**Two-Dimensional Array**

The simplest of the multi-dimensional array is a two-dimensional array. A simple definition of 2D arrays is: A 2D array is an array of one-dimensional arrays.

In Java, a two-dimensional array is stored in the form of rows and columns and is represented in the form of a matrix.

**The general declaration of a two-dimensional array is,**

data\_type [] [] array\_name;

**Here,**

data\_type = data type of elements that will be stored in an array.  
array\_name = name of the two-dimensional array.

**You can create a 2D array using new as follows:**

data\_type [] [] array\_name = new data\_type[row\_size][column\_size];

**Here,**

row\_size = number of rows an array will contain.  
column\_size = number of columns array will contain.

So if you have an array of 3×3, this means it will have 3 rows and 3 columns.

**The layout of this array will be as shown below.**

| **Rows/ Columns** | **Column1** | **Column2** | **Column3** |
| --- | --- | --- | --- |
| **Row1** | [0,0] | [0,1] | [0,2] |
| **Row2** | [1,0] | [1,1] | [1,2] |
| **Row3** | [2,0] | [2,1] | [2,2] |

As shown above, each intersection of row and column stores an element of the 2D array. So if you want to access the first element in the 2d array, then it is given by [0, 0].

**Note** that as the array size is 3×3, you can have 9 elements in this array.

**An integer array named ‘myarray’ of 3 rows and 2 columns can be declared as below.**

int [][] myarray = new int[3][2];

Once the array is declared and created, it is time to initialize it with values.

**Initialize 2d Array**

There are various ways of initializing the 2d array with values. The first method is the traditional method of assigning values to each element.

**The general syntax for initialization is:**

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

**Example:**

|  |
| --- |
| int[][] myarray = new int[2][2];      myarray[0][0] = 1;      myarray[0][1] =  myarray[1][0] =  0;      myarray[1][1] = 1; |

The above statements initialize all the elements of the given 2d array.

**Let’s put it in a program and check the output.**

|  |
| --- |
| public class Main  {      public static void main(String[] args) {          int[][] myarray = new int[2][2];          myarray[0][0] = 1;          myarray[0][1] =  myarray[1][0] =  0;          myarray[1][1] = 1;          System.out.println("Array elements are:");          System.out.println(myarray[0][0] + " " +myarray[0][1]);          System.out.println(myarray[1][0] + " " +myarray[1][1]);      }  } |

**Output:**

This method may be useful when the dimensions involved are smaller. As the array dimension grows, it is difficult to use this method of individually initializing the elements.

The next method of initializing the 2d array in Java is by initializing the array at the time of declaration only.

**The general syntax for this initialization method is as given below:**

data\_type[][] array\_name = {{val\_r1c1,val\_r1c2,...val\_r1cn},

{val\_r2c1, val\_r2c2,...val\_r2cn}, …

{val\_rnc1, val\_rnc2,…val\_rncn}};

**For Example,** if you have a 2×3 array of type int, then **you can initialize it with the declaration as:**

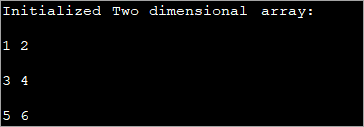
int [][] intArray = {{1, 2, 3},

{4, 5, 6}};

**The following example shows the 2d array declaration with initialization.**

|  |
| --- |
| public class Main  {      public static void main(String[] args) {           //2-d array initialised with values          int[][] intArray = { { 1, 2 }, { 3, 4 },{5,6}};          //print the array          System.out.println("Initialized Two dimensional array:");          for (int i = 0; i < 3; i++) {              for (int j = 0; j < 2; j++) {              System.out.print(intArray [i][j] + " ");              }                System.out.println();          }      }  } |

**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/2d-array-declaration-with-initialization.png)

In the above program, the array is initialized at the time of the declaration itself and then the values are displayed.

