Advanced Programming Language

MSc. Nguyen Cao Dat

dat@hcmut.edu.vn

Module III

DATA STRUCTURES IN JAVA

Content

- **Introduction**
- **Arrays**
- **Sample problem**
- **Two-Dimensional Arrays**
- **Case study**

Introduction

- "Get your data structures correct first, and the rest of the program will write itself."

  - *David Jones*

Introduction

- A data structure is a collection of data organized in some fashion.
- A data structure not only stores data, but also supports the operations for manipulating data in the structure.
- Different types of data structures are optimized for certain types of operations
  - www.nist.gov/dads/
  - en.wikipedia.org/wiki/List_of_data_structures

Core Operations

- **Data Structures will have 3 core operations**
  - a way to add things
  - a way to remove things
  - a way to access things
- **Details of these operations depend on the data structure**
  - Example: List, add at the end, access by location, remove by location
- **More operations added depending on what data structure is designed to do**

Data Structures in Programming Languages

- **Modern programming languages usually have a library of data structures**
  - Java collections framework
  - C++ standard template library
  - .Net framework
  - …

Questions?

Content

- Introduction
- Arrays
- Sample problem
- Two-Dimensional Arrays
- Case study

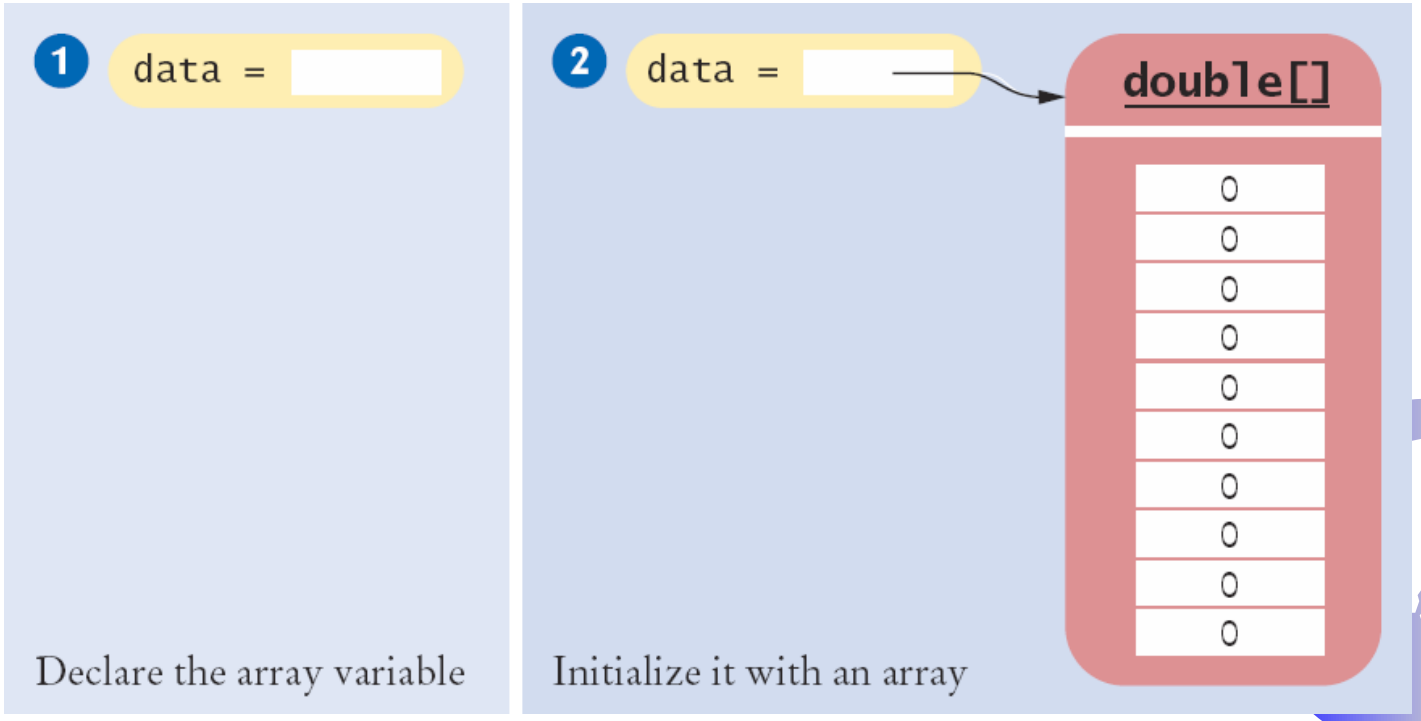
Arrays

- Arrays are one of the most powerful programming tools available
- Provides the programmer with a way of organizing a collection of homogeneous data items into a single data structure
- An array is a data structure which is made up of a number of variables all of which have the same data type

