

CS102

Programming II

2 DIMENSION ARRAYS

Prof. Dr. Basit Qureshi
PhD FHEA SMIEEE MACM
www.drbasit.org

TOPICS

- 2D arrays
- N-D arrays

These slides are modified from the source: Y. Daniel Liang, *Introduction to Java Programming and Data Structures*, 13th ed., Pearson, 2023.

TWO-DIMENSIONAL ARRAY ILLUSTRATION

	[0]	[1]	[2]	[3]	[4]
[0]	0	0	0	0	0
[1]	0	0	0	0	0
[2]	0	0	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

`matrix = new int[5][5];`

(a)

`matrix.length?` 5

`matrix[0].length?` 5

	[0]	[1]	[2]	[3]	[4]
[0]	0	0	0	0	0
[1]	0	0	0	0	0
[2]	0	7	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

`matrix[2][1] = 7;`

(b)

	[0]	[1]	[2]
[0]	1	2	3
[1]	4	5	6
[2]	7	8	9
[3]	10	11	12

```
int[][] array = {  
    {1, 2, 3},  
    {4, 5, 6},  
    {7, 8, 9},  
    {10, 11, 12}  
};
```

(c)

`array.length?` 4

`array[0].length?` 3

DECLARING, CREATING, AND INITIALIZING USING SHORTHAND NOTATIONS

You can also use an array initializer to declare, create and initialize a two-dimensional array. For example,

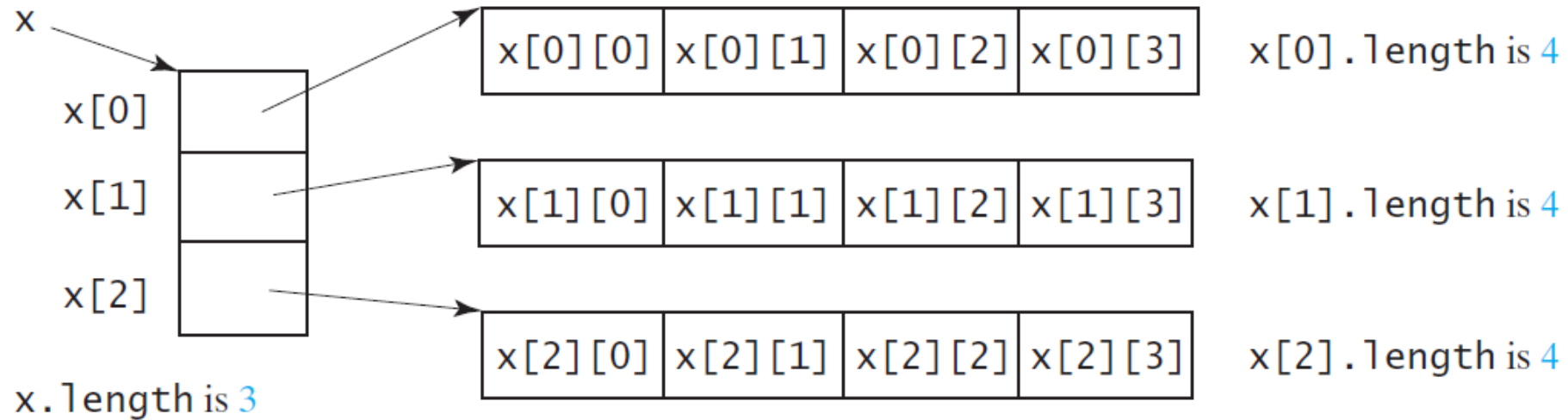
```
int[][] array = {  
    {1, 2, 3},  
    {4, 5, 6},  
    {7, 8, 9},  
    {10, 11, 12}  
};
```

Same as

```
int[][] array = new int[4][3];  
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;  
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;  
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;  
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

LENGTHS OF TWO-DIMENSIONAL ARRAYS

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



LENGTHS OF TWO-DIMENSIONAL ARRAYS, CONT.

