What we will cover:
Processing 2d Arrays (cont.)

Processing 2d Arrays (cont.)
Recall the general structure for processing a 2d array and where each section in the code corresponds to pieces of the array.

```java
int[][] matrix = new int[3][3];
for (int row = 0; row < matrix.length; row++) {
    // actions specific to row, before column processing
    for (int column = 0; column < matrix[row].length; column++) {
        // we can now access matrix[row][column]
    }
    // actions specific to entire row, after column processing
}
```

At this point, anything we put in the inner loop’s body has access to the current element. In addition to initializing, setting, and retrieving specific values in the inner loop, we can get totals for the entire 2d array or just for specific rows.

Total sum of all elements:
```java
int sum = 0;
for (int row = 0; row < matrix.length; row++) {
    for (int column = 0; column < matrix[row].length; column++) {
        sum += matrix[row][column];
    }
}
System.out.println("Sum: "+sum);
```

Sum of only each row:
```java
for (int row = 0; row < matrix.length; row++) {
    int sum = 0;
    for (int column = 0; column < matrix[row].length; column++) {
        sum += matrix[row][column];
    }
    System.out.println("Sum for row "+row+:"+sum);
}
```

Class Exercise 40

We can also slightly manipulate the two for loops to do things specifically to columns. The first bit of code is our standard, “walk through every row and print every column” code. Compare it to the second nested for loop. What does the second bit of code do?

```java
for (int row = 0; row < matrix.length; row++) {
    for (int column = 0; column < matrix[row].length; column++) {
        System.out.print(matrix[row][column] + " ");
    }
    System.out.println();
}
System.out.println();
```

```java
for (int column = 0; column < matrix[0].length; column++) {
    for (int row = 0; row < matrix.length; row++) {
```
```java
System.out.print(matrix[row][column] + " ");
}
System.out.println();
}

Output:
8 1 5
5 2 6
0 0 3
8 5 0
1 2 0
5 6 3

What are some disadvantages to this column-centric code? What kind of arrays can it work on?

The algorithm for calculating the max (or min) of an array typically sets the first array value as the initial max (or min) and then compares that value to every other element in the array. What if we wanted to find the max (or min) row value in a 2d array? We should follow the same algorithm, but adapt it for 2d arrays:

```java
int maxRow = 0;
int maxIndex = 0;

for (int column = 0; column < matrix[0].length; column++) {
    maxRow += matrix[0][column];
}

for (int row = 1; row < matrix.length; row++) {
    int rowTotal = 0;
    for (int column = 0; column < matrix[row].length; column++) {
        rowTotal += matrix[row][column];
    }
    if (rowTotal > maxRow) {
        maxRow = rowTotal;
        maxIndex = row;
    }
}

System.out.println("Row " + maxIndex + " has max value of " + maxRow);
```

Class Exercise 41