SIENAcollege
Computer Science

Computer Science 210
Data Structures
Siena College
Fall 2017

Topic Notes: Two-Dimensional Arrays

Two-Dimensional Arrays

We can create arrays to hold objects of any type, either basic data types like `int` and `double`, or instances of objects.

Nothing stops us from defining arrays of arrays. To declare an array, each of whose elements is an array of `int`:

```
int[][] twoDArray;
```

While it is normally written without parentheses, we can think of the above declaration as defining `twoDArray` as having type `(int [])[]`. Thus each element of `twoDArray` is an array of `int`s.

Despite the fact that Java will treat this as an array of arrays, we usually think about this as a two-dimensional array, with the elements arranged in a two-dimensional table so that `twoDArray[i][j]` can be seen as the element in the ith row and jth column. For example here is the layout for a two-dimensional array `a` with 6 rows (numbered 0 to 5) and 4 columns:

	0	1	2	3
0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
2	a[2][0]	a[2][1]	a[2][2]	a[2][3]
3	a[3][0]	a[3][1]	a[3][2]	a[3][3]
4	a[4][0]	a[4][1]	a[4][2]	a[4][3]
5	a[5][0]	a[5][1]	a[5][2]	a[5][3]

Viewed in this way, our two-dimensional array is a grid, much like a map or a spreadsheet. This is a natural way to store things like tables of data or matrices.

We access elements of two-dimensional arrays in a manner similar to that used for one dimensional arrays, except that we must provide both the row and column to access an element, giving the row number first.

We create a two-dimensional array by providing the number of rows and columns. Thus we can create the two-dimensional array above by writing:

```
int[][] a = new int[6][4];
```

(Though as good programmers, you would define constants for the number of rows and the number of columns.)

A nested `for` loop is the most common way to access or update the elements of a two-dimensional array. One loop walks through the rows and the other walks through the columns. For example, if we wanted to assign a unique number to each cell of our two-dimensional array, we could do the following:

```
for (int row = 0; row < 6; row++) {
    for (int col = 0; col < 4; col++) {
        a[row][col] = 4*row + col + 1;
    }
}
```

This assigns the numbers 1 through 24 to the elements of array `a`. The array is filled by assigning values to the elements in the first row, then the second row, etc. and results in:

```
 1  2  3  4
 5  6  7  8
 9 10 11 12
13 14 15 16
17 18 19 20
21 22 23 24
```

And if we wanted to print the above, we can write a loop:

```
for (int row = 0; row < 6; row++) {
    for (int col = 0; col < 4; col++) {
        System.out.print(a[row][col] + " ");
    }
    System.out.println();
}
```

You could modify the above to be slightly more interesting by computing a multiplication table.

We could just as well process all the elements of column 0 first, then all of column 1, etc., by swapping the order of our loops:

```
for (int col = 0; col < 4; col++)
    for (int row = 0; row < 6; row++)
        ...
```

For the most part, it doesn't matter which order you choose, though for large arrays it is generally a good idea to traverse the array in the same order that your programming language will store the values in memory. For Java (and C, C++), the data is stored by rows, known as *row major* order. However, an two-dimensional array in FORTRAN is stored in *column major* order. You will almost certainly see this again if you go on and take courses like Computer Organization or Operating Systems.

Two-Dimensional Matrices

A very common use of two-dimensional arrays is the representation of matrices. We will look at an example of a class that represents two-dimensional square matrices and provides some basic operations on them.

See Example: Matrix2D

The class is capable of holding a square matrix of `double` values of any positive dimension.

Comments within the example explain much of what is happening. Note in particular the use of the two-dimensional array as an instance variable which stores the matrix entries, the use of exceptions to handle error conditions, and the `main` method that tests out the methods of the class. We will be seeing much more about exception handling soon.