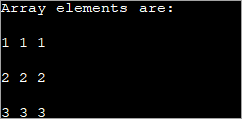
**You can also initialize or assign the values to 2d array using a loop as shown below.**

|  |
| --- |
| int[][] intArray = new int[3][3];          for (int i = 0; i < 3; i++) {              for (int j = 0; j < 3; j++) {                  intArray[i][j] = i+1;              }          } |

**The following program implements the above code.**

|  |
| --- |
| public class Main  {      public static void main(String[] args) {          //declare an array of int          int[][] intArray = new int[3][3];          System.out.println("Array elements are:");          for (int i = 0; i < 3; i++) {              for (int j = 0; j < 3; j++) {                  intArray[i][j] = i+1;           //assign values to each array element                  System.out.print(intArray[i][j] + " "); //print each element              }              System.out.println();          }      }  } |

**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Initialize-or-assign-the-values-to-2d-array-using-a-loop.png)

Each element in the above 2d array is assigned a value ‘i+1’. This makes each element in a row of the array to contain the same value.

**Access And Print 2d Array**

You already know that when initializing the 2d array, you can initialize the individual elements of the array to a value. This is done by using the row index and column index of the array to access a particular element.

Similar to initialization, you can also access the value of the individual element and print it to the user.

**The general syntax for accessing the array element is:**

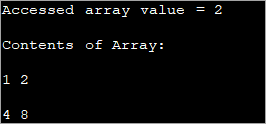
data\_typeval = array\_name[row\_index][column\_index];

Where array\_name is the array whose element is accessed and data\_type is the same as the data type of array.

**The following program shows how an individual element is accessed and printed.**

|  |
| --- |
| public class Main  {      public static void main(String[] args) {          //two dimensional array definition          int[][] intArray = {{1,2},{4,8}};          //Access individual element of array          intval = intArray[0][1];          //print the element          System.out.println("Accessed array value = " + val);          System.out.println("Contents of Array:" );          //print individual elements of array          System.out.println(intArray[0][0] + " " + intArray[0][1]);          System.out.println(intArray[1][0] + " " + intArray[1][1]);        }  } |

**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Individual-element-is-accessed-and-printed.png)

This way you can easily access and print individual array elements using row and column indices enclosed in the square ([]) brackets.

You can print the entire array at once in a tabular format as shown above (also called matrix form) using for loop. As this is a two-dimensional array, you need to have two loops for this. One loop to iterate through rows i.e. the outer loop and the inner loop to traverse the columns.

At any given instant (current iteration), the particular element in the array is given by,

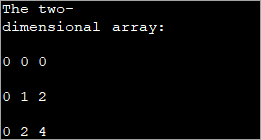
**array\_name[i][j];**

Where ‘i’ is the current row and ‘j’ is the current column.

**The following program shows the printing of a 2d array using a ‘for’ loop.**

|  |
| --- |
| public class Main  {      public static void main(String[] args) {          //two dimensional array definition          int[][] intArray = new int[3][3];          //printing the 2-d array          System.out.println("The two-dimensional array:");          for (int i = 0; i < 3; i++) {              for (int j = 0; j < 3; j++) {                  intArray[i][j] = i\*j;                    //assign value to each array element                  System.out.print(intArray [i][j] + " ");              }              System.out.println("");          }      }  } |

**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Printing-of-a-2d-array-using-a-%E2%80%98for%E2%80%99-loop.png)

In the above program, the 2d array is initialized and then the elements are printed using two for loops. The outer one is used to keep track of rows while the inner for loop is for columns.

**Java 2d Array Length**

A two-dimensional array is defined as the array of a one-dimensional array. Thus, when you need the length of a 2d array it is not as straightforward as in a one-dimensional array.

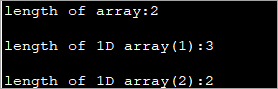
The length property for a two-dimensional array returns the number of rows in the array. Each row is a one-dimensional array. You already know that the two-dimensional array consists of rows and columns. The column size may vary for each row.

Hence you can obtain the size of each row by iterating through the number of rows.

