OKANAGAN

COSC 111

Computer Programming I

Chapter 8 Multidimensional Arrays

Dr. Abdallah Mohamed

Acknowledgement: The slides mainly rely on the textbook of Y. D. Liang titled “Introduction to Java Programming, 10th Ed.”, Pearson Edu. Inc.

Motivations

Data in a table or a matrix 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

Motivations

In Java, …

```
double[][] distances = {
  {0, 983, 787, 714, 1375, 967, 1087},
  {983, 0, 214, 1102, 1763, 1723, 1842},
  {787, 214, 0, 888, 1549, 1548, 1627},
  {714, 1102, 888, 0, 661, 781, 810},
  {1375, 1763, 1549, 661, 0, 1426, 1187},
  {967, 1723, 1548, 781, 1426, 0, 239},
  {1087, 1842, 1627, 810, 1187, 239, 0},
};
```

Declaring, Creating, and Initializing Arrays

Declaring & Creating 2D Arrays

An element in a two-dimensional array is accessed through a row and column index.

Declare and create a 2D array in two statements

```
int[][] x;
x = new int[5][10]; //5 rows, 10 columns
```

Declare and create a 2D array in ONE statement

```
int[][] x = new x[5][10];
```

Declare, create, and initialize a 2D array in ONE statement

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

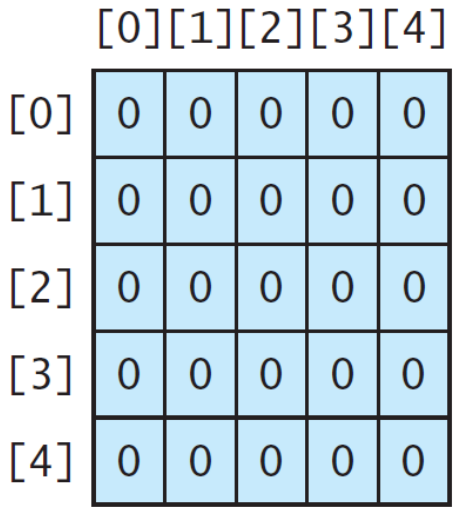
Same as

```
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;
```

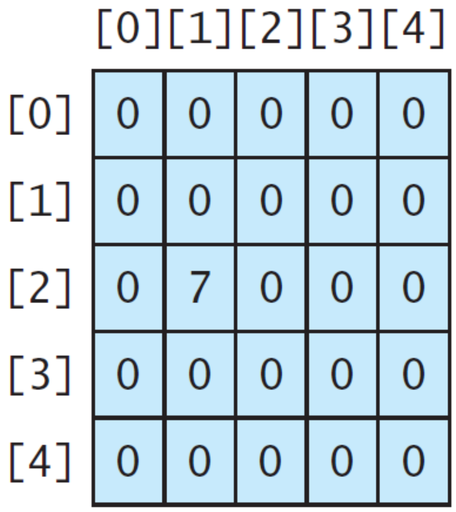
Declaring & Creating 2D Arrays, cont.

Examples:



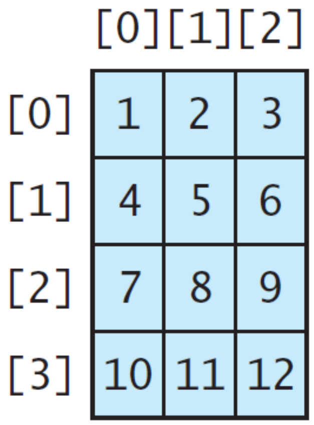
```java
matrix = new int[5][5];
```

(a)



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

(b)



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

(c)

Clicker Question

Which of the following correctly declares a 4x5 array?

