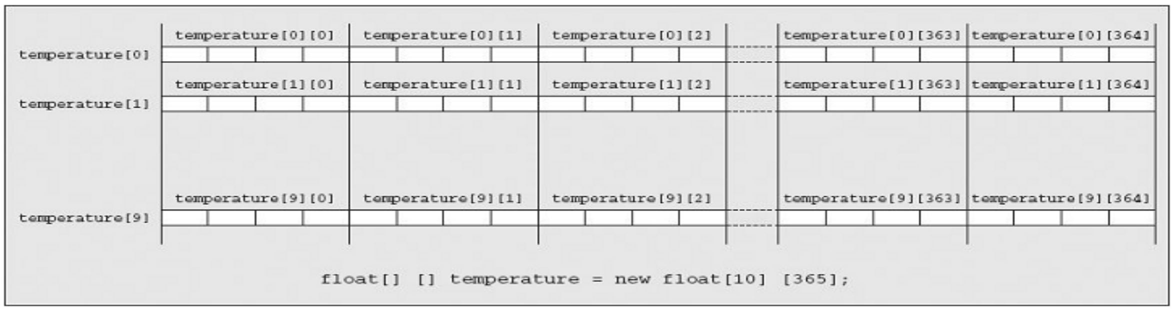
Arrays of Arrays

You have worked only with one-dimensional arrays up to now, that is, arrays that use a single index. Why would you ever need the complications of using more indexes to access the elements of an array?

Consider a specific example. Suppose that you have a fanatical interest in the weather, and you are intent on recording the temperature each day at 10 separate geographical locations throughout the year. After you have sorted out the logistics of actually collecting this information, you can use an array of 10 elements corresponding to the number of locations, where each of these elements is an array of 365 elements to store the temperature values. You declare this array with the statement:

float[][] temperature = new float[10][365];

This is called a two-dimensional array because it has two dimensions — one with index values running from 0 to 9, and the other with index values from 0 to 364. The first index relates to a geographical location, and the second index corresponds to the day of the year. That's much handier than a one-dimensional array with 3650 elements, isn't it?



There are 10 one-dimensional that make up the two-dimensional array, and they each have 365 elements. So to refer to the temperature for day 100 for the sixth location, you use temperature[5][99].

For a fixed value for the second index in a two-dimensional array, varying the first index value is often referred to as accessing a column of the array. Similarly, fixing the first index value and varying the second, you access a row of the array. The reason for this terminology should be apparent

Arrays of Arrays of Varying Length

When you create an array of arrays, the arrays in the array do not need to be all the same length. You could declare an array variable, samples, with the statement:

```
float[][] samples;
samples = new float[6][];
```

The samples variable now references an array with six elements, each of which can hold a reference to a one-dimensional array. You can define these arrays individually if you want:

```
samples[2] = new float[6];           // The 3rd array has 6 elements

samples[5] = new float[101];         // The 6th array has 101 elements
```

This defines two of the six possible one-dimensional arrays that can be referenced through elements of the samples array.

Multidimensional Arrays : long[][][] beans = new long[5][10][30];

You are not limited to two-dimensional arrays either. If you are an international java bean grower with multiple farms across several countries, you could arrange to store the results of your bean counting in the array declared and defined in the following statement:

Video / Print Resources

https://thenewboston.com/index.php - Java - Beginners – Videos 33 and 34

https://thenewboston.com/videos.php?cat=31&video=17998

https://thenewboston.com/videos.php?cat=31&video=17999

http://math.hws.edu/javanotes/c7/s5.html

2D Arrays

1. What are 2D Arrays, why do we need 2D arrays?
2. How to think abstractly about 2D arrays
3. Declare 2D Arrays: Create 2D Arrays
4. How does memory work – Array of Arrays : (do this by comparing this with 1D arrays)
5. How do you initialize?
   a. Initialization List
   b. Nested Loops
      i. How to access number of rows
      ii. How to access number of columns.
6. Maniplulation