

COMP20220

Programming II (Conversion)

Michael O'Mahony

Chapter 8 Multidimensional Arrays

Motivations

Thus far, we have used one-dimensional arrays to model collections of elements...

Higher-dimensional arrays are also supported by Java.

Two-dimensional arrays can be used to represent matrices or tables. For example, the following table which describes the distances between cities can be represented using a two-dimensional array:

Distance Table (in miles)							
	Chicago	Boston	New York	Atlanta	Miami	Dallas	Houston
Chicago	0	983	787	714	1375	967	1087
Boston	983	0	214	1102	1763	1723	1842
New York	787	214	0	888	1549	1548	1627
Atlanta	714	1102	888	0	661	781	810
Miami	1375	1763	1549	661	0	1426	1187
Dallas	967	1723	1548	781	1426	0	239
Houston	1087	1842	1627	810	1187	239	0

Objectives

- ❑ To give examples of representing data using two-dimensional arrays (§8.1).
- ❑ To declare variables for two-dimensional arrays, create arrays, and access array elements in a two-dimensional array using row and column indexes (§8.2).
- ❑ To program common operations for two-dimensional arrays (§8.3).
- ❑ To pass two-dimensional arrays to methods (§8.4).
- ❑ To return two-dimensional arrays from methods (§8.4).
- ❑ Higher dimensional arrays

Declare/Create Two-dimensional Arrays

Syntax:

Declare array reference variable:

```
dataType[][] refVar;
```

Create array and assign its reference to a variable:

```
refVar = new dataType[nrows][ncols];
```

Combine declaration and creation in one statement:

```
dataType[][] refVar = new dataType[nrows][ncols];
```

Example:

A two-dimensional array with 10 rows and 15 columns:

```
int[][] matrix = new int[10][15];
```

Two-dimensional Array Examples

	[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

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

Two-dimensional Array Examples

	[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}  
};
```

Declaring, Creating, and Initializing Arrays Using Shorthand Notation

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

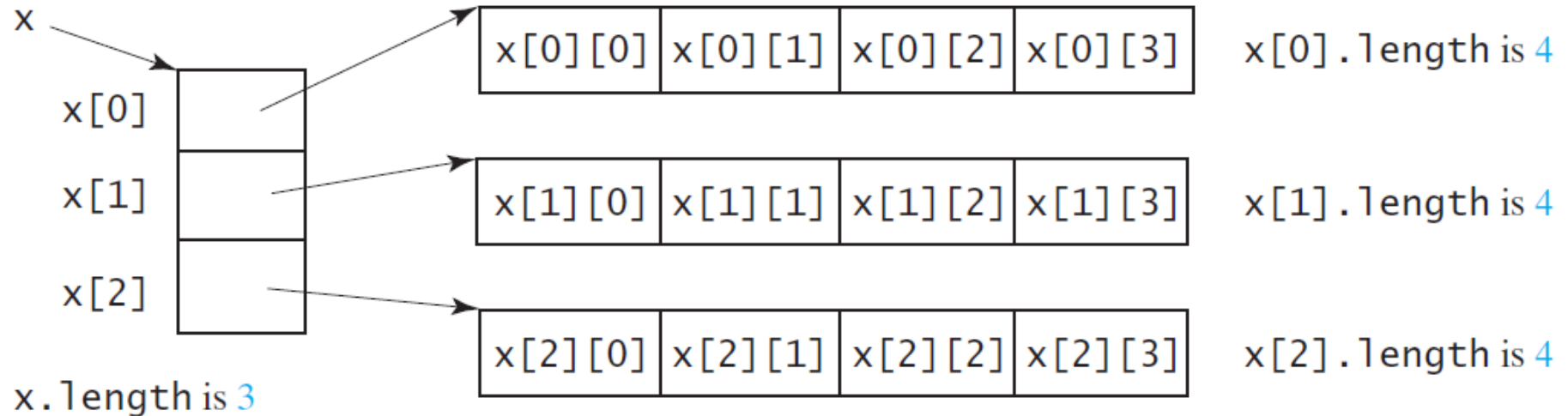
Same as

```
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

A two-dimensional array is actually an array in which each element is a one-dimensional array.

