Overview

- We’re experts now at storing data in an ordered sequence: arrays, lists
- Our ordered collections have only been in one dimension so far, though.
- Sometimes we want to represent data in multiple dimensions
  - Images as grids of pixels, arrays at different points in time, matrices
Learning Objectives

After this lecture, you should be able to...

○ declare a 2D array
○ initialize a 2D array with the new keyword or with an initializer list
○ identify the dimensions of a 2D array
○ access 2D array values
○ modify 2D array values
○ traverse a 2D array using the for-loop
○ solve problems using 2D arrays
Two-Dimensional Arrays

- A one-dimensional array stores an ordered sequence of elements.
- A two-dimensional array can be thought of as a table of elements, with rows and columns.

![Diagram of one-dimensional and two-dimensional arrays.](image)
## Arrays of Arrays

- Ultimately, a 2D array is an **array of arrays**.

2D arrays are declared and initialized by specifying the size of each dimension separately:

```
int[][] matrix = new int[3][7];
```
Declaration and Initialization

- Here’s the general way to initialize a *rectangular* 2D array:
  - `arrayType[][] arrayName = new arrayType[rows][cols];`

- To reference a single element at row $r$ and column $c$:
  - `int value = arrayName[r][c];`

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrayName</td>
<td>int[][]</td>
<td>2D array of integers, or an array of integer arrays</td>
</tr>
<tr>
<td>arrayName[3]</td>
<td>int[]</td>
<td>Array of integers</td>
</tr>
<tr>
<td>arrayName[3][4]</td>
<td>int</td>
<td>integer</td>
</tr>
</tbody>
</table>
Iterating over a Jagged 2D Array

```java
int nRows = 10;
int nCols = 5;
double[][][] a = new double[nRows][nCols];
for (int i = 0; i < nRows; i++) {
    for (int j = 0; j < nCols; j++) {
        a[i][j] = 1.0;
    }
}
```

*Initialize a 2D array with 10 rows and 5 columns. Then, set each value to 1.0.*
2D Arrays Can Be Jagged/Ragged

- A jagged (or ragged) 2D array has different numbers of columns in each row.
  - Alternatively, each array stored in the array of arrays has a different length.

```java
int numRows = 9;
double[][] a = new double[numRows][];
for (int i = 0; i < a.length; i++) {
    a[i] = new double[numRows - i];
}
```

Initializing an array of null arrays, then initializing each inner array with a different length.
Iterating over a Jagged 2D Array unsafely

```java
int numRows = 9;
double[][] a = new double[numRows][];
for (int i = 0; i < numRows; i++) {
    a[i] = new double[numRows - i];
}
for (int i = 0; i < numRows; i++) {
    for (int j = 0; j < nCols; j++) {
        a[i][j] = 1.0;
    }
}
```

What happens if we run this code?
Iterating over a Jagged 2D Array Safely

```java
int numRows = 9;
double[][] a = new double[numRows][];
for (int i = 0; i < numRows; i++) {
a[i] = new double[numRows - i];
}

for (int i = 0; i < a.length; i++) {
    for (int j = 0; j < a[i].length; j++) {
        a[i][j] = 1.0;
    }
}
```
Explicit 2D Array Initialization

- `int scores[][] = {{44, 55, 66, 77}, {36}, {87, 97}, {68, 78, 88}};`

- (Write out the array of arrays.)