Republic of the Philippines

BATANGAS STATE UNIVERSITY

The National Engineering University

Alangilan Campus

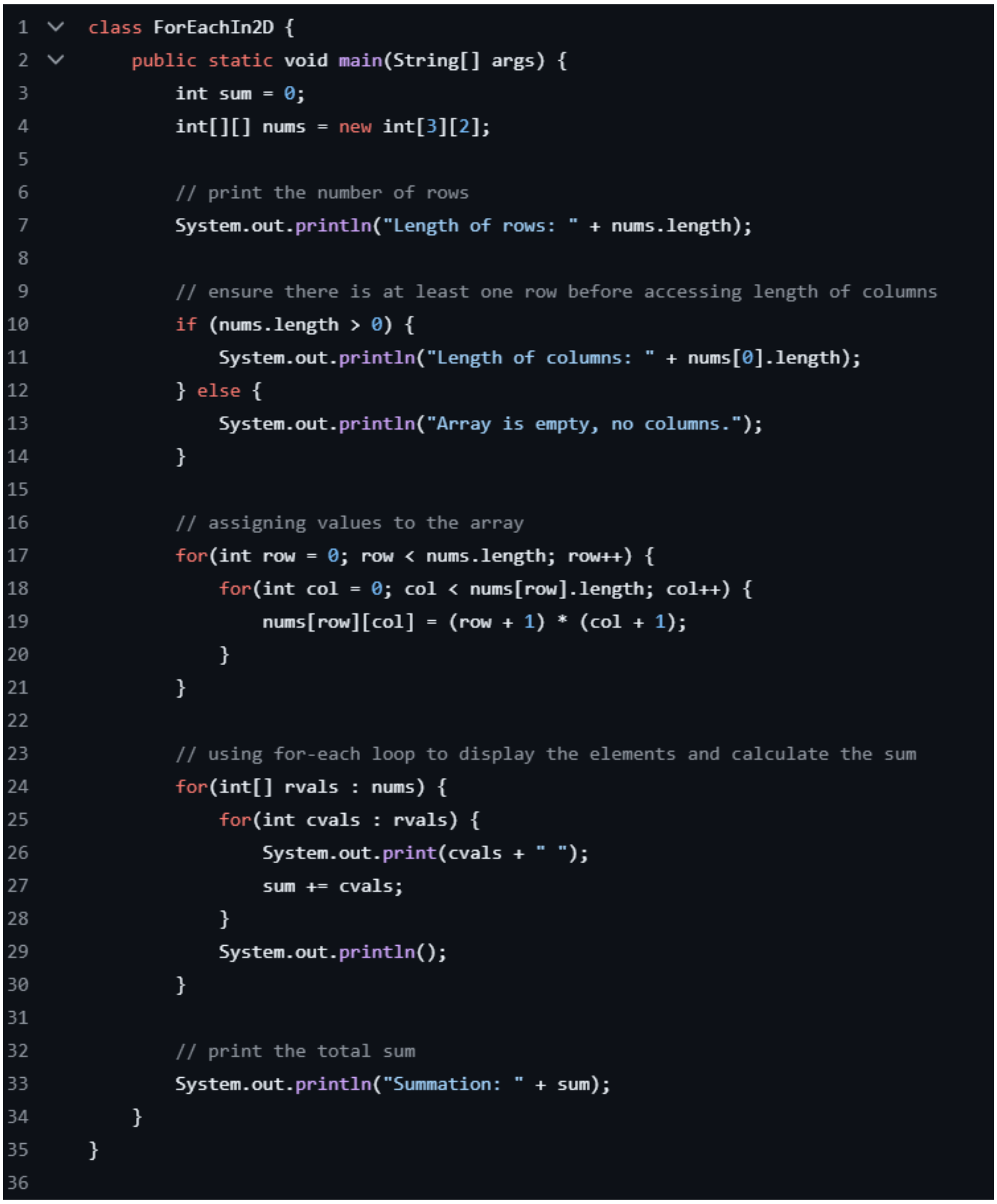
Golden Country Homes, Alangilan, Batangas City, Philippines 4200
Tel Nos.: (0 43) 425-0139; 425-0143 local 2223
E-mail Address: cics.alangilan@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

PRACTICE ASSESSMENT 2

CS 211 – Object Oriented Programming
Computer Science Department
1st Semester AY 2025-2026

Name: Alyzza Monique Q. Aragon
Date: September 17, 2025
Section: CS- 2101
SR-Code: 24-02347



```java
class ForEachIn2D {
    public static void main(String[] args) {
        int sum = 0;
        int[][] nums = new int[3][2];

        // print the number of rows
        System.out.println("Length of rows: " + nums.length);

        // ensure there is at least one row before accessing length of columns
        if (nums.length > 0) {
            System.out.println("Length of columns: " + nums[0].length);
        } else {
            System.out.println("Array is empty, no columns.");
        }

        // assigning values to the array
        for(int row = 0; row < nums.length; row++) {
            for(int col = 0; col < nums[row].length; col++) {
                nums[row][col] = (row + 1) * (col + 1);
            }
        }

        // using for-each loop to display the elements and calculate the sum
        for(int[] rvals : nums) {
            for(int cvals : rvals) {
                System.out.print(cvals + " ");
                sum += cvals;
            }
            System.out.println();
        }

        // print the total sum
        System.out.println("Summation: " + sum);
    }
}

```

Step 1. Understand the Purpose and Context

What is the goal of the code?