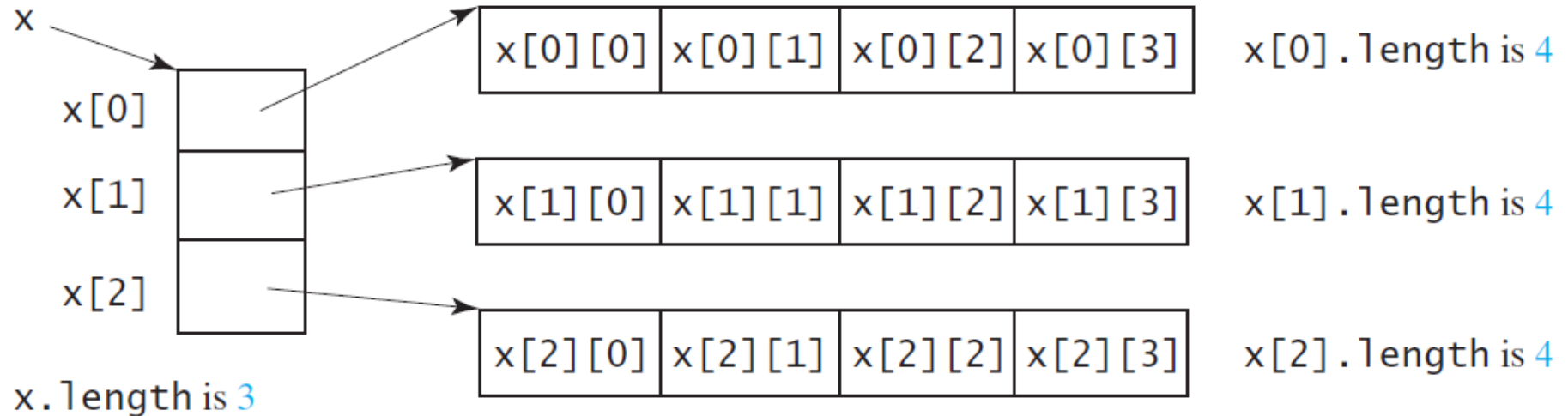
```
int[][] x = new int[3][4];
```



Lengths of Two-dimensional Arrays

A two-dimensional array is actually an array in which each element is a one-dimensional array.

