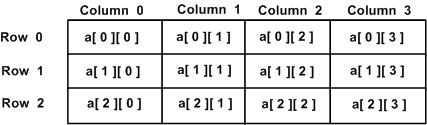
Multi-Dimensional Array [Java]

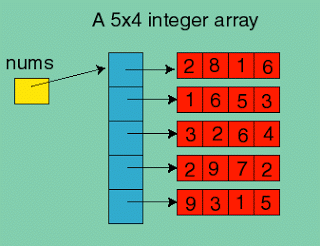
We have arrays of two, three or more dimensions. We will look at multi dimensional arrays, with special focus on two dimensional arrays which are quite frequently used.

**2-D Array**

Visualizing two-dimensional arrays



**Multi-dimensional arrays are built from multiple 1-D arrays**



**Array declaration:**

The following statement creates a 2-D array of integers, which contains 3 arrays containing 4 integers each.

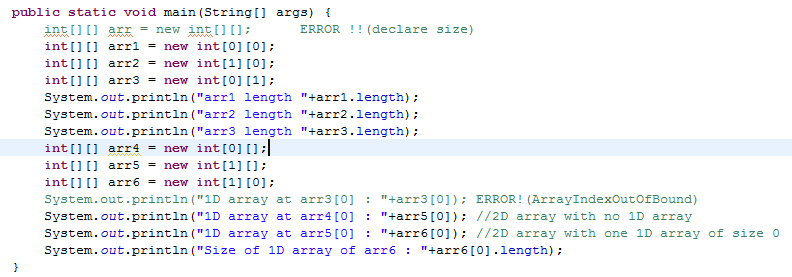
int[][] a=new int[3][4];

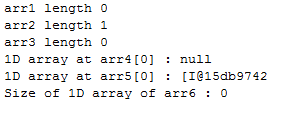
**Note:**

Always declare size of main array

Example :

int[][][] array = new int[3][][];





**Accessing Array elements:**

Elements of this array are accessed by specifying the index numbers, here two of them. The first representing the array number and the second representing the index element in that particular array.

a[0][2] = 34;

**Array initialization:**

user input array

Scanner scanner = new Scanner(System.in);

for ( int i=0; i<a.length; i++) {

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

a[i][j] = scanner.nextInt();

}

}

A 2-D array can also be initialized in the following way.

int [ ] [ ] d = {

{ 1, 5, 74, 2 },

{ 4, 68, 45, 65 },

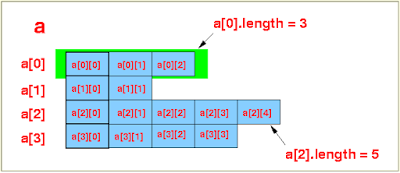
{ 5, 0, 34, 54 }

}:

The above array a[][] contains three arrays { 1,5,74,2}, {4,68,45,65} and {5,0,34,54}. This particular 2D array contains equal number of elements in each of its sub arrays.

**Creating variable size array:**

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



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

b [ 0 ] = new int [ 3 ];

b [ 1 ] = new int [ 1 ];

b [ 2 ] = new int [ 2 ];

Observe that in the first statement, we haven't specified any integer in the second pair of square brackets following [3].

b[][] is reference type and so are b[0], b[1] and b[2].  
b[0][0], b[0][1] .... are of primitive type int.

Create 2D arrays which have different number of elements in its sub arrays.

int [ ] [ ] c = {

{ 4, 56, 7},

{ 1 },

{ 45, 78 }

};

The above 2-D array contains three arrays. The first of these has 3 elements, the second has 1 element and the last of these has 2 elements. Printing 2-D array elements:

for ( int i = 0; i < a.length; i++) {

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

System.out.print ( a[ i ][ j ] );

}

System.out.println( );

}

The code works even when the rows of the array are of different lengths.

**Searching in 2-D Array:**

for ( int i = 0; i < a.length; i++ ) {

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

if ( a [ i ] [ j ] == target) {

System.out.println(" Element found at [" + i + "] [ "+j+" ]" );

return;

}

}

}