9.3-Multi Dimensional Array

Understanding multi-dimensional arrays

A multi-dimensional array in Java is essentially an array of arrays. This structure allows us to store data in a matrix format, where data is organized in rows and columns. Think of it as a table where each cell can hold a value. For example, if you want to store a grid of numbers, a multi-dimensional array is the perfect choice.

Example of a Single-Dimensional Array

```
int nums[] = {3, 7, 2, 4};
```

In this example, nums is a single-dimensional array containing four elements. Now, imagine combining several such arrays. This combination leads to a multi-dimensional array, which can be visualized as a table with rows and columns.

Syntax to Declare a Multi-Dimensional Array in Java

There are several ways to declare a multi-dimensional array in Java:

```
dataType[][] arrayRefVar;     // Commonly used

dataType [][]arrayRefVar;

dataType arrayRefVar[][];

dataType []arrayRefVar[];
```

Example:

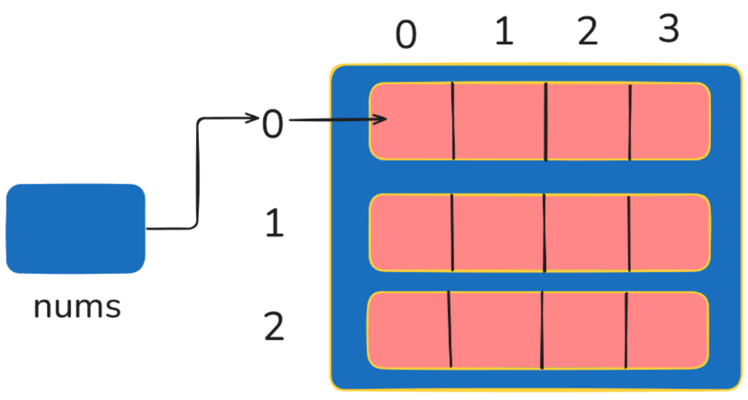
```
int nums[][] = new int[3][4];
```

This declaration creates a 2D array with 3 rows and 4 columns. Each row is an array containing 4 elements

In Java, arrays can have unequal rows, meaning not all rows need to have the same number of columns.

Understanding the Structure

In the example **int nums[][] = new int[3][4];**, the first bracket (i.e., 3) represents the number of rows, and the second bracket (i.e., 4) indicates the number of columns in each row.



TELUSKO

.

Accessing and Iterating Over Multi-Dimensional Arrays

Using Nested Loops: To access elements in a multi-dimensional array, nested loops are often used. The outer loop iterates over the rows, while the inner loop iterates over the columns.

```
for (int i = 0; i < 3; i++) {     // Iterate over rows
    for (int j = 0; j < 4; j++) { // Iterate over columns
        System.out.print(nums[i][j] + " ");
    }
    System.out.println();         // Move to the next line after each row
}
```

This code snippet will print out the elements of the 2D array row by row. Initially, all values will be 0 since the default value of an int in Java is 0.

Generating Random Values in a 2D Array

We can populate the array with random values using the Math.random() method. Since this method returns a double, it needs to be cast to int, and the values can be scaled as needed.

```
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 4; j++) {
        nums[i][j] = (int) (Math.random() * 10);
        System.out.print(nums[i][j] + " ");
    }
    System.out.println();
}
```

This will fill the array with random numbers between 0 and 9 and print the array's contents.

Enhanced For Loop

You can also use an enhanced for loop to iterate over multi-dimensional arrays. However, in a 2D array, the outer loop returns entire arrays (i.e., rows), and the inner loop returns individual elements within those rows.

```
for (int[] row : nums) {
    for (int element : row) {
        System.out.print(element + " ");
    }
    System.out.println();
    }
```

This approach is cleaner and often easier to read, especially when dealing with arrays that don't require complex indexing.