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

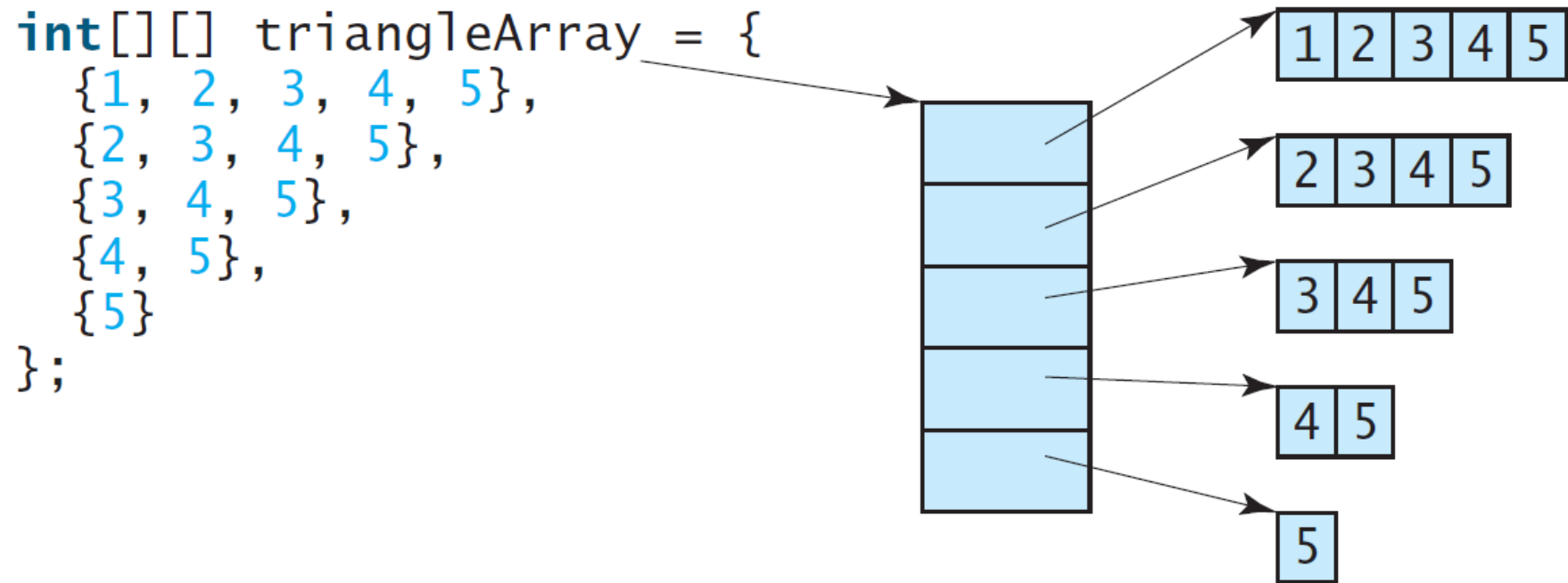


```
x[3].length => ArrayIndexOutOfBoundsException
```

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:



Ragged Arrays

If you do not know the values in a ragged array in advance, but do know the sizes, you can create a ragged array using the following syntax:



```
int[][] r = new int[5][];  
r[0] = new int[5];  
r[1] = new int[4];  
r[2] = new int[3];  
r[3] = new int[2];  
r[4] = new int[1];
```

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

Processing Two-Dimensional Arrays

Some examples:

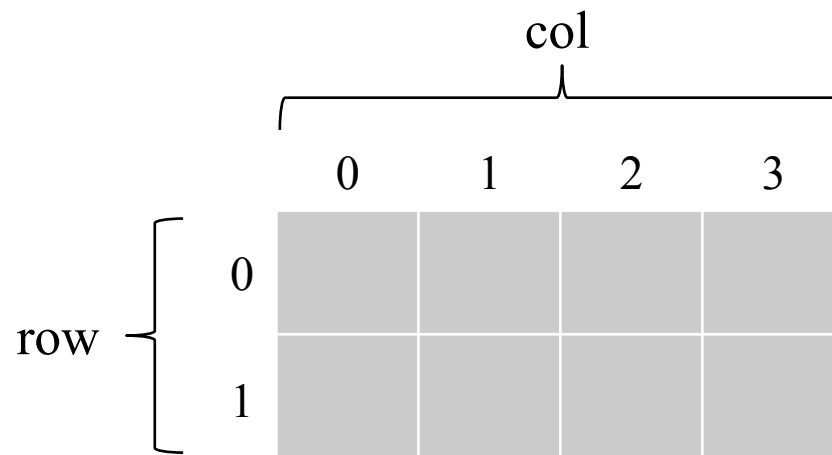
1. Initializing arrays with random values
2. Printing arrays
3. Summing all elements

Initializing arrays with random values

```
int[][] matrix = new int[2][4];
```

```
...
```