A. `int[] arr = new int[4,5];`

B.
```
int[] arr;
arr = new int[4][5];
```

C. `int[][] arr = new int[4,5];`

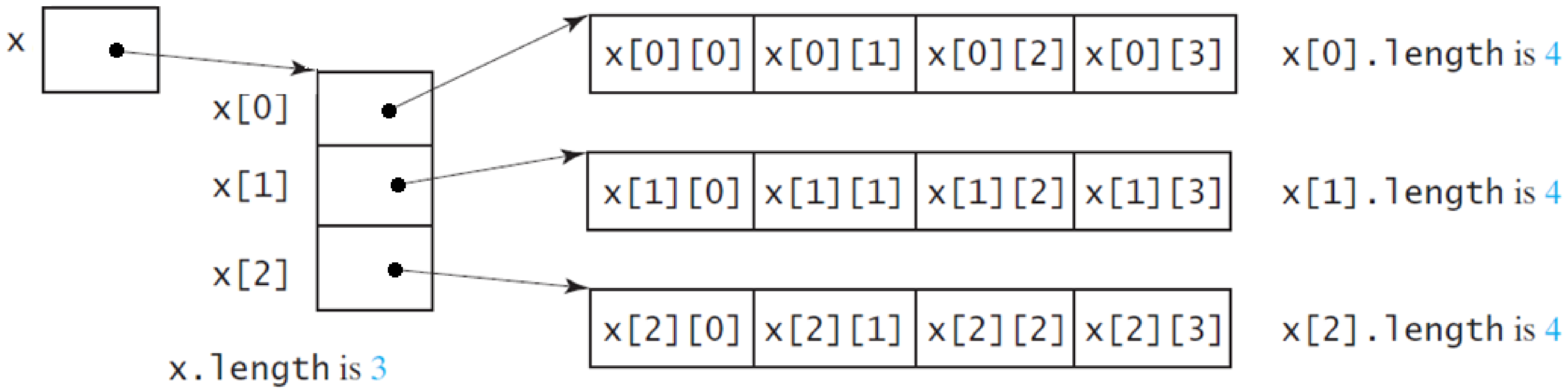
D.
```
int[][] arr;
arr = new int[4][5];
```

E. None of the above

How Java Implements 2D Arrays

In Java, a two-dimensional array is actually an array in which each element is a one-dimensional array.

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


x
x[0]
x[1]
x[2]
x.length is 3
x[0][0] x[0][1] x[0][2] x[0][3]
x[1][0] x[1][1] x[1][2] x[1][3]
x[2][0] x[2][1] x[2][2] x[2][3]
x[0].length is 4
x[1].length is 4
x[2].length is 4

Practice

What is the length of each of the following 2D arrays?

$$\begin{array}{c|c|c|c|c|c|}
 & [0] & [1] & [2] & [3] & [4] \\ \hline
[0] & 0 & 0 & 0 & 0 & 0 \\ \hline
[1] & 0 & 0 & 0 & 0 & 0 \\ \hline
[2] & 0 & 0 & 0 & 0 & 0 \\ \hline
[3] & 0 & 0 & 0 & 0 & 0 \\ \hline
[4] & 0 & 0 & 0 & 0 & 0 \\ \hline
\end{array}$$

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

$$\begin{array}{c|c|c|c|}
 & [0] & [1] & [2] \\ \hline
[0] & 1 & 2 & 3 \\ \hline
[1] & 4 & 5 & 6 \\ \hline
[2] & 7 & 8 & 9 \\ \hline
[3] & 10 & 11 & 12 \\ \hline
\end{array}$$