**The following program gives the length of the array (number of rows) as well as the size of each row.**

|  |
| --- |
| public class Main  {      public static void main(String[] args) {              //initialize 2-d array     int[][] myArray = { { 1, 2, 3 }, { 4, 5 } };          System.out.println("length of array:" + myArray.length); //number of rows          for(int i=0;i<myArray.length;i++)          //length of each row          System.out.println("length of 1D array(" + (i+1) +"):" + myArray[i].length); |

**Output:**

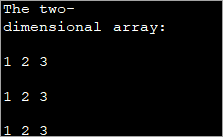
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Length-of-the-arrayand-size-of-each-row.png)

A two-dimensional array defined above has two rows. Each row is a one-dimensional array. The first 1D array has 3 elements (3 columns) while the second row has 2 elements.

**The following Java program shows the usage of length property to print the 2d array.**

|  |
| --- |
| public class Main  {      public static void main(String[] args) {          //two dimensional array definition          int[][] myarray = new int[3][3];          //printing the 2-d array          System.out.println("The two-dimensional array:");          for (int i = 0; i <myarray.length; i++) {              for (int j = 0; j <myarray [i].length; j++) {                  myarray[i][j] = j+1;                    //assign value to each array element                  System.out.print(myarray[i][j] + " ");              }              System.out.println("");          }      }  } |

**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/usage-of-length-property-to-print-the-2d-array.png)

As already mentioned, the outer loop represents the rows and the inner for loop represents the columns.

**Note:** The terminating condition in both loops uses the length property, first to iterate through rows and then through columns.

**Java MultiDimensional Arrays**

We have already seen Two-dimensional arrays. Java supports arrays with more than two dimensions.

**The general syntax of a multi-dimensional array is as follows:**

data\_type [d1][d2]…[dn] array\_name = new

data\_type[d1\_size][d2\_size]…[dn\_size];

**Here,**

d1,d2…dn = dimensions of the multi-dimensional array  
[d1\_size][d2\_size]… [dn\_size] = respective sizes of the dimensions  
data\_type = data type of the array elements  
array\_name = name of multi-dimensional array

As an example of one more multi-dimensional array other than 2d array, let’s discuss the details of three dimensional (3d) arrays.

**Three-Dimensional Arrays In Java**

We already discussed that an array gets more complex as their dimensions increase. Three-dimensional arrays are complex for multi-dimensional arrays. A three dimensional can be defined as an array of two-dimensional arrays.

**The general definition of a Three-dimensional array is given below:**

data\_type [] [] [] array\_name = new data\_type [d1][d2][d3];

**Here,**

d1, d2, d3 = sizes of the dimensions  
data\_type = data type of the elements of the array  
array\_name = name of the 3d array

**Example of 3d array definition is:**

int [] [] [] intArray = new int[2][3][4];

The above definition of 3d array can be interpreted as having 2 tables or arrays, 3 rows and 4 columns that totals up to 2x3x4 = 24 elements.

**This means that in a 3d array, the three dimensions are interpreted as:**

* **The number of Tables/Arrays:** The first dimension indicates how many tables or arrays a 3d array will have.
* **The number of Rows:** The second dimension signifies the total number of rows an array will have.
* **The number of Columns:** The third dimension indicates the total columns in the 3d array.

**Initialize 3d Array**

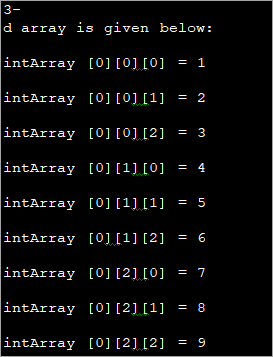
The approaches used to initialize a 3d array are the same as the ones used for initializing Two-dimensional arrays.

You can either initialize the array by assigning values to individual array elements or initialize the array during the declaration.

**The example below shows the initialization of the 3d array while declaration.**

|  |
| --- |
| public class Main  {      public static void main(String[] args) {            //initialize 3-d array          int[][][] intArray = { { { 1, 2, 3}, { 4, 5, 6 } ,  { 7, 8, 9 } } };          System.out.println ("3-d array is given below :");          //print the elements of array          for (int i = 0; i < 1; i++)              for (int j = 0; j < 3; j++)              for (int z = 0; z < 3; z++)                  System.out.println ("intArray [" + i              + "][" + j + "][" + z + "] = " + intArray [i][j][z]);      }  } |

**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Initialization-of-the-3d-array-while-declaration.png)

After initializing the 3d array during declaration, we have accessed the individual elements of the array and printed them.

