Java Foundations Notes

Chapter 7 - Arrays

CSC110

Suppose you are writing a program to keep track of your golf scores over 4 rounds. You may think to declare 4 variables like this:

#### int round1 = 83;

#### int round2 = 78;

#### int round3 = 92;

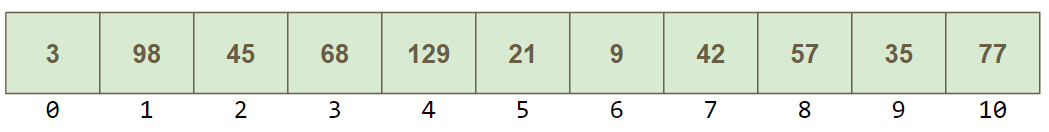
#### int round4 = 85;

However, Java provides a better way to store related values called an array.

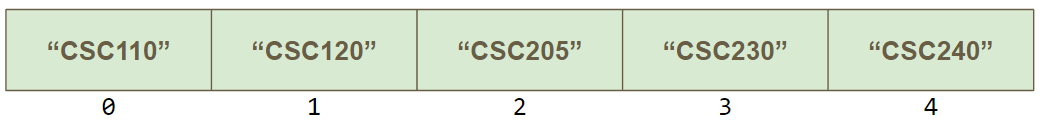
# Arrays

An **array** is an object that holds a list of values. The values are stored in locations called **cells**. Each cell has a numeric index that can be used to refer to a specific location in the array.

Here is an array of integers:



Here is another example array, this time storing Strings:



Each array has a name that represents the entire array. The name is assigned when we declare the array. You must also indicate the *size* of the array in the declaration:

#### int[] dailyCounts = new int[7];

#### double measurements = new double[6];

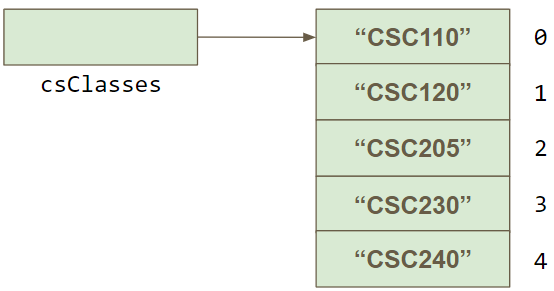
#### String top100movies = new String[100];

We can declare the String array from above that holds CS courses as follows:

#### String[] csClasses = {"CSC110", "CSC120", "CSC205",

#### "CSC230", "CSC240"}

Then we can represent what this array looks like as follows (here we represent the array vertically instead of horizontally, but this is the same array as shown above):



Here, csClasses is an String array object reference variable that refers to the String array that contains 5 strings.

Going back to our golf example, we can declare an array to hold our golf scores as follows:

#### int[] scores = new int[4];

We can even initialize the array when you create it, to save on even more typing:

#### int[] scores = {83, 78, 92, 85};

Each cell has a numeric **index** that we can use to access it. The indexes start at 0, so always remember that the *first* array index is 0, the second element is stored at index 2, and so on. Our scores array has indexes 0, 1, 2, & 3. We can fill our golf score array like this:

#### scores[0] = 83; // instead of round1

#### scores[1] = 78;

#### scores[2] = 92;

#### scores[3] = 85;

If we want to iterate through the entire array, we can use a for loop:

#### for (int round=0;round<scores.length;round++) {

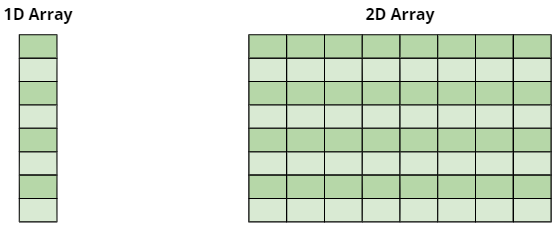
#### System.out.println("Round " + round + " score:" + scores[round]);

**}**

# Two-Dimensional Arrays

We can also have a **two-dimensional array** which you can think of as a table with rows & columns. To be precise, a two-dimensional array is an *array of arrays*.

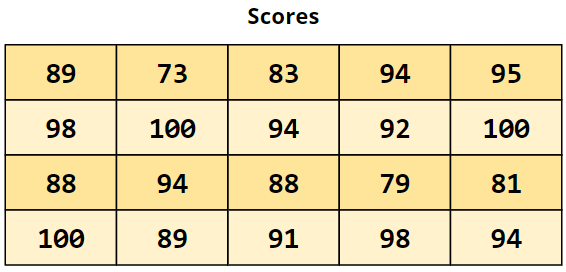
While a 1D array can be thought of as a list, a 2D array can be thought of as a table with rows and columns.



