CENTRE FOR
CONTINUING EDUCATION

DESKTOP APPLICATION DEVELOPMENT WITH JAVA – CEJV569

AMIN RANJ BAR

Your Instructor

Amin Ranj Bar
- PhD at Computer Science from McGill University
- Faculty of Champlain college, Computer Science and Software Engineering Department of Concordia University

My expertise:
- Computer Networks
- Online Social Networks
- Network Security
- Programming languages

E-mail: cejv.ranjbar@gmail.com

Java I

- DESKTOP APPLICATION DEVELOPMENT WITH JAVA
- Day/Time: Mon 18:00 to 22:30 Room: FB107
- **2017-04-09 to 2017-06-11**

Course Prerequisite

- CEJV416

Course Description

- This course will continue the work done in CEJV 416 with an emphasis on developing desktop applications using the JavaFX and JDBC frameworks.
- Students will learn how to develop systems that are composed of presentation, business (domain) and persistence layers.
- The data structures that make up the Java Collections Framework will be explored and then applied to a range of problems.
- Additional topics in this course will include concurrent programming using threads, file access using NIO and the development of Create, Read, Update and Delete (CRUD) applications using JDBC.
- Upon completion of this course, the student will have acquired the necessary skills to begin developing real world software solutions.

Course Objectives

- Utilize industry standards in program design, coding and testing
- Apply patterns when structuring code
- Understand the purpose of and then use in code data structures such as Stacks, Queues, Deques and Maps
- Write object oriented code for accessing relational databases
- Implement basic threads for concurrent processing
- Read and write text files and properties files
- Develop multi panel GUI layouts utilizing a range of JavaFX Components
- Employ the techniques of internationalization in a program

Course Methodology

- Lectures
- In-class exercises
- Lab assignments

Learning Resources:

- Class notes, presentations and sample code are available on Moodle
- Recommended books:
  - Introduction to Java Programming, 11th Edition, Y. Daniel Liang, ISBN-13: 978-0134611037

Course Content

	Topics
1	Overview of Java Multidimensional arrays Processing two-dimensional arrays Passing multidimensional arrays to methods
2	Object Oriented Thinking Array of Objects Immutable objects and classes Abstraction Encapsulation Class relationships Processing primitive data type values as objects Examining the Object class Inherited methods of Object

Course Content

	Topics
3	Coding to the interface Decoupling code Comparable and cloneable interfaces Polymorphism

4	Persisting data to text and binary files
	  NIO File processing
	  File class
	Reading Data From the web

Course Content

	Topics
5	Building GUI programs with JavaFX Catalog the available components High level and low level event handling Using the Gluon Scenebuilder editor to create multi panel interfaces Developing software for multiple languages Internationalization
6	Event Driven Programming Animation
7	Persisting data to a relational database JDBC coding Create, Update, Read & Delete coding Testing code JUnit

Course Content

	Topics
8	Employing data structures from the Java Collections Framework Interfaces Implementations Algorithms Stacks, Queues, Deques and Maps Applying software patterns such as Abstract Factory Decorator Facade Factory method Singleton Proxy Adapter Iterator MVC
9	Employing concurrent programming Threading Tasks Synchronization Locks

Communication outside course hours

- If you have any questions please use my email address of cejv.ranjbar@gmail.com
- I will do my best to respond within 48 hours

Assessment/Evaluation:

Assignments	70%
Final Exam	30%

A minimum grade of 60% is required to successfully complete this course.

Software:

The IDE for this course is NetBeans 8 and is available for free from:

http://netbeans.org

Download the Java EE bundle.

The version of Java will be 1.9 JDK available for free from:

http://www.oracle.com/technetwork/java/index.html

Download the most recent versions

Scene Builder 9 will be employed for creating GUI user interfaces by drag and drop

http://gluonhq.com/labs/scene-builder/

Download the most recent versions

This environment can be setup on Windows, Mac, and Linux

Assignment Submissions:

- All submissions must be in electronic form.
- The NetBeans project folder and its contents must be compressed into a zip file and submitted on Moodle.

