during Runtime:

This topic is forced n taking user-defined input into a multidimensional array during runtime. It is focused on the user first giving all the input to the program during runtime and after all entered input, the program will give output with respect to each input accordingly. It is useful when the user wishes to make input for multiple Test-Cases with multiple different values first and after all those things done, program will start providing output.

As an example, let's find the total number of even and odd numbers in an input array. Here, we will use the concept of a 2-dimensional array. Here are a few points that explain the use of the various elements in the upcoming code:

- Row integer number is considered as the number of Test-Cases and Column values are considered as values in each Test-Case.
- One for() loop is used for updating Test-Case number and another for() loop is used for taking respective array values.
- As all input entry is done, again two for() loops are used in the same manner to execute the program according to the condition specified.
- The first line of input is the total number of TestCases.
- The second line shows the total number of first array values.
- The third line gives array values and so on.

Implementation:

```java
import java.util.Scanner;

public class GFGTestCase {
    public static void main(
        String[] args)
    {
        // Scanner class to take
        // values from console
        Scanner scanner = new Scanner(System.in);
        // totalTestCases = total
        // number of TestCases
        // eachTestCaseValues =
        // values in each TestCase as
        // an Array values
        int totalTestCases, eachTestCaseValues;

        // takes total number of
        // TestCases as integer number
        totalTestCases = scanner.nextInt();

        // An array is formed as row
        // values for total testCases
        int[][] arrayMain = new int[totalTestCases][];

        // for loop to take input of
        // values in each TestCase
        for (int i = 0; i < arrayMain.length; i++) {
            eachTestCaseValues = scanner.nextInt();
            arrayMain[i] = new int[eachTestCaseValues];
            for (int j = 0; j < arrayMain[i].length; j++) {
                arrayMain[i][j] = scanner.nextInt();
            }
        } // All input entry is done.

        // Start executing output
        // according to condition provided
        for (int i = 0; i < arrayMain.length; i++) {

            // Initialize total number of
            // even & odd numbers to zero
            int nEvenNumbers = 0, nOddNumbers = 0;

            // prints TestCase number with
            // total number of its arguments
            System.out.println(
                "TestCase " + i + " with "
                + arrayMain[i].length + " values:");
            for (int j = 0; j < arrayMain[i].length; j++) {
                System.out.print(arrayMain[i][j] + " ");

                // even & odd counter updated as
                // eligible number is found
                if (arrayMain[i][j] % 2 == 0) {
                    nEvenNumbers++;
                }
                else {
                    nOddNumbers++;
                }
            }
            System.out.println();

            // Prints total numbers of
            // even & odd
            System.out.println(
                "Total Even numbers: " + nEvenNumbers
```

```
                    + ", Total Odd numbers: " + nOddNumbers);
            }
        }
    }
    // This code is contributed by Udayan Kamble.
```

```
Input:
2
2
1 2
3
1 2 3

Output:
TestCase 0 with 2 values:
1 2
Total Even numbers: 1, Total Odd numbers: 1
TestCase 1 with 3 values:
1 2 3
Total Even numbers: 1, Total Odd numbers: 2

Input:
3
8
1 2 3 4 5 11 55 66
5
100 101 55 35 108
6
3 80 11 2 1 5

Output:
TestCase 0 with 8 values:
1 2 3 4 5 11 55 66
Total Even numbers: 3, Total Odd numbers: 5
TestCase 1 with 5 values:
100 101 55 35 108
Total Even numbers: 2, Total Odd numbers: 3
```

```
TestCase 2 with 6 values:
3 80 11 2 1 5
Total Even numbers: 2, Total Odd numbers: 4
```