We declare a 2D array by specifying the size of each dimension separately.

#### int[][] scores = new int[4][5];

Suppose that we fill the above array with the following values:

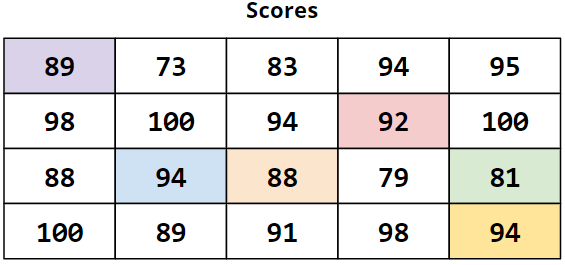


We can reference individual elements using two index values, one for the row, one for the column:

#### value = scores[3][2];

In this case, the variable value would be assigned the value 91, since that is the value stored in row index 3 (the 4th row) and column index 2 (the 3rd row). Always keep in mind that array indexes start with 0!

Here are a few more examples of accessing individual locations in the 2D array that I have color coded so you can see visually where they are

****

**scores[0][0] = 89**

**scores[1][3] = 92**

**scores[2][1] = 94**

**scores[2][2] = 88**

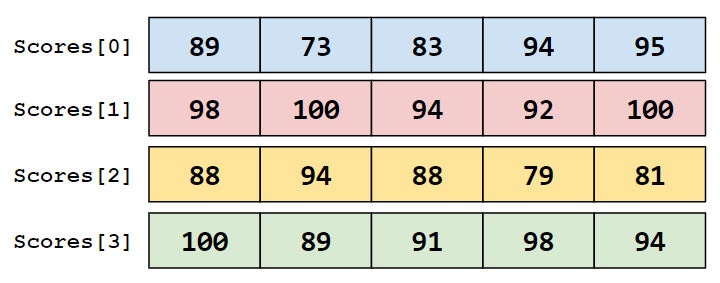
**scores[2][4] = 81**

**scores[3][4] = 94**

In addition to accessing individual elements of the 2D array, we can also access individual *rows*. Recall that a 2D array can be thought of as an array of arrays, where each index refers to a 1D array. Therefor, the array stored in one row can be specified using a single index:

#### value = scores[2];

Here is an example of the 2D scores array being accessed as 4 individual arrays:



### Review of Accessing Elements in an Array

Suppose we have a 2D array declared as follows:

#### int[][] table = new int[5][10];

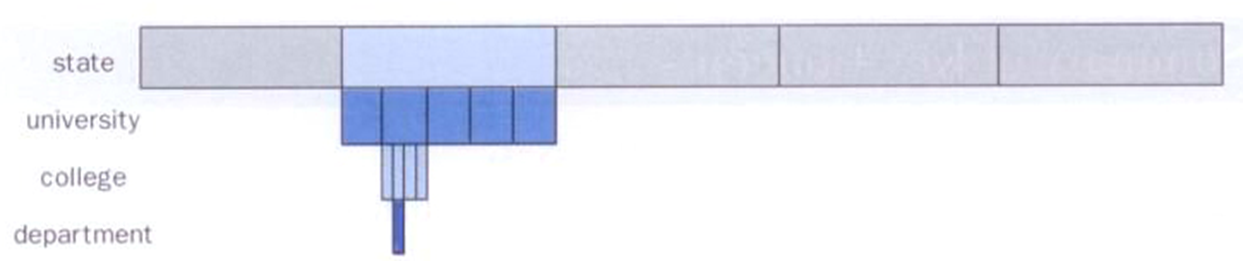
| **Expression** | **Declaration** | **Declaration** |
| --- | --- | --- |
| **table** | **int[][]** | 2D array of integer, or array of integer arrays |
| **table[5]** | **int[]** | Array of integers |
| **table[5][2]** | **int** | Integer |

# Multi-dimensional Arrays

Any array with more than one dimension is a multidimensional array. This includes two-dimensional arrays. Each dimension subdivides the previous one into the specified number of elements. Each dimension has its own length constant.

Because each dimension is an array of array references, the arrays within one dimension can be of different lengths. These are sometimes called **ragged arrays**.

For example, you could have a 4 dimensional array that you could visualize as follows:



In general, 2-dimensional arrays are the most common multidimensional array, but there are situations where a multidimensional array can be helpful.