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

matrix.length returns 5 *(# of rows)*

matrix[0].length? 5 *(# of columns within the 1st row)*

array.length? 4

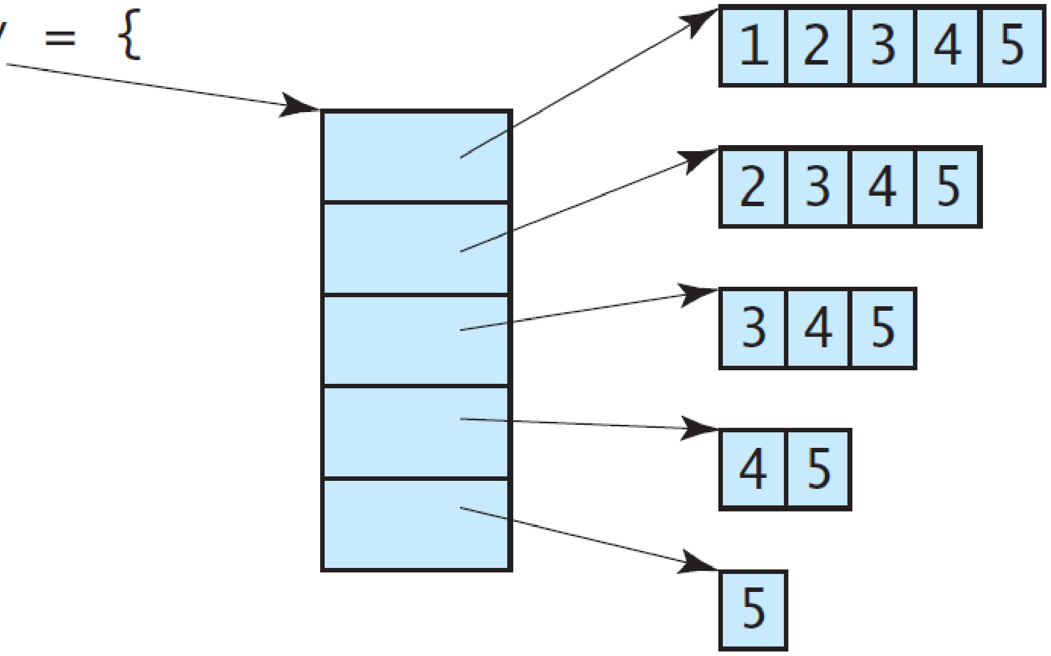
array[0].length? 3

Ragged Arrays

A ragged array is the one in which the **rows can have different lengths**..

- Remember that each row in a two-dimensional array is itself an array.
- **For example,**

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



What is the length of

triangleArray[0], → 5

triangleArray[1], → 4

… ,

triangleArray[4]. → 1

Ragged Arrays, cont'd

The following code declares a ragged array without initializing it

```
int[][] arr = new int[3][];
arr[0] = new int[3];
arr[1] = new int[10];
arr[2] = new int[5];
```

second dimension is omitted as it will be different for each row

Clicker Question

What is the value of i?

```
int[][] arr = new int[5][3];
int i = arr[0].length;
```

A. error

B. 0

C. 3

D. 5

E. 15

Clicker Question

What is the value of i?

```
int[][] arr = new int[5][3];
int i = arr.length;
```

A. error

B. 0

C. 3

D. 5

E. 15

Clicker Question

Assume `int[][] arr = new int[2][3];`

Which of the following assigns the correct number of rows to `rows` variable?

A. `rows = matrix.length;`

B. `rows = matrix[0].length;`

C. `rows = matrix[1].length;`

D. Either B or C

E. All of the above

Clicker Question

Assume **`int[][] arr = new int[2][3];`**

Which of the following assigns the correct number of columns to **`cols`** variable?

A. **`cols = matrix.length;`**

B. **`cols = matrix[0].length;`**

C. **`cols = matrix[1].length;`**

D. Either B or C

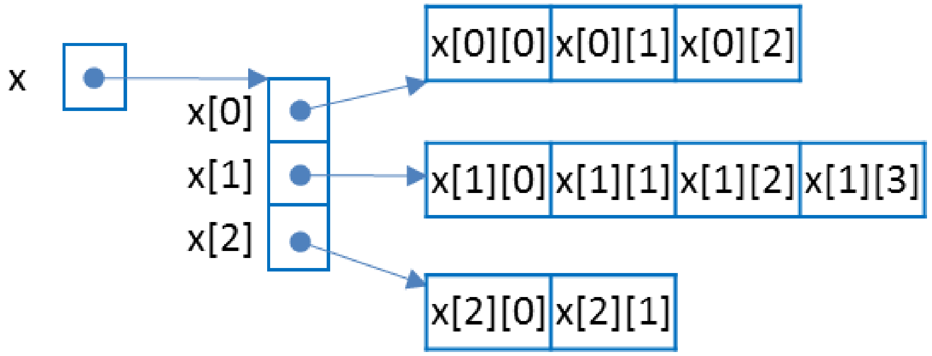
E. All of the above

Processing 2D Arrays

Processing 2D Array

Similarly to 1D arrays, you may use for loops for accessing 2D arrays. A common syntax to process **all elements evenly** is as follows:

```
for (int r = 0; r < x.length; r++){
   for (int c = 0; c < x[r].length; c++){
        //statements that are applied to all elements evenly
   }
}
```



Processing 2D Arrays: Examples

Initializing a 2D array with random values:

```
for (int r = 0; r < matrix.length; r++)
      for (int c = 0; c < matrix[r].length; c++)
           matrix[r][c] = (int)(Math.random() * 1000);
```

Initializing arrays with input values

```
Scanner input = new Scanner(System.in);
System.out.println("Enter " + matrix.length + " rows and " +
                                              matrix[0].length + " columns: ");
for (int r = 0; r < matrix.length; r++)
     for (int c = 0; c < matrix[r].length; c++)
          matrix[r][c] = input.nextInt();
```

Processing 2D Arrays: Examples, cont.

