Multidimensional Arrays Chapter

Contents

[Introduction 1](#_Toc46589851)

[How Does This Work? 1](#_Toc46589852)

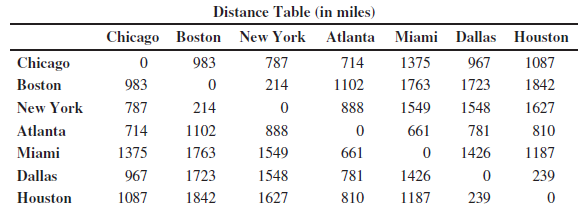
[Example in use: 2](#_Toc46589853)

[Expanding it to Multidimensional Arrays 2](#_Toc46589854)

# Introduction

Java only has 1D data structure. But, that 1D data structure can hold a 1D array and that can hold a 1D array, and so on.

When we think of a table, like below, then we can hold those values in a 2D “structure” where it really is a 1D data structure holding a 1D data structure.



If we count the cities going vertically that provides the number of rows.

If we count the cities going horizontally that provides the number of columns.

In Java, we would then set this up as:

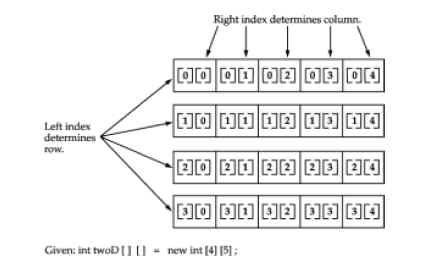
int [][] distanceTable = new int [7][7];

This provides a data structure of how many rows by how many columns, as we normally think about a table structure.

# How Does This Work?

Let’s take another example: int [][] matrix = new int [4][5];

Let’s visualize this:



Then we see that the first row is a 1D array of 5 elements, the 2nd row is a 1D array of 5 elements, and the third row is a 1D of elements of 5 elements and the fourth row is a 1D array of 5 elements.

That also means there are 20 elements/values being held in this structure – 4 rows \* 5 columns = 20 elements.

## Example in use:

int [][] table = new int [2][2];

// creating a variable so we can see how the flow of this 2D table works  
int spot = 0;

for(int row = 0; row < table.length; row++){

for(int column = 0; column < table[row].length; column++){

// you have to bound the length since each table can be a different size

table[row][column] = spot; // assigning a value

spot++;

}

}

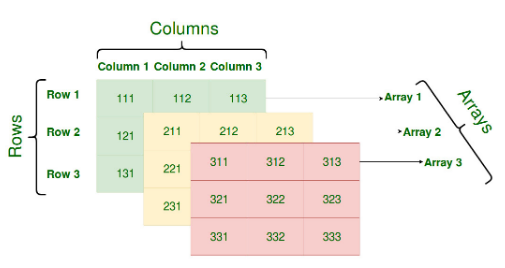
# Expanding it to Multidimensional Arrays

In Java, you can create n-dimensional arrays for any positive integer n.

For each dimension, we just add another [] and another for loop to iterate through that [].

We have to keep in mind that since each 1D array can be a different length, we have to bound it so that we do not go out of bounds.

We can think of a “cube” with rows, columns, and depth, thus three [], as shown below.



Then to declare it:

int [] [] [] threeD = new int [3][3][3];

to iterate through it, we need three for loops:

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();

}

Another way to look at it:

