

Multidimensional Arrays



SoftUni Team
Technical Trainers



SoftUni

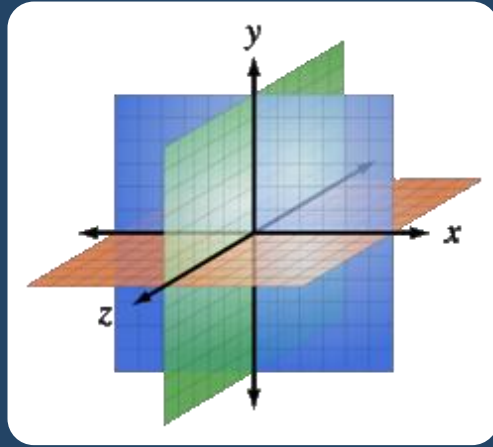


Software University

<https://softuni.bg>

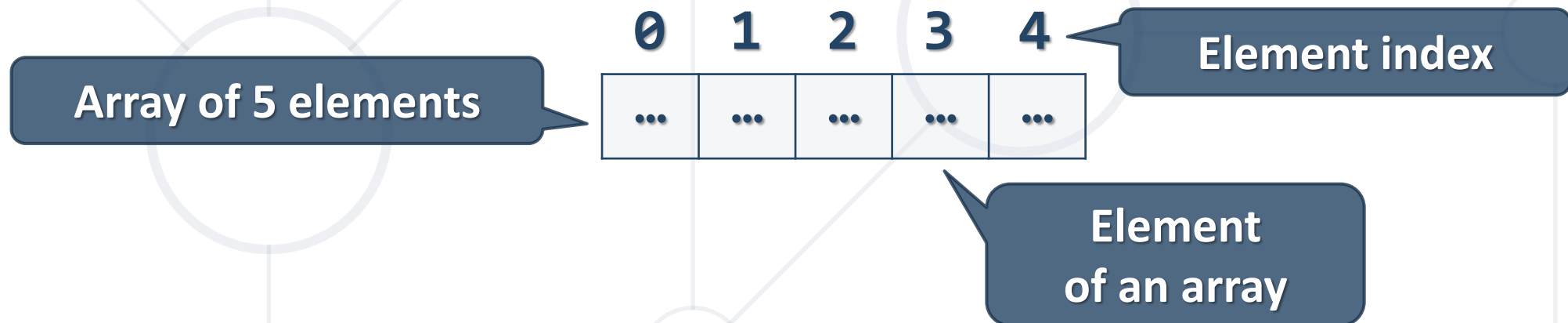
1. **Arrays** in Java
2. What is a **Multidimensional** Array?
3. **Declaring** and **Creating**
Multidimensional Arrays
4. **Initializing** Multidimensional Arrays
5. Accessing **Elements**
6. Reading a **Matrix**





Multidimensional Arrays

- In programming, an **array** is a sequence of elements
 - All elements are of the same type
 - The order of the elements is fixed
 - Has fixed size (**length**)



Working with Arrays in Java

- Allocating an array:

Array of 10 elements

```
int[] numbers = new int[10];
```

- Assigning values to the array elements:

```
for (int i = 0; i < numbers.length; i++)  
    numbers[i] = i + 1;
```

- Accessing array elements:

All elements are of the same type

```
numbers[3] = 20;  
numbers[5] = numbers[2] + numbers[7];
```

Element index

What is Multidimensional Array?

- An array is a systematic arrangement of similar objects
- Arrays can have more than one dimension, e.g. matrices
- The most used multidimensional arrays are the 2-dimensional

Matrix	COLUMNS			
R O W S	[0][0]	[0][1]	[0][2]	[0][3]
	[1][0]	[1][1]	[1][2]	[1][3]
	[2][0]	[2][1]	[2][2]	[2][3]
	[3][0]	[3][1]	[3][2]	[3][3]

Column Index

Row Index

- Declaring multidimensional arrays:

```
int[][] intMatrix;  
float[][] floatMatrix;  
String[][][] strCube;
```

- Creating a multidimensional array

- Use **new** keyword
- Must specify the size of at least one dimension

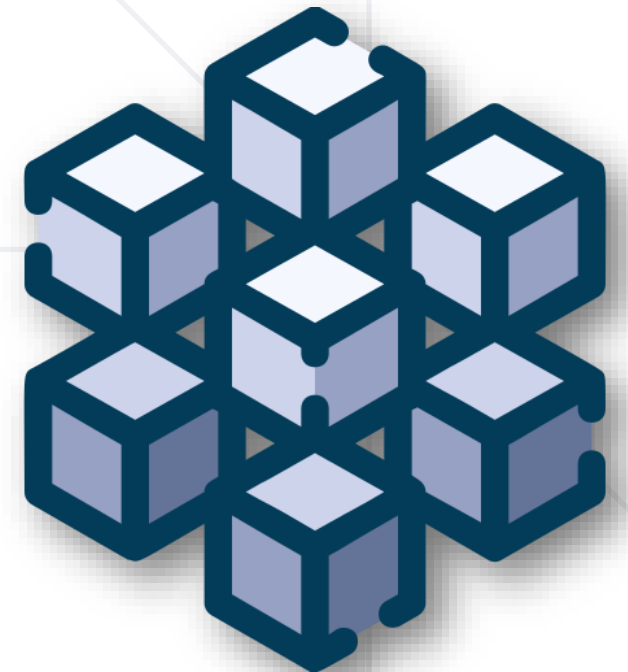
```
int[][] intMatrix = new int[3][];  
float[][] floatMatrix = new float[8][2];  
String[][][] stringCube = new String[5][5][5];
```

Initializing Multidimensional Arrays

- Initializing a multidimensional array with values:

```
int[][] matrix = {  
    {1, 2, 3, 4}, // row 0 values  
    {5, 6, 7, 8} // row 1 values  
};
```

- Matrices are represented by a list of rows
 - Each row consists of a list of values



- Accessing N-dimensional array element:

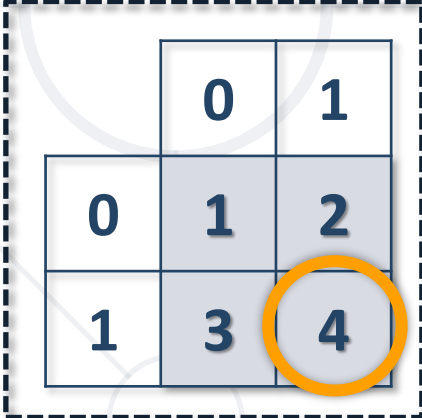
```
nDimensionalArray[index1] ... [indexn]
```

- Getting element value example:

```
int[][] array = {{1, 2}, {3, 4}};  
int element = array[1][1]; // element11 = 4
```

- Setting element value example:

```
int[][] array = new int[3][4];  
for (int row = 0; row < array.length; row++)  
    for (int col = 0; col < array[0].length; col++)  
        array[row][col] = row + col;
```



	0	1
0	1	2
1	3	4

Reading a Matrix – Example

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int rows = Integer.parseInt(scanner.nextLine());
    int cols = Integer.parseInt(scanner.nextLine());

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

    for (int row = 0; row < rows; row++) {
        String[] inputTokens = scanner.nextLine().split(" ");
        for (int column = 0; column < cols; column++) {
            matrix[row][column] =
                Integer.parseInt(inputTokens[column]);
        }
    }
}
```

Problem: Compare Matrices

- Write a program that reads **two integer matrices** (2D arrays) from the console and **compares** them element by element
- Print equal if the matrices match, and not equal if they don't match

Input	Output
2 1 2 3 2 1 3 2 1 2 3 2 1 3	Equal

Solution: Compare Matrices

```
int[] dimentions = Arrays.stream(scanner.nextLine())
                          .split("\\s++")
                          .mapToInt(Integer::parseInt)
                          .toArray();

int firstMatrixRows = dimentions[0];
int firstMatrixCols = dimentions[1];
// TODO: continue...
```

Solution: Compare Matrices

```
for (int i = 0; i < firstMatrixRows; i++) {  
    int[] arr = Arrays.stream(scanner.nextLine()  
                        .split("\\s+"))  
                        .mapToInt(Integer::parseInt)  
                        .toArray();  
    firstMatrix[i] = arr;  
}  
// TODO: read the second matrix...
```

Solution: Compare Matrices

```
static boolean matricesAreEqual(int[][] firstMatrix, int[][] secondMatrix) {  
    if (firstMatrix.length != secondMatrix.length) return false;  
    for (int row = 0; row < firstMatrix.length; row++) {  
        if (firstMatrix[row].length != secondMatrix[row].length)  
            return false;  
        for (int col = 0; col < firstMatrix[row].length; col++) {  
            if (firstMatrix[row][col] != secondMatrix[row][col]) return false;  
        }  
    }  
    return true;  
}
```

Problem: Positions of

- Write a program that reads a **matrix** of integers, then a **number** and prints all the positions at which that number appears in the matrix
- The matrix definition on the console will contain a line with two positive integer numbers **R** and **C**
- If the number does not appear in the matrix, print "**not found**"

Input	Output
2 3	0 1
1 2 3	1 1
4 2 2	1 2
2	

Solution: Positions of

```
// TODO Read matrix...
```

```
int searchNumber = Integer.parseInt(scanner.nextLine());
```

```
boolean isFound = false;
```

```
for (int row = 0; row < matrix.length; row++)
```

```
    for (int col = 0; col < matrix[row].length; col++)
```

```
        if (matrix[row][col] == searchNumber) {
```

```
            System.out.println(row + " " + col); isFound = true;
```

```
        }
```

```
if(!isFound)
```

```
    System.out.println("not found");
```


Problem: Sum of All Elements of Matrix

- Read a matrix from the console
- Print the number of **rows**
- Print the number of **columns**
- Print the sum of all **elements**

Input	Output
3, 6	3
7, 1, 3, 3, 2, 1	6
1, 3, 9, 8, 5, 6	76
4, 6, 7, 9, 1, 0	

Check your solution here: <https://alpha.judge.softuni.org/contests/multidimensional-arrays-lab/1459>

Solution: Sum of All Elements of Matrix

```
public static void main(String[] args) {  
    String sizes = scanner.nextLine();  
    int[][] matrix = matrixReader(sizes);  
    // TODO implement method matrixReader(String sizes)  
    System.out.println(matrix.length);  
    System.out.println(matrix[0].length);  
  
    int sum = 0;  
    for (int row = 0; row < matrix.length; row++) {  
        for (int col = 0; col < matrix[row].length; col++) {  
            sum += matrix[row][col];  
        }  
    }  
    System.out.println(sum);  
}
```

Gets length of 0th dimension (rows)

Gets length of 1st dimension (columns)

Problem: Maximum Sum of 2X2 Submatrix

- Find the 2x2 square with max sum in a given matrix
 - Read the matrix from the console
 - Find the biggest **sum** of 2x2 submatrix
 - Print the result in form of a new matrix

Input	Output
3, 6 7, 1, 3, 3, 2, 1 1, 3, 9, 8, 5, 6 4, 6, 7, 9, 1, 0	<div>9 8 7 9</div> <div>33</div>

Solution: Maximum Sum of 2X2 Submatrix

```
int bestSum = Integer.MIN_VALUE;
int resultRow;
int resultCol;
for (int row = 0; row < matrix.length - 1; row++)
    for (int col = 0; col < matrix[row].length - 1; col++)
        int sum = matrix[row][col] + matrix[row][col + 1] +
            matrix[row + 1][col] + matrix[row + 1][col + 1];
        if (sum > bestSum)
            bestSum = sum;
            resultRow = row;
            resultCol = col;
```

- **Multidimensional Array?**
 - Arrays can have more than one dimension, e.g. matrices
- **Declaring and Creating**
 - Use **new** keyword
- **Initializing Multidimensional Arrays**



- Software University – High-Quality Education, Profession and Job for Software Developers
 - softuni.bg, about.softuni.bg
- Software University Foundation
 - softuni.foundation
- Software University @ Facebook
 - facebook.com/SoftwareUniversity



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg>
- © Software University – <https://softuni.bg>

