20. Multi-Dimensional Arrays

Introduction

- Quick Review:
  - Arrays
  - Iterating over Arrays
  - Using Arrays
  - The Cosmology of Collections
- In this lesson:
  - Multi-dimensional arrays
  - Iterating over them.

A Multiplication Table

- Suppose we want to implement a multiplication table (for 0..5)
- We want to represent 6 rows of integers, where each row has 6 items. If each row is an array of integers, then we need an array of arrays of integers. We can do this in Java:

```
int[][] products = new int[6][6];
```

  - We are creating an array of arrays of integers.
  - Pictures.
  - What is the type of "products"?

Accessing our array

- Terms: *rows* and *columns*

```
int[][] products = new int[6][6];

int[] firstRow = products[0];       // this is the first row

firstRow[0] = 0;              // the first element of the first row

// we can combine our "subscripts"

products[4][3] = 12;
products[2][2] = 4;
products[0][5] = 0;
products[5][0] = 0;
```

  - For each of the above, show which element of the array we're talking about.

Finding the Element

- Conceive of a way of thinking about the array and stick with it.
  - The first subscript is the *row index*
  - The second subscript is the *column index*
- This interpretation jibes with our pictures.
- (Note: Our examples have the same number of rows and columns. This doesn't have to be the case):

```
// a poor way to collect weather data:

double[][] samples = new double[12][31];
```

Processing the Array (1)

- Let's write a loop to fill in the array for us:

```
int[][] products = new int[6][6];

// the outer loop iterates over the rows
for (int row = 0; row < 6; row = row + 1) {
  // the inner loop iterates over the columns
  for (int col = 0; col < 6; col = col + 1) {
    products[row][col] = row * col;
  }
}
```

  - Notice the indenting, and the names used.

2D Array Processing Pattern

- Many 2D array traversals look like this:

```
for (int row=0; row < array.length; row = row + 1) {

  for (int col=0; col < array[row].length; col = col + 1) {
  ... array[row][col] ...
  }

}
```

  - The inner loop iterates over a given row
  - The outer loop iterates over each of the rows.

Processing the Array (2)

- Let's make the loop resilient to different array dimensions:

```
int[][] products = new int[6][6];

// the outer loop iterates over the rows
for (int row = 0; row < products.length; row = row + 1) {

  // the inner loop iterates over the columns
  for (int col = 0; col < products[row].length; col = col + 1) {
    products[row][col] = row * col;
  }

}
```

Exercise: Printing the Table

- Write a loop that prints the multiplication table

Exercise: Traversing on a diagonal

- Why would you ever want to do this? Homework 7:

```
e e h t
t r e e
g r r r
f a h a
```