جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم
قسم الانظمة الطبية الذكية

Lecture: (3)

Arrays Part III

Subject: Computer Programming II
Level: First
Lecturer: Dr. Maytham N. Meqdad

Multidimensional Arrays in Java

Multidimensional Arrays can be defined in simple words as array of arrays. Data in multidimensional arrays are stored in tabular form (in row major order).

Syntax:

data_type[1st dimension][2nd dimension][]..[Nth dimension] **array_name** = **new data_type**[size1][size2]….[sizeN];

Where:

- **data_type**: Type of data to be stored in the array. For example: int, char, etc.
- **dimension**: The dimension of the array created. For example: 1D, 2D, etc.
- **array_name**: Name of the array
- **size1, size2, …, sizeN**: Sizes of the dimensions respectively.

Examples:

```
Two dimensional array:
int[][] twoD_arr = new int[10][20];

Three dimensional array:
int[][][] threeD_arr = new int[10][20][30];
```

Size of multidimensional arrays: The total number of elements that can be stored in a multidimensional array can be calculated by multiplying the size of all the dimensions.

For example: The array **int[][] x = new int[10][20]** can store a total of (10*20) = 200 elements. Similarly, array **int[][][] x = new int[5][10][20]** can store a total of (5*10*20) = 1000 elements.

- Example program demonstrating the use of a four-dimensional (4D) array in Java. This program initializes a 4D array, assigns values to it, and prints those values.

```
public class FourDimensionalArrayExample {
    public static void main(String[] args) {
        // Declare and initialize a 4D array with dimensions 2x3x4x5
        int[][][][] array = new int[2][3][4][5];

        // Assign values to the array
        int value = 1;
        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array[i].length; j++) {
                for (int k = 0; k < array[i][j].length; k++) {
                    for (int l = 0; l < array[i][j][k].length; l++) {
                        array[i][j][k][l] = value++;
                    }
                }
            }
        }

        // Print the array
        System.out.println("The 4D array:");
        for (int i = 0; i < array.length; i++) {
            System.out.println("Dimension 1, index " + i + ":");
            for (int j = 0; j < array[i].length; j++) {
                System.out.println("  Dimension 2, index " + j + ":");
                for (int k = 0; k < array[i][j].length; k++) {
                    System.out.print("    Dimension 3, index " + k + ": ");
                    for (int l = 0; l < array[i][j][k].length; l++) {
                        System.out.print(array[i][j][k][l] + " ");
                    }
                    System.out.println();
                }
            }
        }
    }
}
```

- When you run this program, it will output the values assigned to the 4D array in a structured format. The exact output is quite lengthy due to the size of the array, but it will show the values organized by each dimension, such as:

The 4D array:
Dimension 1, index 0:
Dimension 2, index 0:
Dimension 3, index 0: 1 2 3 4 5
Dimension 3, index 1: 6 7 8 9 10

```
    Dimension 3, index 2: 11 12 13 14 15
    Dimension 3, index 3: 16 17 18 19 20
  Dimension 2, index 1:
    Dimension 3, index 0: 21 22 23 24 25
    Dimension 3, index 1: 26 27 28 29 30
    Dimension 3, index 2: 31 32 33 34 35
    Dimension 3, index 3: 36 37 38 39 40
  Dimension 2, index 2:
    Dimension 3, index 0: 41 42 43 44 45
    Dimension 3, index 1: 46 47 48 49 50
    Dimension 3, index 2: 51 52 53 54 55
    Dimension 3, index 3: 56 57 58 59 60
Dimension 1, index 1:
  Dimension 2, index 0:
    Dimension 3, index 0: 61 62 63 64 65
    Dimension 3, index 1: 66 67 68 69 70
    Dimension 3, index 2: 71 72 73 74 75
    Dimension 3, index 3: 76 77 78 79 80
  Dimension 2, index 1:
    Dimension 3, index 0: 81 82 83 84 85
    Dimension 3, index 1: 86 87 88 89 90
    Dimension 3, index 2: 91 92 93 94 95
    Dimension 3, index 3: 96 97 98 99 100
  Dimension 2, index 2:
    Dimension 3, index 0: 101 102 103 104 105
    Dimension 3, index 1: 106 107 108 109 110
    Dimension 3, index 2: 111 112 113 114 115
    Dimension 3, index 3: 116 117 118 119 120
```

Two – dimensional Array (2D-Array)

Two – dimensional array is the simplest form of a multidimensional array. A two – dimensional array can be seen as an array of one – dimensional array for easier understanding.