Rights and Responsibilities:

Plagiarism
The most common offense under the Academic Code of Conduct is plagiarism which the Code defines as “the presentation of the work of another person as one’s own or without proper acknowledgement.”

This could be material copied word for word from books, journals, internet sites, professor’s course notes, etc. It could be material that is paraphrased but closely resembles the original source. It could be the work of a fellow student, for example, an answer on a quiz, data for a lab report, a paper or assignment completed by another student. It might be a paper purchased through one of the many available sources. Plagiarism does not refer to words alone - it can also refer to copying images, graphs, tables, and ideas. “Presentation” is not limited to written work. It also includes oral presentations, computer assignments and artistic works. Finally, if you translate the work of another person into French or English and do not cite the source, this is also plagiarism.

Rights and Responsibilities:

Plagiarism

In Simple Words:
Do not copy, paraphrase or translate anything from anywhere without saying from where you obtained it!
Source: http://provost.concordia.ca/academicintegrity/plagiarism

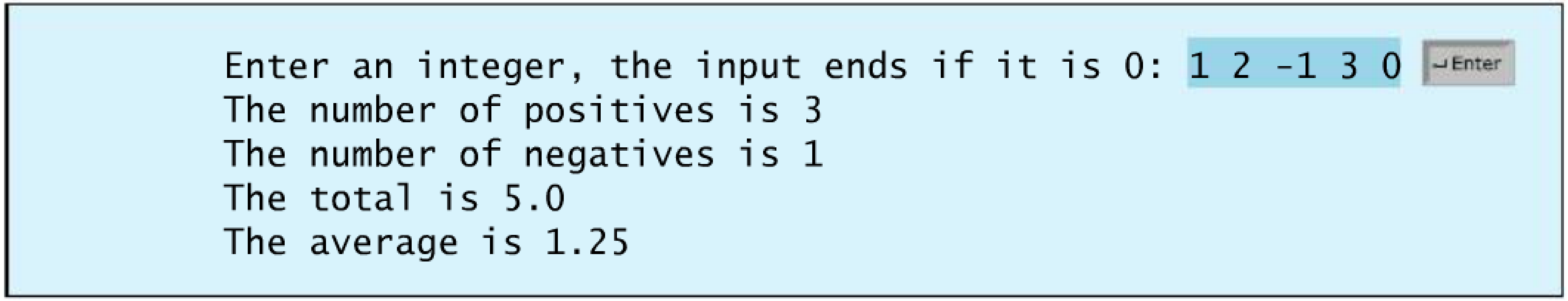
CONCORDIA SECURITY

- Add 514-848-3717 into their cellphone
- This will permit quick contact when needed with Concordia Security, who are familiar with the location of buildings at Concordia and can direct emergency services to the right location.
- https://www.youtube.com/watch?v=15bomtL0XMA

