Two-Dimensional Arrays

CST 231 - Week 13

What We'll Cover Today

- What are 2D arrays?
- Creating and accessing 2D arrays
- The nested loop pattern
- Working with rows and columns

Why 2D Arrays?

1D Arrays

We've been working with 1D arrays (lists of values):

```
[5, 3, 8, 2, 9]
```

2D Arrays

But sometimes we need to organize data in a grid or table:

```
5  3  8
2  9  1
7  4  6
```

Examples: game boards, spreadsheets, images, matrices

Creating a 2D Array

Method 1: Specify size (empty array)

```
int[][] grid = new int[3][4];
// 3 rows, 4 columns
```

Method 2: Initialize with values (using braces)

```
int[][] grid = {
    {1, 2, 3, 4},
    {5, 6, 7, 8},
    {9, 10, 11, 12}
};
```

Both create a 3x4 grid (3 rows, 4 columns)

Accessing Elements

Use [row][col]

You must use both indices:

```
int[][] grid = new int[3][4];
grid[0][0] = 5;   // top-left
grid[2][3] = 9;   // bottom-right
grid[1][2] = 7;   // middle somewhere
```

Common Error

You cannot do this:

```
grid[5] = 10;
// ERROR! Must specify both row and column
```

Understanding .length

Important: `array.length` gives you the number of **ROWS**, not total elements!

Why? Each row is actually a separate 1D array!

```
int[][] grid = new int[3][4];

System.out.println(grid.length);
// prints 3 (rows)

System.out.println(grid[0].length);
// prints 4 (columns in row 0)

System.out.println(grid[1].length);
// prints 4 (columns in row 1)
```

The Nested Loop Pattern

To process every element in a 2D array, use nested loops:

- Outer loop: goes through each row
- Inner loop: goes through each column in that row

```
for (int row = 0; row < grid.length; row++) {
    for (int col = 0; col < grid[row].length;
col++) {
        // do something with grid[row][col]
    }
}
```

Common Pattern - Filling an Array

```
for (int row = 0; row < grid.length; row++) {
    for (int col = 0; col < grid[row].length; col++) {
        grid[row][col] = someValue;
    }
}
```

Common Pattern - Printing an Array

```
for (int row = 0; row < grid.length; row++) {
    for (int col = 0; col < grid[row].length; col++) {
        System.out.print(grid[row][col] + " ");
    }
    System.out.println();  // new line after each row
}
```

Sometimes You Need More Loops

For some operations, you need a loop around your nested loops (That's three loops total!):

```
// For each number 0-9
for (int target = 0; target < 10; target++) {

    // Search the entire array
    for (int row = 0; row < grid.length; row++) {
        for (int col = 0; col < grid[row].length; col++) {
            // check if grid[row][col] == target
        }
    }
}
```

Key Takeaways

- 2D arrays organize data in rows and columns
- Must use both indices: `array[row][col]`
- `array.length` = number of rows
- `array[row].length` = number of columns in that row
- Nested loops: outer = rows, inner = columns
- Each row is an independent 1D array

Now Let's Code!

Then you'll practice with file I/O and 2D arrays!

Fill an Array

We'll build a method to fill a 2D array with random numbers.

Print it Nicely

We'll build a method to print the 2D array to the console.

Three Loops

We'll build a method that uses three loops to search the array.