Printing arrays:

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

Finding the sum of all elements:

```
int total = 0;
for (int r = 0; r < matrix.length; r++)
    for (int c = 0; c < matrix[r].length; c++)
        total += matrix[r][c];
```

Processing 2D Arrays: Examples, cont.

Summing all elements by column:

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

Write code to:

- Find the row that has the largest sum.
- Find the smallest index of the largest element.
- To randomly shuffle array’s elements.

Practice

Objective: write a program that grades multiple-choice test.

Assume the following data is given and you are required to display the grade for each student.

Students' Answers

	0	1	2	3	4	5	6	7	8	9
Student 0	A	B	A	C	C	D	E	E	A	D
Student 1	D	B	A	B	C	A	E	E	A	D
Student 2	E	D	D	A	C	B	E	E	A	D
Student 3	C	B	A	E	D	C	E	E	A	D
Student 4	A	B	D	C	C	D	E	E	A	D
Student 5	B	B	E	C	C	D	E	E	A	D
Student 6	B	B	A	C	C	D	E	E	A	D
Student 7	E	B	E	C	C	D	E	E	A	D

Key to the Questions:

	0	1	2	3	4	5	6	7	8	9
Key	D	B	D	C	C	D	A	E	A	D

Practice, cont.

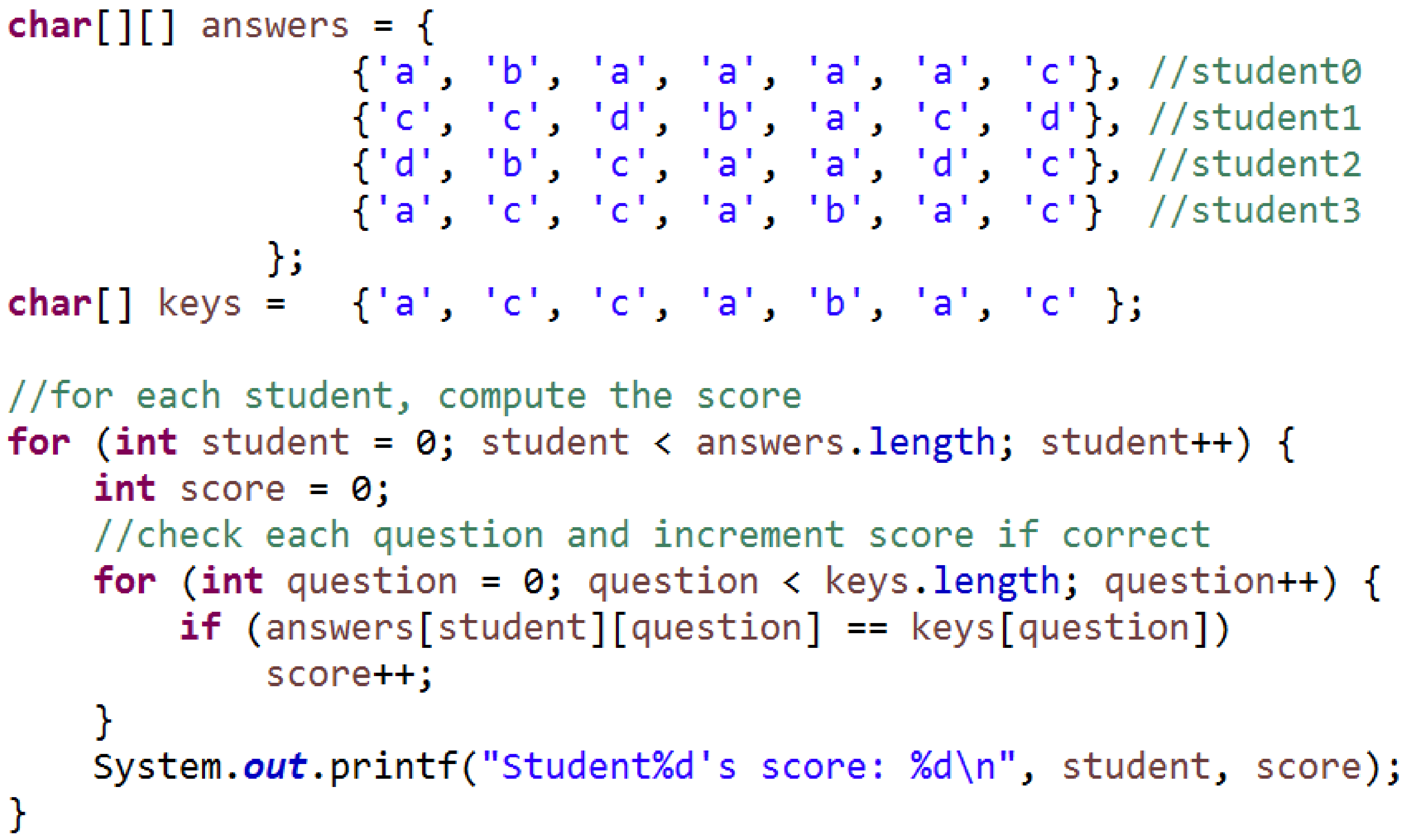
Algorithm:

```
1. Store data for students' answer and key in arrays.

2. For each student (rows),
    a. initialize a counter for counting student's correct answers.
    b. For each question (columns),
        • if student's answer is correct, increment the counter
    c. display the count of correct answers for that student.

3. End the program
```