Declaring Arrays

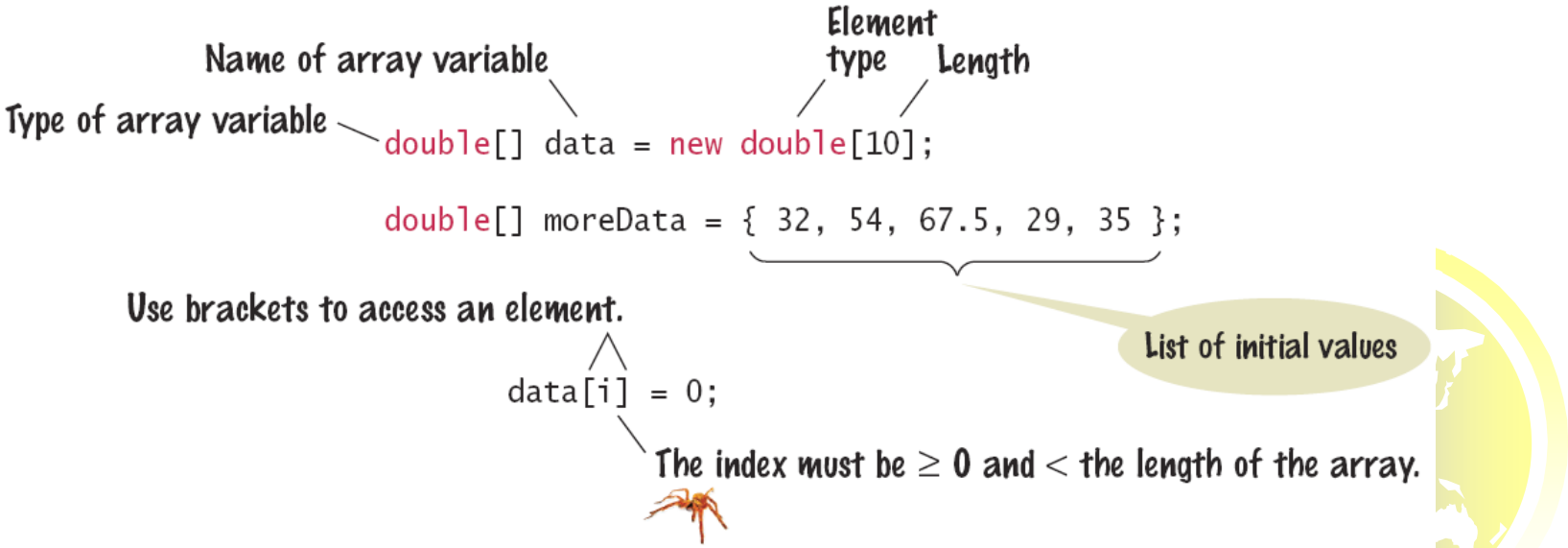
- **Declaring an array is a two step process**

```
1) double[] data;     // declare array variable
2) data = new double[10];  // initialize size
```



Syntax: Arrays

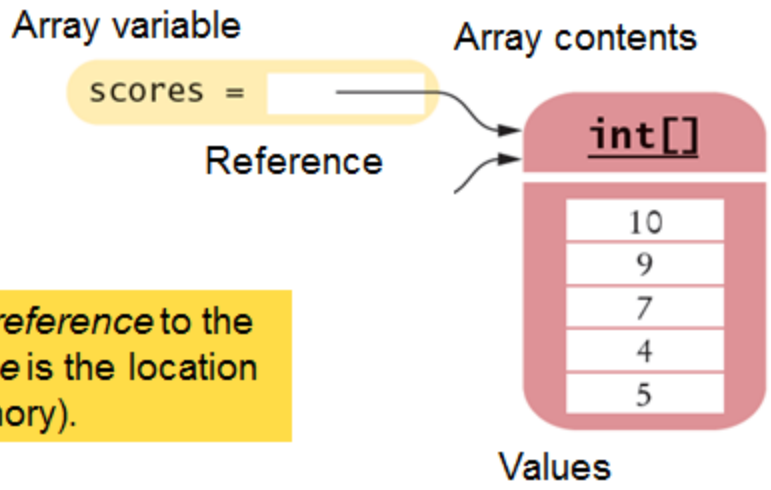
- **To declare an array, specify the:**
  - Array variable name
  - Element data type
  - Length (number of elements)



Array References

- **Make sure you understand the difference between:**
  - Array variable: The "symbolic name" (reference value) to the array
  - Array contents: Memory where the values are stored

```
int[] scores = { 10, 9, 7, 4, 5 };
```



An array variable contains a *reference* to the array contents. The *reference* is the location of the array contents (in memory).

