Lecture 11
Multidimensional arrays
Two-dimensional Arrays

- Just an array of arrays
- useful for storing data in a table, or pixel information, for example
- syntax is very similar to one-dimensional array
Two-dimensional Arrays

- Syntax:
  ```java
elementType[][] arrayRefVar;
```

- Example:
  ```java
int[][] matrix;
```
Two-dimensional Arrays

• Creating a two-dimensional array:

```java
int[][] matrix = new int[5][5];
```
Two-dimensional Arrays

- Initializing a two-dimensional array:

```java
int[][] matrix = {
    {1, 2, 3, 4, 5},
    {6, 7, 8, 9, 10},
    {11, 12, 13, 14, 15},
    {16, 17, 18, 19, 20},
    {21, 22, 23, 24, 25}
};
```
Two-dimensional Arrays

- Accessing item in a two-dimensional array:
Two-dimensional Arrays

- Accessing item in a two-dimensional array:

  ```
  int num = matrix[2][3];
  ```
Two-dimensional Arrays
Two-dimensional Arrays

```
matrix = new int[5][5];

matrix[2][1] = 7;

int[][] array = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9},
    {10, 11, 12}
};
```
Two-dimensional Arrays

- Not all rows need to be the same length.
- Called a Ragged Array
- You do, however, need to know at least how many rows there will be (length of outermost array)

```java
int[][] triangleArray = {
    {1, 2, 3, 4, 5},
    {2, 3, 4, 5},
    {3, 4, 5},
    {4, 5},
    {5}
};
```
Two-dimensional Arrays

```java
int[][] triangleArray = new int[5][];
triangleArray[0] = new int[5];
triangleArray[1] = new int[4];
triangleArray[2] = new int[3];
triangleArray[3] = new int[2];
triangleArray[4] = new int[1];

triangleArray[0][3] = 50;
triangleArray[4][0] = 45;
```
Two-dimensional Arrays

- Processing two-dimensional arrays:
Two-dimensional Arrays

- Processing two-dimensional arrays: For loops!
Back to Eclipse!

• We’ll start using Eclipse again

• (Please make sure you still have it installed 😊 )

• There are very powerful tools in full-featured IDEs like Eclipse such as debugging and easier-to-use packaging and deployment.

• This will be very important when we get to Processing, or if you’re building Android apps or other large projects
Debugging walkthrough
Two-dimensional Arrays

- Processing two-dimensional arrays: For loops!

```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(" ");
}
```
Let’s find which row has the largest sum from the following table:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>12</td>
<td>6</td>
<td>23</td>
</tr>
<tr>
<td>45</td>
<td>43</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>56</td>
<td>23</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>67</td>
<td>32</td>
<td>34</td>
<td>29</td>
</tr>
<tr>
<td>78</td>
<td>3</td>
<td>45</td>
<td>56</td>
</tr>
</tbody>
</table>
Two-dimensional Arrays - Practice

Let’s find which row has the largest sum from the table

```java
int[][] matrix = {
    {7, 12, 6, 23},
    {45, 43, 3, 5},
    {56, 23, 1, 4},
    {67, 32, 34, 29},
    {78, 3, 45, 56}
};

int largestRowIndex = 0;
int largestSum = 0;

for (int row = 0; row < matrix.length; row++) {
    int currentRowSum = 0;
    for (int column = 0; column < matrix[row].length; column++) {
        currentRowSum += matrix[row][column];
    }
    if (currentRowSum > largestSum) {
        largestSum = currentRowSum;
        largestRowIndex = row;
    }
}
System.out.println("The largest sum was "+ largestSum + " found in row "+ largestRowIndex);
```
Multi-dimensional Arrays

- Yo dawg, I heard you like arrays, so I put an array in your array of arrays...
Multi-dimensional Arrays

- Useful for a text file of data, multiple items of data associated with a timestamp, etc.

```java
double[][][] scores = new double[6][5][2];
```
Multi-dimensional Arrays

Which student \rightarrow \text{scores} [i] \rightarrow \text{Multiple-choice or essay} [k]

Which exam \rightarrow \text{scores} [j] \rightarrow

Multiple-choice or essay [k]
Multi-dimensional Arrays

Student 1 scores

```java
int[][][] scores = {
    {{20,30}, {15, 25}, {10,20}}, // Student 1
    {{20,30}, {13, 22}, {16,25}}, // Student 2
    {{19,29}, {14, 24}, {12,22}}, // Student 3
    {{18,30}, {12, 28}, {14,28}}, // Student 4
    {{17,26}, {18, 27}, {20,30}}}; // Student 5
```
Multi-dimensional Arrays

Midterm 2 scores for all students

```java
int[][][] scores = {
    {{20, 30}, {15, 25}, {10, 20}},   // Student 1
    {{20, 30}, {13, 22}, {16, 25}},   // Student 2
    {{19, 29}, {14, 24}, {12, 22}},   // Student 3
    {{18, 30}, {12, 28}, {14, 28}},   // Student 4
    {{17, 26}, {18, 27}, {20, 30}}    // Student 5
};
```
Multi-dimensional Arrays

Midterm2 coding section scores for all students

```java
int[][][] scores = {
    {{20,30}, {15, 25}, {10,20}},  // Student 1
    {{20,30}, {13, 22}, {16,25}},  // Student 2
    {{19,29}, {14, 24}, {12,22}},  // Student 3
    {{18,30}, {12, 28}, {14,28}},  // Student 4
    {{17,26}, {18, 27}, {20,30}}};  // Student 5
```
Practice: Multi-dimensional Arrays

```java
int[][][][] scores = {
    {{20,30}, {15, 25}, {10,20}}, // Student 0
    {{20,30}, {13, 22}, {16,25}}, // Student 1
    {{19,29}, {14, 24}, {12,22}}, // Student 2
    {{18,30}, {12, 28}, {14,28}}, // Student 3
    {{17,26}, {18, 27}, {20,30}}}; // Student 4
```

Given the array above, write a program that includes a method called `getAverageForStudent` takes a multi-dimensional array and a student index as parameters, and returns the average score for a given student.

For example, if I passed in the above array and an ID of 0, I’d get 40.0
public class ComputeScores {

    public static void main(String[] args) {
        int[][][][] scores = {
            {{20,30}, {15, 25}, {10,20}}, // Student 1
            {{20,30}, {13, 22}, {16,25}}, // Student 2
            {{19,29}, {14, 24}, {12,22}}, // Student 3
            {{18,30}, {12, 28}, {14,28}}, // Student 4
            {{17,26}, {18, 27}, {20,30}}}; // Student 5

        System.out.println("The average score for the student is " + getAverageForStudent(scores, 0) * 2 + "]%";
    }

    ...
public static double getAverageForStudent(int[][][] scores, int studentID){
    double average = 0.0;
    int allExamTotal = 0;
    for (int i = 0; i < scores[studentID].length; i++) { // Loop over just the student's scores
        int testTotal = 0;
        for (int j = 0; j < scores[studentID][i].length; j++) { // Loop over both sections for each exam
            testTotal += scores[studentID][i][j]; // Add to the current test total
        }
        allExamTotal += testTotal; // Add test total to the all-exam total
    }
    average = allExamTotal / scores[studentID].length; // compute average
    return average;
}