Tutorial Link https://course.testpad.chitkara.edu.in/tutorials/Java: Multidimensional Array/5aabd0e226e8ba7398f58cc1

TUTORIAL

Java: Multidimensional Array

Multi-Dimensional Array

Multiple-subscripted arrays, like two subscripts are often used to represent tables of values consisting of information arranged in rows and columns. To identify a particular table element, we must specify the two subscripts — by convention, the first identifies the element's row and the second identifies the element's column. Arrays that require two subscripts to identify a particular element are called double-subscripted arrays or two-dimensional arrays.

Note that multiple-subscripted arrays can have more than two subscripts. Java does not support multiple-subscripted arrays directly, but does allow the programmer to specify single-subscripted arrays whose elements are also single-subscripted arrays, thus achieving the same effect. In general, an array with m rows and n columns is called an m-by-n array.

Every element in array a is identified by an element name of the form a[row][column]; a is the name of the array and row and column are the subscripts that uniquely identify the row and column of each element in a. Notice that the names of the elements in the first row all have a first subscript of 0; the names of the elements in the fourth column all have a second subscript of 3.

Multiple-subscripted arrays can be initialized with initializer lists in declarations like a single-subscripted array. A double-subscripted array b[2][2] could be declared and initialized with

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

The values are grouped by row in braces. So, 1 and 2 initialize b[0][0] and b[0][1], and 3 and 4 initialize b[1][0] and b[1][1]. The compiler determines the number of rows by counting the number of initializer sublists (represented by sets of braces) in the initializer list. The compiler determines the number of columns in each row by counting the number of initializer values in the initializer sublist for that row.

Multiple-subscripted arrays are maintained as arrays of arrays. The declaration

```
int b[][] = { { 1, 2 }, { 3, 4, 5 } };
```

creates integer array b with row 0 containing two elements (1 and 2) and row 1 containing three elements (3, 4 and 5). A multiple-subscripted array with the same number of columns in every row can be allocated dynamically. For example, a 3-by-4 array is allocated as follows:

```
int b[][];
b = new int[ 3 ][ 4 ];
```

In this case, we the literal values 3 and 4 to specify the number of rows and number of columns, respectively. Note that programs also can use variables to specify array dimensions. As with single-subscripted arrays, the elements of a double-subscripted array are initialized when new creates the array object.

A multiple-subscripted array in which each row has a different number of columns can be allocated dynamically as follows:

```
int b[][];
b = new int[ 2 ][ ]; // allocate rows
b[ 0 ] = new int[ 5 ]; // allocate columns for row 0
b[ 1 ] = new int[ 3 ]; // allocate columns for row 1
```

The preceding code creates a two-dimensional array with two rows. Row 0 has five columns and row 1 has three columns. In a m x n two-

dimensional array to total the element values use following code:

```java
import java.util.Scanner;
// Other imports go here

class Main
{
  public static void main(String[] args)
  {
    int[][] a = {{1,2,3}, {1,2,3}, {1,1,1}};
    int total = 0, m=3, n=3;
    for ( int row = 0; row < m; row++ )
      for ( int column = 0; column < n; column++ )
        total += a[ row ][ column ];

    System.out.println("Sum of the elements is : " + total);
  }
}
```

The for structure totals the elements of the array one row at a time. The outer for structure begins by setting the row subscript to 0 so that the elements of the first row may be totaled by the inner for structure. The outer for structure then increments row to 1 so that the second row can be totaled. Then, the outer for structure increments row to 2 so that the third row can be totaled. The result can be displayed when the nested for structure terminates.