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



Printing arrays

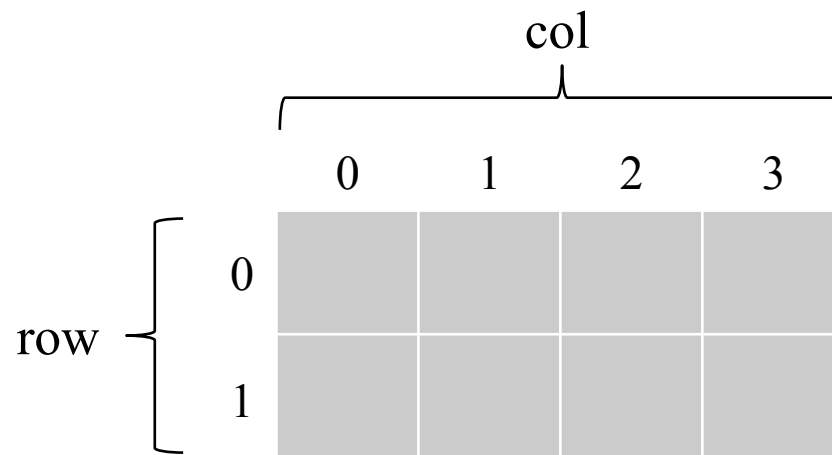
```
int[][] matrix = new int[2][4];
```

```
...
```

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

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

```
}
```



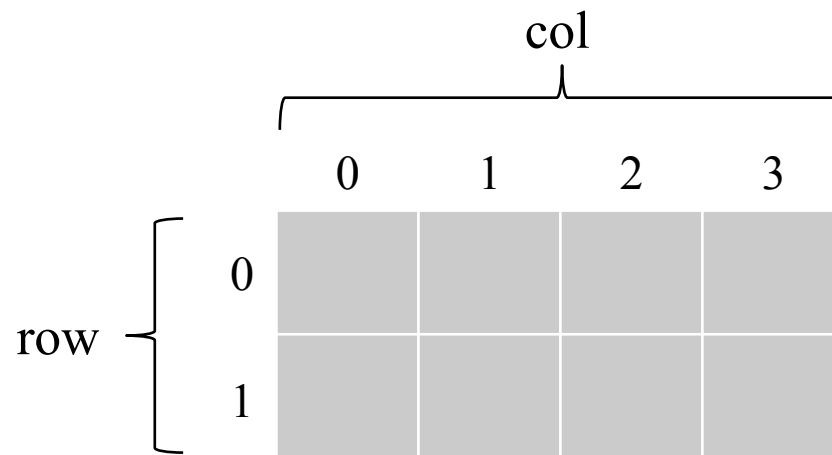
Summing all elements

```
int[][] matrix = new int[2][4];
```

```
...
```

```
int sum = 0;
```

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



Two-Dimensional Arrays Example

Example program showing how to:

- Pass a two-dimensional array to a method
- Return a two-dimensional array from a method

PassTwoDimensionalArray

Multidimensional Arrays

Previously – seen examples of using two-dimensional arrays to represent tables and matrices...

In Java, you can create n -dimensional arrays for any integer $n \geq 3$:

- A two-dimensional array consists of an array of one-dimensional arrays
- A three-dimensional array consists of an array of two-dimensional arrays
- ...

For example, use a three-dimensional array to store exam scores for a class of **six** students with **five** exams, and each exam has **two** parts (multiple-choice and essay).