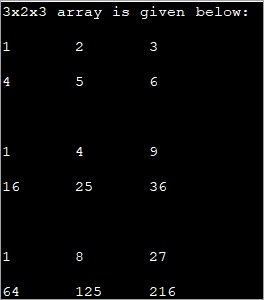
**Acces And Print 3d Array**

Again, printing and accessing array elements in a three-dimensional array is similar to that in two-dimensional arrays.

**The program below uses for loops to access the array elements and print them to the console.**

|  |
| --- |
| public class Main  {      public static void main(String[] args) {          //initialize 3-d array          int[][][] myArray = { { { 1, 2, 3 }, { 4, 5, 6 } },  { { 1, 4, 9 }, { 16, 25, 36 } },                  { { 1, 8, 27 }, { 64, 125, 216 } } };          System.out.println("3x2x3 array is given below:");          //print the 3-d array          for (int i = 0; i < 3; i++) {          for (int j = 0; j < 2; j++) {          for (int k = 0; k < 3; k++) {          System.out.print(myArray[i][j][k] + "\t");          }          System.out.println();                      }          System.out.println();          }      }  } |

**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/For-loops-to-access-the-array-elements.png)

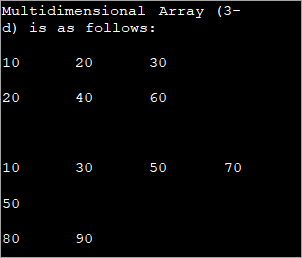
The above program displays a tabular representation of a three-dimensional array. As shown, it is a 3x2x3 array which means that it has 3 tables, 2 rows and 3 columns and thus 18 elements.

It is already mentioned that the column size can vary in a multi-dimensional array. The example below demonstrates a three-dimensional array with varied column sizes.

**This program also uses enhanced for loop to traverse through the array and display its elements.**

|  |
| --- |
| public class Main  {      public static void main(String[] args) {          //initialize 3-d array          int[][][] intArray = {                {{10, 20, 30},{20, 40, 60}},                { {10, 30,50,70},{50},{80, 90}}                };      System.out.println("Multidimensional Array (3-d) is as follows:");      // use for..each loop to iterate through elements of 3d array      for (int[][] array\_2D: intArray) {          for (int[] array\_1D: array\_2D) {              for(intelem: array\_1D) {              System.out.print(elem + "\t");               }             System.out.println();          }          System.out.println();      }      }  } |

**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/For-loop-to-traverse-through-the-array.png)

The input array used is a Three-dimensional array with a varied length of columns. The enhanced for each loop used for each dimension displays the contents of the array in a tabular format.

**Frequently Asked Questions**

**Q #1) What do you mean by Two dimensional array?**

**Answer:** A Two-dimensional array is called an array of arrays and is usually organized in the form of matrices consisting of rows and columns. A Two-dimensional array finds its use mostly in relational databases or similar data structures.

**Q #2) What is a Single-dimensional array in Java?**

**Answer:** One-dimensional array in Java is an array with only one index. This is the simplest form of arrays in Java.

**Q #3) What is the difference between a one-dimensional array and a two-dimensional array?**

**Answer:** One-dimensional array stores a single sequence of elements and has only one index. A two-dimensional array stores an array of arrays of elements and uses two indices to access its elements.

**Q #4) What does it mean to be two dimensional?**

**Answer:** Two-dimensional means having only two dimensions. In a geometric world, objects that have only height and width are two-dimensional or 2D objects. These objects do not have thickness or depth.

Triangle, rectangles, etc. are 2D objects. In software terms, two dimensional still means having two dimensions and we usually define data structures like arrays which can have 1, 2 or more dimensions.

**Q #5) Which one comes first in an array – Rows or Columns?**

**Answer:** Two-dimensional arrays are represented as matrices and matrices are usually written in terms of rows x columns. **For Example,** a matrix of size 2×3 will have 2 rows and 3 columns. Hence for the 2D array as well, rows come first and columns next.