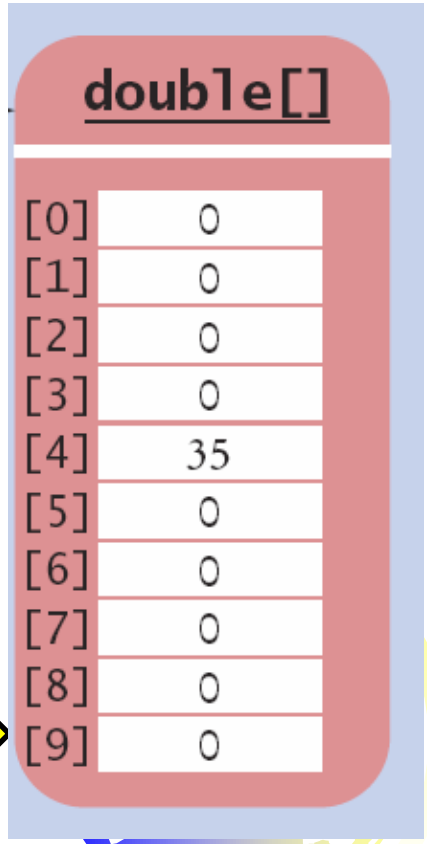
Array Index Numbers

- **Array index numbers start at 0**
  - The rest are positive integers
- **A 10 element array has indexes 0 through 9**
  - There is no element 10!

The first element is at index 0:

```
public static void main(String[] args)
{
  double data[];
  data = new double[10];
}
```

The last element is at index 9:



Pseudocode: Writing Out the Contents of an Array

- **Writing out the elements can be represented using a DO loop (for all elements in the array)**

```
DO index = 1 to number_of_elements
        Print array(index)
ENDDO
```

Java: Writing Out the Contents of an Array

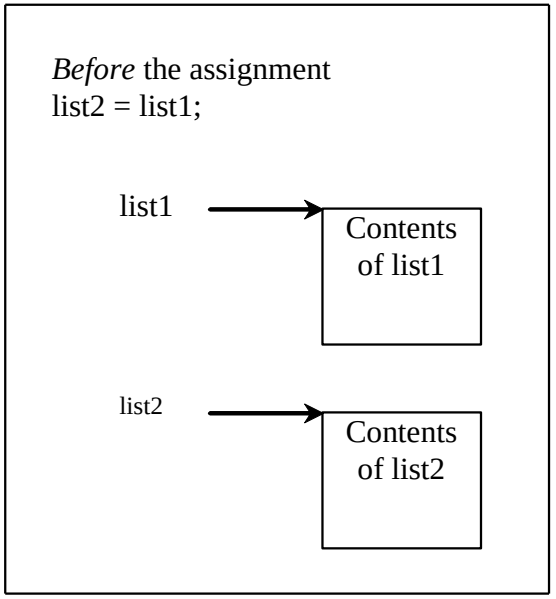
- **A DO loop in pseudocode is a for loop in Java**

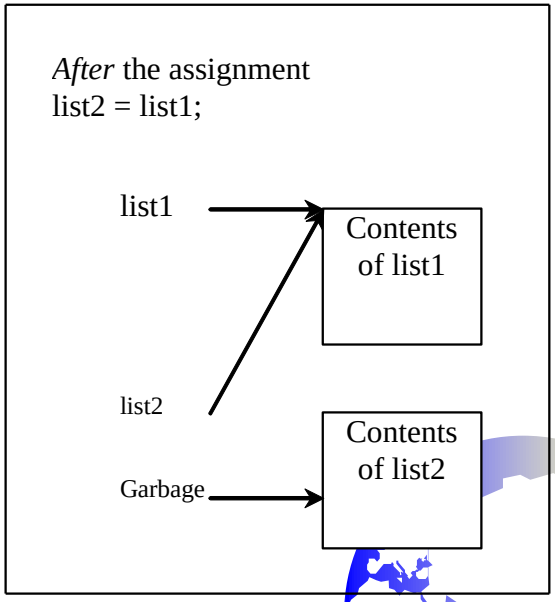
```
for (int x = 0; x < arrayName.length; x++) {
   JOptionPane.showMessageDialog(null, arrayName[x]);
}
```

Copying Arrays

Often, in a program, you need to duplicate an array or a part of an array. In such cases you could attempt to use the assignment statement (=), as follows:

list2 = list1;





Java: Copying Arrays

- **Using a loop:**

```
int[] sourceArray = {2, 3, 1, 5, 10};
int[] targetArray = new
  int[sourceArray.length];

for (int i = 0; i < sourceArrays.length; i++)
   targetArray[i] = sourceArray[i];
```