```
int[][] array = {  
    {1, 2, 3},  
    {4, 5, 6},  
    {7, 8, 9},  
    {10, 11, 12}  
};
```

array.length

array[0].length

array[1].length

array[2].length

array[3].length

array[4].length

ArrayIndexOutOfBoundsException

RAGGED ARRAYS

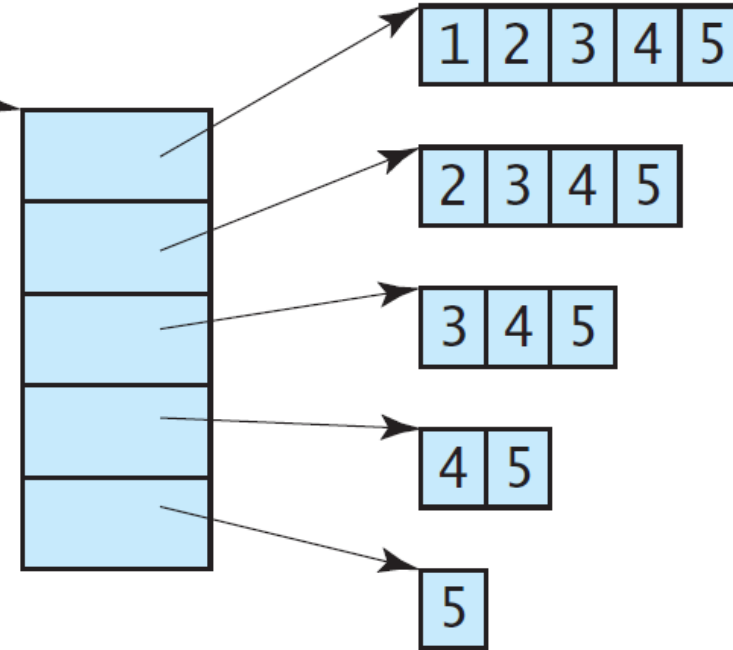
Each row in a two-dimensional array is itself an array. So, the rows can have different lengths. Such an array is known as *a ragged array*. For example,

```
int[][] matrix = {  
    {1, 2, 3, 4, 5},  
    {2, 3, 4, 5},  
    {3, 4, 5},  
    {4, 5},  
    {5}  
};
```

```
matrix.length is 5  
matrix[0].length is 5  
matrix[1].length is 4  
matrix[2].length is 3  
matrix[3].length is 2  
matrix[4].length is 1
```

RAGGED ARRAYS, CONT.

```
int[][] triangleArray = {  
    {1, 2, 3, 4, 5},  
    {2, 3, 4, 5},  
    {3, 4, 5},  
    {4, 5},  
    {5}  
};
```



PROCESSING TWO-DIMENSIONAL ARRAYS

Examples

1. (Initializing arrays with input values)
2. (Printing arrays)
3. (Summing all elements)
4. (Summing all elements by column)

INITIALIZING ARRAYS WITH INPUT VALUES

```
java.util.Scanner input = new Scanner(System.in);  
System.out.println("Enter " + matrix.length + " rows and " +  
    matrix[0].length + " columns: ");  
for (int row = 0; row < matrix.length; row++) {  
    for (int column = 0; column < matrix[row].length; column++) {  
        matrix[row][column] = input.nextInt();  
    }  
}
```

INITIALIZING ARRAYS WITH RANDOM VALUES

```
for (int row = 0; row < matrix.length; row++) {  
    for (int column = 0; column < matrix[row].length; column++) {  
        matrix[row][column] = (int) (Math.random() * 100);  
    }  
}
```

PRINTING ARRAYS

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

SUMMING ALL ELEMENTS

```
int total = 0;
for (int row = 0; row < matrix.length; row++) {
    for (int column = 0; column < matrix[row].length; column++) {
        total += matrix[row][column];
    }
}
```

SUMMING ELEMENTS BY COLUMN

```
for (int column = 0; column < matrix[0].length; column++) {  
    int total = 0;  
    for (int row = 0; row < matrix.length; row++)  
        total += matrix[row][column];  
    System.out.println("Sum for column " + column + " is " + total);  
}
```

MULTIDIMENSIONAL ARRAYS

Occasionally, you will need to represent **n-dimensional** data structures. In Java, you can create n-dimensional arrays for any integer n.

The way to declare two-dimensional array variables and create two-dimensional arrays can be generalized to declare n-dimensional array variables and create n-dimensional arrays for $n \geq 3$.

MULTIDIMENSIONAL ARRAYS

```
double[][][] scores = {  
    {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},  
    {{4.5, 21.5}, {9.0, 22.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},  
    {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},  
    {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},  
    {{8.5, 26.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},  
    {{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}}};
```

Which student

Which exam

Multiple-choice or essay

scores[i] [j] [k]

```
graph TD; A[Which student] --> B[scores[i]]; C[Which exam] --> D[scores[j]]; E[Multiple-choice or essay] --> F[scores[k]]
```