Practice, cont.



```
char[][] answers = {
                {'a', 'b', 'a', 'a', 'a', 'a', 'c'}, //student0
                {'c', 'c', 'd', 'b', 'a', 'c', 'd'}, //student1
                {'d', 'b', 'c', 'a', 'a', 'd', 'c'}, //student2
                {'a', 'c', 'c', 'a', 'b', 'a', 'c'}  //student3
            };
char[] keys =   {'a', 'c', 'c', 'a', 'b', 'a', 'c' };

//for each student, compute the score
for (int student = 0; student < answers.length; student++) {
    int score = 0;
    //check each question and increment score if correct
    for (int question = 0; question < keys.length; question++) {
        if (answers[student][question] == keys[question])
            score++;
    }
    System.out.printf("Student%d's score: %d\n", student, score);
}
```

Special cases

Using one for loop

Code to read the price and quantity of several items.

	price	quantity
Item 0		
Item 1		
Item 2		
Item 3		

```
int[][] table = new int[4][2];          //4 rows, 2 cols
for (int item = 0; item < 2; item++) { //for each item record
      System.out.printf("Enter the price of item#%d: ",item);
      table[item][0] = input.nextInt();
      System.out.printf("Enter the number of items: ");
      table[item][1] = input.nextInt();
}
```

Two for loops that don’t follow the standard format

Code to read the ID and 3 grades for several students.

	ID	Grade 1	Grade 2	Grade 3
Student 0				
Student 1				
Student 2				

```
int[][] grades = new int[3][4];                // 3 students, 4 entries
for (int student = 0; student < grades.length; student++) {
       System.out.printf("Enter the ID of Student#%d: ", student+1);
       grades[student][0] = input.nextInt();
       for (int assignment = 1; assignment <= 3; assignment++) {
              System.out.printf("Enter grade for A%d: ", assignment);
              grades[student][assignment] = input.nextInt();
       }
}
```

2-D Arrays to/from Methods

Multidimensional Arrays & Methods

Same rules studied before (in Chapter 7) apply here!

- Passing 2-D Arrays to Methods:
  - When passing a 2-D array to a method, the reference of the array is passed to the method.
  - You have to have method parameters declared of the same type and dimension of the arguments.

- Returning 2-D Arrays to Methods:
  - When a method returns an array **the reference of the array is returned**.