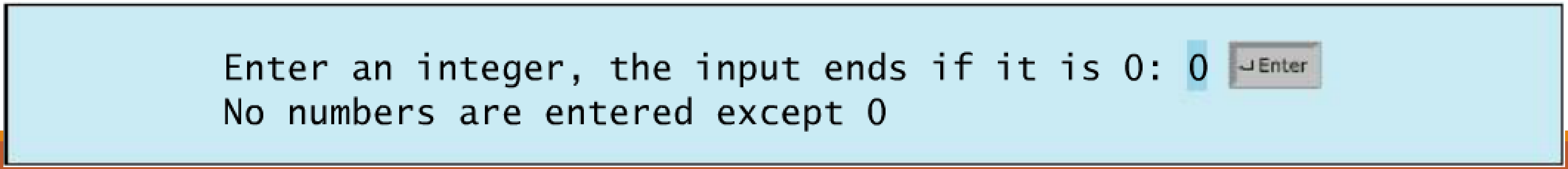
Let’s Get Started

Exercise - 1

(*Count positive and negative numbers and compute the average of numbers*) Write a program that reads an unspecified number of integers, determines how many positive and negative values have been read, and computes the total and average of the input values (not counting zeros). Your program ends with the input **0**. Display the average as a floating-point number. Here is a sample run:



```
Enter an integer, the input ends if it is 0: 1 2 -1 3 0
The number of positives is 3
The number of negatives is 1
The total is 5.0
The average is 1.25
```



```
Enter an integer, the input ends if it is 0: 0
No numbers are entered except 0
```

Exercise - 2

(*Sum the digits in an integer*) Write a method that computes the sum of the digits in an integer. Use the following method header:

```
public static int sumDigits(long n)
```

For example, `sumDigits(234)` returns `9` (2 + 3 + 4). (*Hint*: Use the `%` operator to extract digits, and the `/` operator to remove the extracted digit. For instance, to extract 4 from 234, use `234 % 10` (= 4). To remove 4 from 234, use `234 / 10` (= 23). Use a loop to repeatedly extract and remove the digit until all the digits are extracted. Write a test program that prompts the user to enter an integer and displays the sum of all its digits.

Exercise - 3

(*Eliminate duplicates*) Write a method that returns a new array by eliminating the duplicate values in the array using the following method header:

```java
public static int[] eliminateDuplicates(int[] list)
```

Write a test program that reads in ten integers, invokes the method, and displays the result. Here is the sample run of the program:

```
Enter ten numbers: 1 2 3 2 1 6 3 4 5 2 ↵Enter
The distinct numbers are: 1 2 3 6 4 5
```

CENTRE FOR
CONTINUING EDUCATION

Multidimensional Arrays

Motivations

- Thus far, you have used one-dimensional arrays to model linear collections of elements.
- You can use a two-dimensional array to represent a matrix or a table.
  - For example, the following table that describes the distances between the cities can be represented using a two-dimensional array.

Motivations

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

```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},
};
```

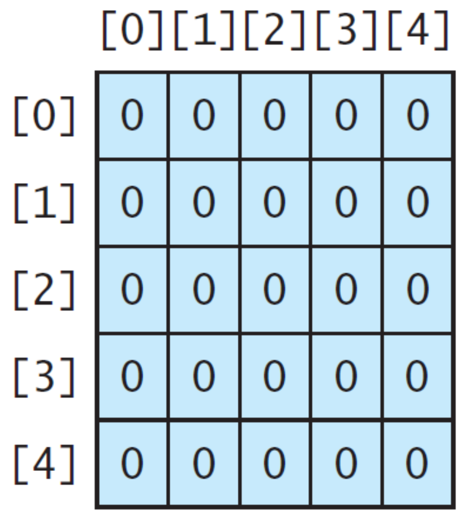
Declare/Create Two-dimensional Arrays

```
// Declare array ref var
dataType[][] refVar;

// Create array and assign its reference to variable
refVar = new dataType[10][10];

// Combine declaration and creation in one statement
dataType[][] refVar = new dataType[10][10];

// Alternative syntax
dataType refVar[][] = new dataType[10][10];
```

Declaring Variables of Two-dimensional Arrays and Creating Two-dimensional Arrays

```
int[][] matrix = new int[10][10];
 or
int matrix[][] = new int[10][10];
matrix[0][0] = 3;

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

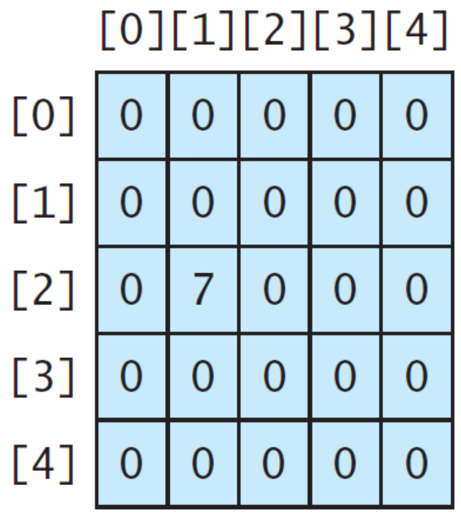
Two-dimensional Array Illustration



	[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];
```