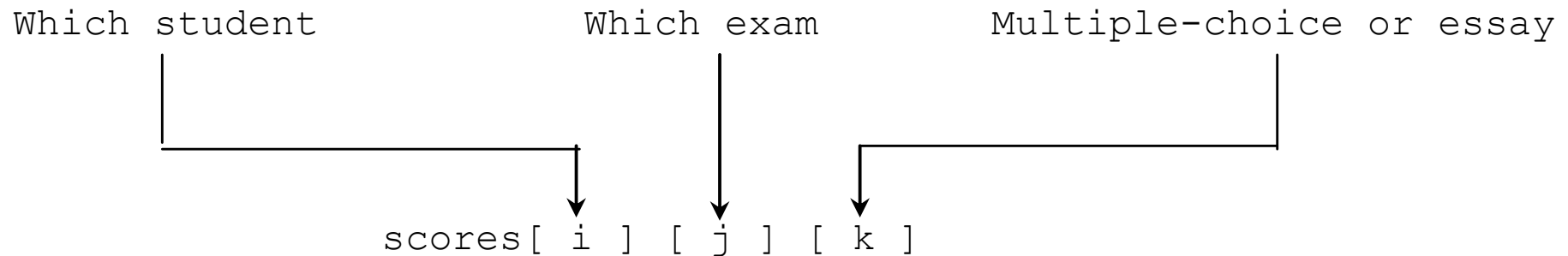
```
double[][][] scores = new double[6][5][2];
```

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}, {9.2, 21.5}, {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}}  
};
```

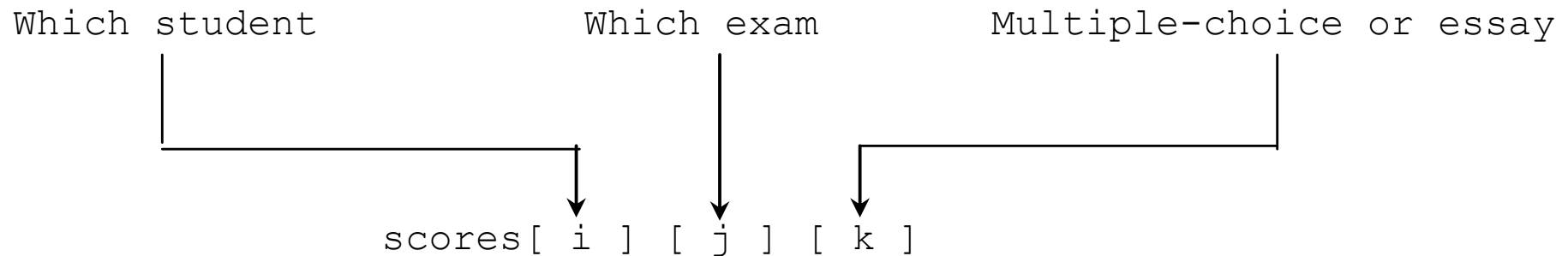
Multidimensional Arrays