Clicker Question

What is the value of **`arr`** array?

```
public static void main(String[] args) {
   int[][] arr = { {1,2,3},
                   {4,5,6} };
   arr = zeros(2,2);
}
public static int[][] zeros(int n, int m) {
   return new int[n][m];
}
```

A. 1 2 3
   4 5 6

B. 0 0 0
   0 0 0

C. 1 2
   4 5

D. 0 0
   0 0

Optional Readings Multidimensional Arrays

Multidimensional Arrays

If you need to represent n-dimensional data structures, you can create nD arrays.

- A 2D array is an array of 1D arrays
- A 3D array is an array of 2D arrays.
- …

Example:

- `int[]     x = new int[10];  //1D array`
- `int[][]   y = new int[5][12];  //2D array`
- `int[][][] z = new int[2][7][3];  //3D array`

This defines

- x array of 10 integers, y of 5 by 12 matrix of integers, and z of 2 by 7 by 3 array of integers.

You can then access these elements, for example:

- `y[2][3] = x[0] + z[2][1][5];`

Example: Calculating Total Scores

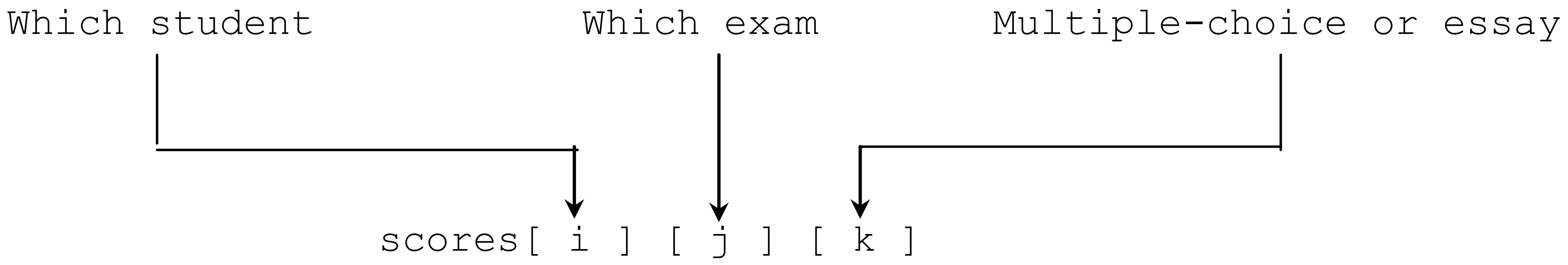
Write a program that calculates **the total score for students** in a class. Suppose the scores are stored in a 3D array named scores.

- The first index in scores refers to a student,
- the second refers to an exam, and
- the third refers to the part of the exam.

Suppose there are 7 students, 5 exams, and each exam has two parts--the multiple-choice part and the programming part.

- e.g., for the i's student on the j's exam: scores[i][j][0] represents the score on the multiple-choice part, and scores[i][j][1] represents the score on the programming part.

Your program displays the total score for each student.



Problem: Calculating Total Scores, cont.

```
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} }
};
```

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**.

```
double[][][] scores={
  { {7.5, 20.5},
    {9.0, 22.5},
    {15,  33.5},
    {13,  21.5},
    {15,  12.5} },
  { {4.5, 21.5},
    {9.0, 22.5},
    {15, 34.5},
    {12, 20.5},
    {14,   9.5} },
…
```

Problem: Calculating Total Scores, cont.

Algorithm:

- Store data for students' answer and key in arrays.

```
1. for each student,
     a. initialize a variable, totalScore, for summing the student's score.
     b. For each exam,
          For each question,
               add the question's grade to totalScore
     c. display the count of correct answers for that student.

2. End the program
```

Problem: Calculating Total Scores, cont.

```
public class TotalScore {
    public static void main(String args[]) {
        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}},
          {{1.5, 29.5}, {6.4, 22.5}, {14, 30.5}, {10, 30.5}, {16, 6.0}}};

        // Calculate and display total score for each student
        for (int i = 0; i < scores.length; i++) {
            double totalScore = 0;
            for (int j = 0; j < scores[i].length; j++)
                for (int k = 0; k < scores[i][j].length; k++)
                    totalScore += scores[i][j][k];

            System.out.println("Student " + i + "'s score is " + totalScore);
        }
    }
}
```