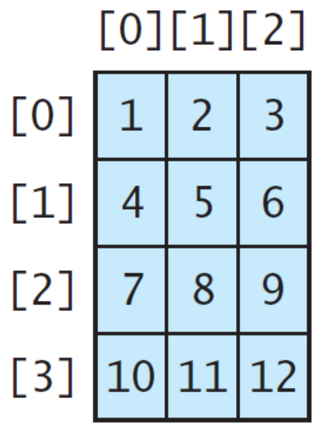
(a)



	[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)

matrix.length? 5

matrix[0].length? 5

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,

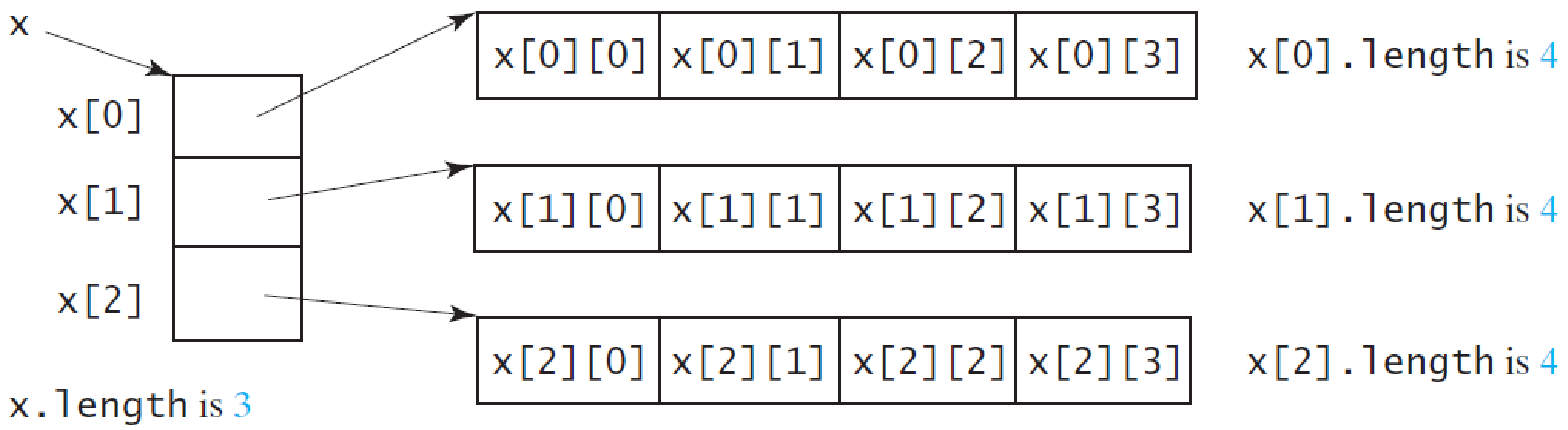
```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

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];
```