- It is to showcase how to incorporate 2D array matrix into a code by utilizing efficient loops and proper code convention. The first goal is to print the length of rows from the specified size of the array which is declared at line 4 of the code hence to also verify if the length of the matrix columns are not empty through if-else statement. After that, inside a nested for-loop is the assigning of values into the array. Then through a for-each loop the program, it will display the elements of the array and calculate its sum. Lastly, is to reveal the overall summation of the elements.

What kind of data or structure does the code operate on?

- The kind of structure utilized in the code is an array which is specified to be a 2D-array matrix as declared in line 4 and as per definition, an array is a fixed-size collection of element that is being accessed by index and is a perfect option for matrices (int[3][2]) .Through dynamic memory allocation, it uses the keyword new to dynamically allocate memory into the array. Each element is accessed through rows and columns indices. As it represents a grid-like structure, We have learned that we can maintain it’s space alignment by adding modification to the placements of the digits as it increases to the right side.

Step 2. Examine Control Flow and Structure

Identify the main constructs: loop types, array declarations, method signatures.

- **Array declaration**: In line 4, we have declared the array in this way (int[][] nums = new int[3][2] as it creates a matrix with 3 rows and 2 columns with an initial default values;

- **Loop types**: Nested for loops – is used to assign values to each elements in the 2D array, iterates by row then by column. For-Each loop – Traverses the 2D array in a readable way.

- **Method Signatures**: The constructs of the method signatures are as follows: Access modifier (public) since the code is accessible anywhere; static – since it belongs to the class; void – since it returns nothing; main – since it is the starting point of the program and; String[] args – as it accepts command-line arguments.

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

- It is to demonstrate how to traverse a 2D array using a for-each loop in a structured and readable way. The outer loop access each row from the matrix, while the inner loop iterates through the individual elements of that row. This method avoids the use of index variables, making the code more focused on data processing. Through this loop structure, the program prints each value and computes the total sum. Lastly, it outputs the final summation of all elements in the array.

Step 3. Consider Behavior & Output

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

- On the assigning of values in the array, row loops 0, 1 and 2 (because nums.length is 3) while col loops 0 and 1 (because nums[row].length is 2). To tabulate it:

row = 0, col = 0	(0+1) * (0+1) = 1*1 = 1	nums[0][0] = 1
row = 0, col = 1	(0+1) * (1+1) = 1*2 = 2	nums[0][1] = 2
row = 1, col = 0	(1+1) * (0+1) = 2*1 = 2	nums[1][0] = 2
row = 1, col = 1	(1+1) * (1+1) = 2*2 = 4	nums[1][1] = 4
row = 2, col = 0	(2+1) * (0+1) = 3*1 = 3	nums[2][0] = 3
row = 2, col = 1	(2+1) * (1+1) = 3*2 = 6	nums[2][1] = 6

Hence the array looks like this,

row0 = [1, 2]

row1 = [2, 4]

row2 = [3, 6]

The code processes integers stored in the matrix and it print them and sum them.

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

- I expected the length of rows to be 3 since in line 7, I printed the initialized size of the array. In the length of columns it verifies first if the nums.length is greater than 0 then it proceeds into printing the length of columns which is 2 by looking through the elements in row 1(nums[0].length). Then through the nested- for loop the assignment of values into the array is being implemented. On the next for-each loop I expected to see the arranged array output which is row0 = [1, 2] | row1 = [2, 4] | row2 = [3, 6]. Then finally, when printing the summation it automatically calculates the sum of all the elements and it must show Summation: 18.

Step 4. Summarize Your Insights

Conclude with a concise reflection, using the guide I provided.

- This code demonstrates the fundamentals of working with a 2D array in Java, it showcases how arrays can be declared, traversed, and manipulated through different loop structures. Its main purpose is to show the full workflow from verifying array size, assigning values with nested for loops, to reading and processing data with a for-each loop. The use of (row+1) * (col+1) during assignment creates a multiplication table-like pattern, which makes the logic clear and structured. The program outputs the size of the array, prints the matrix in a row-by-row structure, and then displays the summation of all elements, which equals 18. This verifies the correct assignment and iteration. In conclusion, it shows the importance of selecting the right control structure for clarity and efficiency, making it a strong foundational example of matrix operations in Java.