My Analysis of a Two-Dimensional Array Implementation

This code is basically just a simple example of how to work with a 2D array in Java. The main point is to show off the two common ways to loop through one: using a regular for loop with indexes to put stuff in, and then using the easier for-each loop to read everything back out.

It sets up a 2D integer array called nums. The interesting thing to remember about Java is that a 2D array isn't a true matrix like we might think. It's actually an array of arrays. So when we say new int[3][2], we're first making an outer array that can hold three items, and each of those three items is its own separate array that holds two integers.

To fill it up, it uses a classic nested for loop, one for the rows and one for the columns. It calculates the value for each spot with the formula (row + 1) * (col + 1). So, if we trace it, nums[0][0] becomes (0+1) * (0+1), which is 1. Then nums[0][1] is (0+1) * (1+1), which is 2, and so on. In the end, we get an array that looks like {{1, 2}, {2, 4}, {3, 6}}.

Then, to print everything out, it switches to a for-each loop, which is honestly way cleaner since you don't have to mess with indexes. The outer loop, for(int[] r : nums), grabs an entire row at a time, like the {1, 2} array first. Then the inner loop, for(int c : r), goes through each individual number in that row. As it loops, it just prints the number and adds it to a sum variable.

So, the output in the console should look like this:

Length of rows: 3

Length of columns: 2

1 2

2 4

3 6

Summation: 18

This all explains why the console output looks the way it does. First, nums.length gives us 3, which is the number of rows. The loops print each number followed by a space, and the println after the inner loop makes sure each row is on a new line. Finally, it just prints the total sum, which is 18 if we add all the numbers up (1 + 2 + 2 + 4 + 3 + 6).