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

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

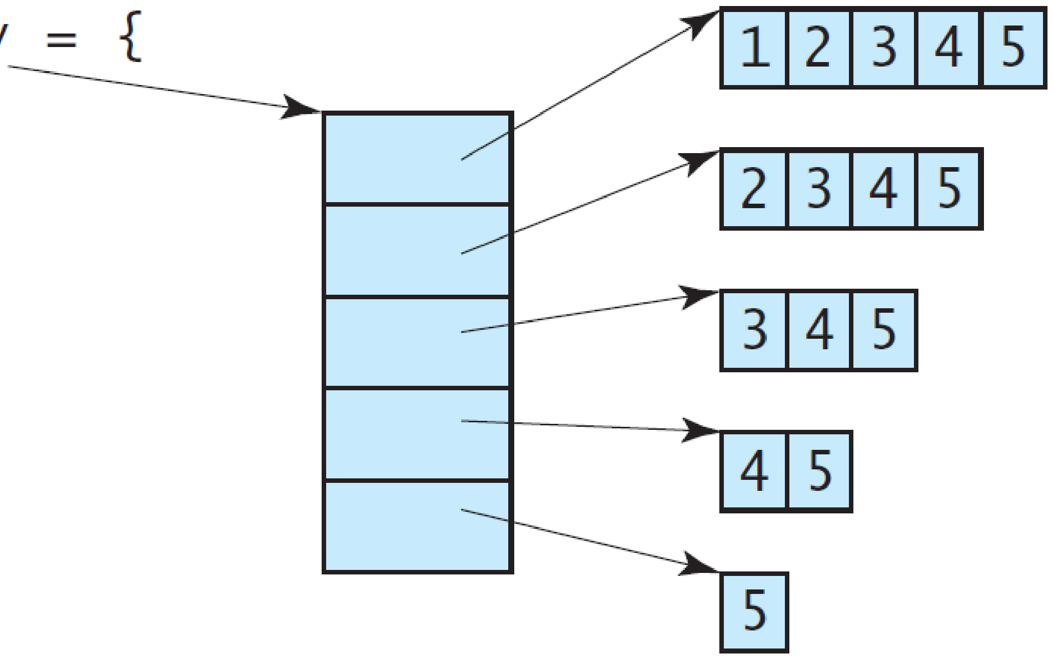
Each row in a two-dimensional array is itself an array. So, the rows can have different lengths. Such an array is known as *a ragged array*. For example,

```
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}
};
```



Example - Passing Two-Dimensional Arrays to Methods

- We want to write a program to get a 2-dimensional array from the user and then find the sum of all its elements.
- Finding the sum and getting the array should be separate methods.

Exercise – 5
Processing Two-Dimensional Arrays

1. Getting the number of columns and arrows from the user
2. Initializing the array with integer random values between 0 and 100.
3. Printing the array
4. Summing all elements
5. Summing all elements by column
6. Which row has the largest sum
7. Finding the smallest index of the largest element

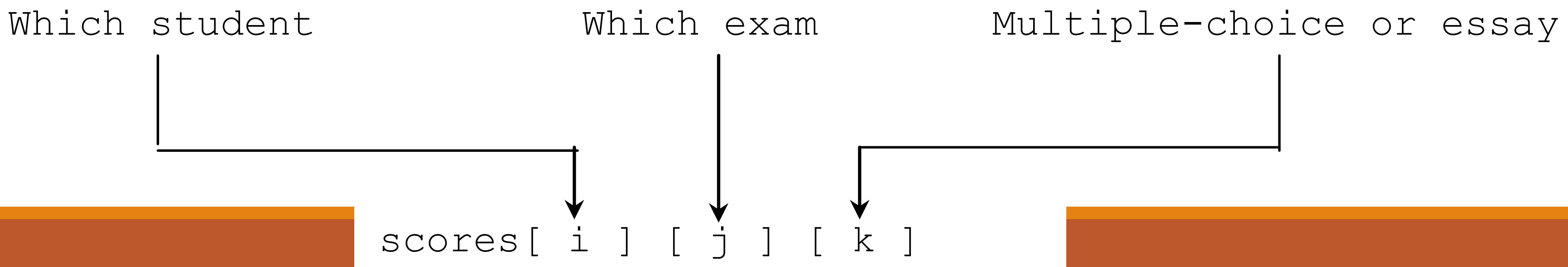
Multidimensional Arrays

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 >= 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}}};
```



Exercise 6
Grading Multiple-Choice Test

Students' answer

	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

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

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