- **Declaration – Syntax:**

```
data_type[][] array_name = new data_type[x][y];
        For example: int[][] arr = new int[10][20];
```

- **Initialization – Syntax:**

```
array_name[row_index][column_index] = value;
        For example: arr[0][0] = 1;
```

Example:

```
import java.io.*;

public class GFG {
    public static void main(String[] args)
    {

        int[][] arr = new int[10][20];
        arr[0][0] = 1;

        System.out.println("arr[0][0] = " + arr[0][0]);
    }
}
```

Output

```
arr[0][0] = 1
```

Example: Implementing 2D array with by default values with 4*4 matrix

```
public class TwoDArray {
    public static void main(String[] args) {
        int rows = 4;
        int columns = 4;

        int[][] array = new int[rows][columns];

        int value = 1;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                array[i][j] = value;
                value++;
            }
        }

        System.out.println("The 2D array is: ");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                System.out.print(array[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Output

```
The 2D array is:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

Representation of 2D array in Tabular Format:

A two – dimensional array can be seen as a table with 'x' rows and 'y' columns where the row number ranges from 0 to (x-1) and column number ranges from 0 to (y-1). A two – dimensional array 'x' with 3 rows and 3 columns is shown below:

	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

Print 2D array in tabular format:

To output all the elements of a Two-Dimensional array, use nested for loops. For this two for loops are required, One to traverse the rows and another to traverse columns.

Example:

```
import java.io.*;
class GFG {
    public static void main(String[] args)
    {

        int[][] arr = { { 1, 2 }, { 3, 4 } };

        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print(arr[i][j] + " ");
            }

            System.out.println();
        }
    }
}
```

Output

```
1 2
3 4
```

Example: Implementation of 2D array with User input

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args)
    {
        Scanner scan = new Scanner(System.in);

        System.out.print("Enter number of rows: ");
        int rows = scan.nextInt();

        System.out.print("Enter number of columns: ");
        int columns = scan.nextInt();

        int[][] multidimensionalArray= new int[rows][columns];

        // Now you can use the array like a regular
        // 2-dimensional array
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                multidimensionalArray[i][j]= (i + 1) * (j + 1);
            }
        }
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                System.out.print(multidimensionalArray[i][j]+ " ");
            }

            System.out.println();
        }
        scan.close();
    }
}
```

Output

```
Enter number of rows: 3
Enter number of columns: 3
1 2 3
2 4 6
3 6 9
```

Processing 2D-Arrays java

1. Initializing a 2D Array

You can initialize a 2D array in Java using either a direct assignment or a nested loop.

Direct Assignment:

```
int[][] matrix = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
```

Using Nested Loops:

```
int rows = 3;
int cols = 3;
int[][] matrix = new int[rows][cols];

for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        matrix[i][j] = i * cols + j + 1;
    }
}
```

2. Iterating Over a 2D Array

To iterate over the elements of a 2D array, you typically use nested loops.

Using Nested For Loops:

```
for (int i = 0; i < matrix.length; i++) { // Iterate over rows
    for (int j = 0; j < matrix[i].length; j++) { // Iterate over columns
        System.out.print(matrix[i][j] + " ");
    }
    System.out.println();
}
```

3. Performing Operations on a 2D Array

Calculating the Sum of All Elements:

```
int sum = 0;
for (int i = 0; i < matrix.length; i++) {
    for (int j = 0; j < matrix[i].length; j++) {
        sum += matrix[i][j];
    }
}
System.out.println("Sum of all elements: " + sum);
```

Finding the Maximum Element:

```
int max = matrix[0][0];
for (int i = 0; i < matrix.length; i++) {
    for (int j = 0; j < matrix[i].length; j++) {
        if (matrix[i][j] > max) {
            max = matrix[i][j];
        }
    }
}
System.out.println("Maximum element: " + max);
```

Transposing a 2D Array (Converting Rows to Columns):

```
int[][] transpose = new int[cols][rows];
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        transpose[j][i] = matrix[i][j];
    }
}
```

4. Example of a Complete Program

Example: initializes a 2D array, prints it, calculates the sum of its elements, and finds the maximum value:

```
public class ArrayProcessing {
    public static void main(String[] args) {
        // Initialize a 2D array
        int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        // Print the 2D array
        System.out.println("Original Matrix:");
```

```
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }

        // Calculate the sum of all elements
        int sum = 0;
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                sum += matrix[i][j];
            }
        }
        System.out.println("Sum of all elements: " + sum);

        // Find the maximum element
        int max = matrix[0][0];
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                if (matrix[i][j] > max) {
                    max = matrix[i][j];
                }
            }
        }
        System.out.println("Maximum element: " + max);
    }
}
```

This code demonstrates how to work with 2D arrays in Java, covering initialization, iteration, and basic operations. Modify and expand upon these examples based on your specific needs.