Geevoi A. Plaza September 17, 2025

CS-2105 CS 211

S1: Purpose and Context

1. What is the goal of the code?
   - The main goal of the code is to compute the sum of all elements in the array while printing them. It aims to demonstrate how to work with a 2D array including initialization, inputting values, and traversal.
2. What kind of data or structure does the code operate on?
   - The code operates on a 3×2 two-dimensional **array** of integers (`int[][]`).

S2: Control Flow and Structure

1. Identify the main constructs: loop types, array declarations, method signatures.
   - Array Declaration: `int[][] nums = new int[3][2];` creates the 2D array.
   - Loops:
     - Nested for loop is used to assign values:
       ```
       for(int row = 0; row < nums.length; row++) {
           for(int col = 0; col < nums[row].length; col++) {
                   nums[row][col] = (row + 1) * (col + 1);
           }
       }
       ```
     - Nested for-each loop is used to read, display and sum values
       ```
       for(int[] rvals : nums) {
           for(int cvals : rvals) {
               System.out.print(cvals + " ");
               sum += cvals;
           }
           System.out.println();
       }
       ```

- Method Signatures: From the method declaration, `public static void main(String[] args)`, which includes the:
    - access modifier (`public`);
    - return type (`void`);
    - method name (`main`); and
    - parameter list (`String[] args`).

2. How did the "for-each" loop iterate through rows and columns? How does it navigate the 2D array?

- The outer for-each loop (`for(int[] rvals : nums) {...}`) retrieves each row (`rvals`) as a 1D array. Then, in each iteration of rows, the inner for-each loop (`for(int cvals : rvals) {...}`) retrieves each element (`cvals`) in that row (`rvals`). This allows for navigation of every element of every row of the 2D array without explicitly managing indices.

S3: Behavior & Output

1. What values does the code process, and how does it output or manipulate data?

- The code assigns values to the array based on `(row + 1) * (col + 1)`, producing a multiplication pattern. Using nested for loops, it assigns the resulting values into the 2D array. For output, it switches to a for-each loop that prints the elements element by element, row by row while also summing them.

2. If you execute it, what would you expect to see, and why?

- When the program is executed, it will print the array values for each element of each row based on the formula previously shown, outputting:
    - `1 2`
    - `2 4`
    - `3 6`
- Finally, it will print the total sum of all elements:
    - `Summation: 18`
- This happens because the array is filled using the formula `(row + 1) * (col + 1)`, which produces `[ [1, 2], [2, 4], [3, 6] ]`. The for-each loop then

prints these values in a table-like format while accumulating their sum, which equals 18.

S4: Summary

The program is a good reference to contrast regular loops (which was used for populating data) with for-each loops (which was used for traversing the data), showing when each is appropriate.

A for-each loop is used to traverse the 2D array by automatically retrieving each row as a 1D array and then each element inside that row. In the program, the outer for-each loop handles the rows, while the inner for-each loop handles the values inside them. This makes it simple to display and sum the elements without worrying about indices.

I <3 for-each loops ye.