```
double[][][] scores = {  
    {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},  
    {{4.5, 21.5}, {9.2, 21.5}, {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}}  
};
```



Multidimensional Arrays

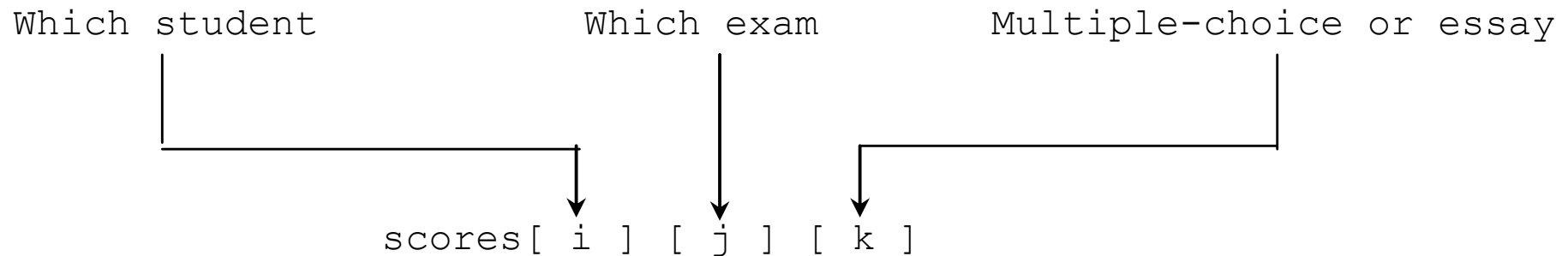
```
double[][][] scores = {  
    {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},  
    {{4.5, 21.5}, {9.2, 21.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},  
    {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},  
    {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},  
    {{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}}  
};
```



`scores[0][1][0]` refers to ??

Multidimensional Arrays

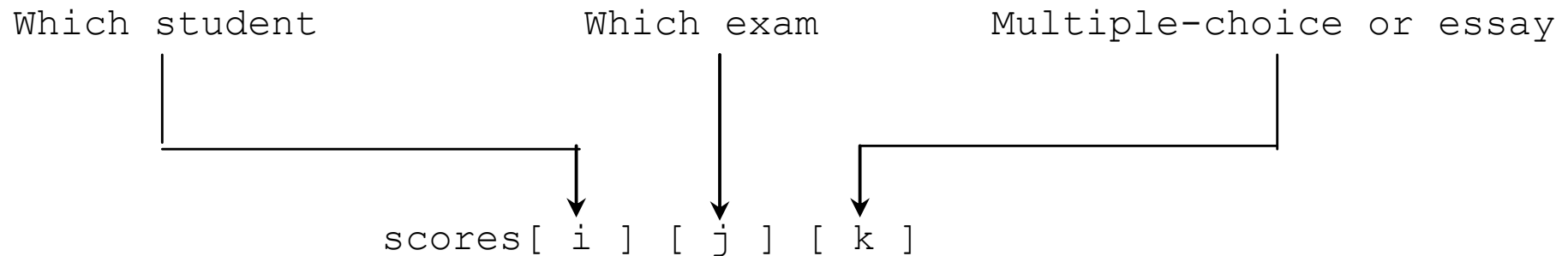
```
double[][][] scores = {  
    {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},  
    {{4.5, 21.5}, {9.2, 21.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},  
    {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},  
    {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},  
    {{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}}  
};
```



scores[0][1][0] refers to the **multiple-choice** score for the **first student's second exam**, which is **9.0**.

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}, {9.2, 21.5}, {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}}  
};
```

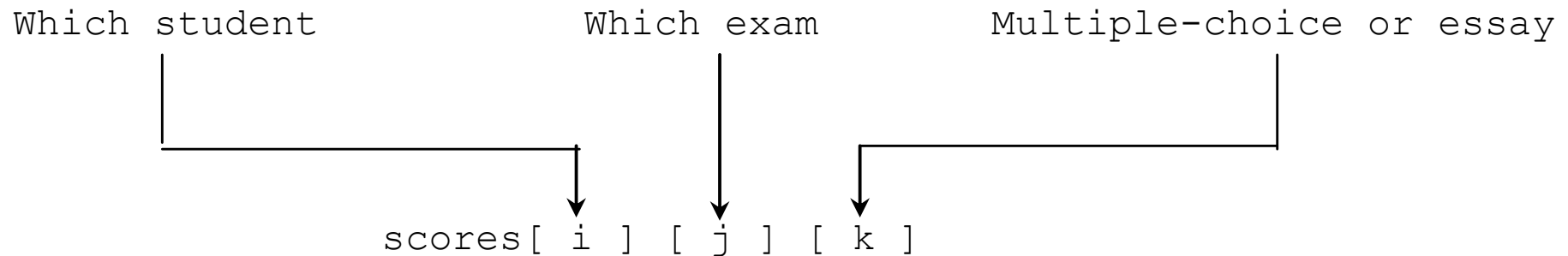


scores[0][1][0] refers to the **multiple-choice** score for the **first student's second exam**, which is 9.0.

scores[0][1][1] refers to ??

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}, {9.2, 21.5}, {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}}  
};
```



`scores[0][1][0]` refers to the **multiple-choice** score for the **first student's second exam**, which is 9.0.

`scores[0][1][1]` refers to the **essay** score for the **first student's second exam**, which is **22.5**.

Next topics

Part II: Object-orientated Programming

Chapter 9:

- Objects and Classes