Java: The `arraycopy` Utility

```
arraycopy(sourceArray, src_pos,
  targetArray, tar_pos, length);
```

- **Example:**

```
System.arraycopy(sourceArray, 0,
  targetArray, 0, sourceArray.length);
```

Passing Arrays to Methods

```
public static void printArray(int[] array) {
  for (int i = 0; i < array.length; i++) {
    System.out.print(array[i] + " ");
  }
}
```

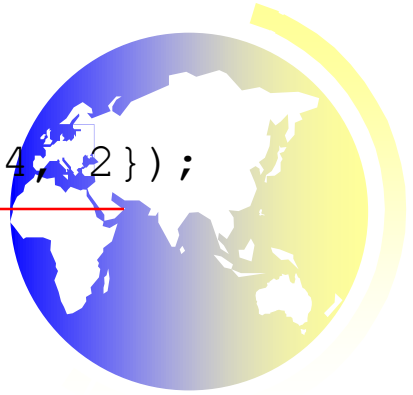
Invoke the method

```
int[] list = {3, 1, 2, 6, 4, 2};
printArray(list);
```

Invoke the method

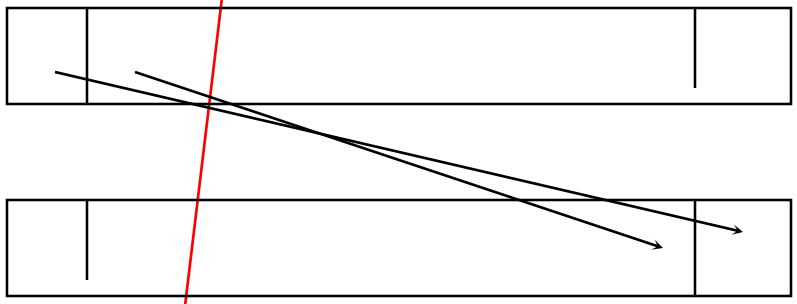
```
printArray(new int[]{3, 1, 2, 6, 4, 2});
```

Anonymous array



Returning an Array from a Method

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```



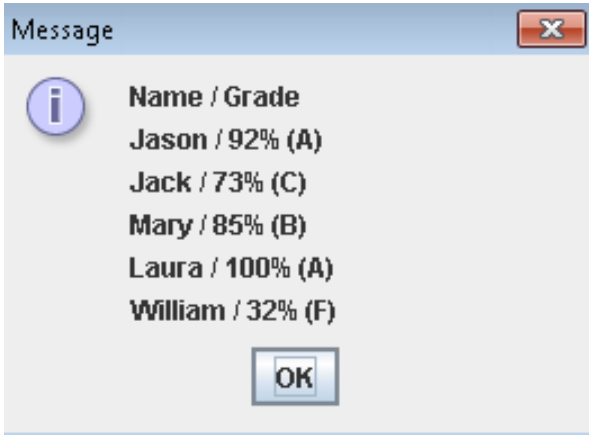

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

Content

- Introduction
- Arrays
- Sample problem
- Two-Dimensional Arrays
- Case study

Problem Scenario: Maintaining a Gradebook

- You have been tasked to maintain a "database" of up to 100 student names and course percentages for a graduate class. The number of students will be known. Once the information is input by the user, determine the letter grade for each student based on minimum criteria (e.g. 90-A, 80-B, 70-C, 0-F). Display a report of all students entered and their grades



What do you do now?

Remember the Steps!

- Step 1: Define the problem
- Step 2: Group the activities
- Step 3: Create a hierarchy chart/flowchart EACH method
- Step 4: Establish the mainline logic
- Step 5: Create the solution algorithm (pseudocode for EACH method)
- Step 6: Desk check the algorithm
- Step 7: Create the Program
- Step 8: Test the Program
- Step 9: Document the Program

Questions?

Content

- Introduction
- Arrays
- Sample problem
- Two-Dimensional Arrays
- Case study

Two-Dimensional Arrays

- **Often times you may have to deal with information within a two-dimensional layout**
  - Tax Schedules
  - Gradebook
- **Layout mimics a spreadsheet**
  - Rows and columns
  - Also known as a "matrix"
  - Sometimes called an "array of arrays"

Declaring Two-Dimensional Arrays

- Declaring a two-dimensional array is the same as a one-dimensional array but with two 'pairs' of square braces

Gold	Silver	Bronze
1	0	1
1	1	0
0	0	1
1	0	0
0	1	1
0	1	1
1	1	0

```
final int COUNTRIES = 7;
final int MEDALS = 3;
int[][] counts = new int[COUNTRIES][MEDALS];
```

Declaring Two-Dimensional Arrays

- The array can also be initialized with values

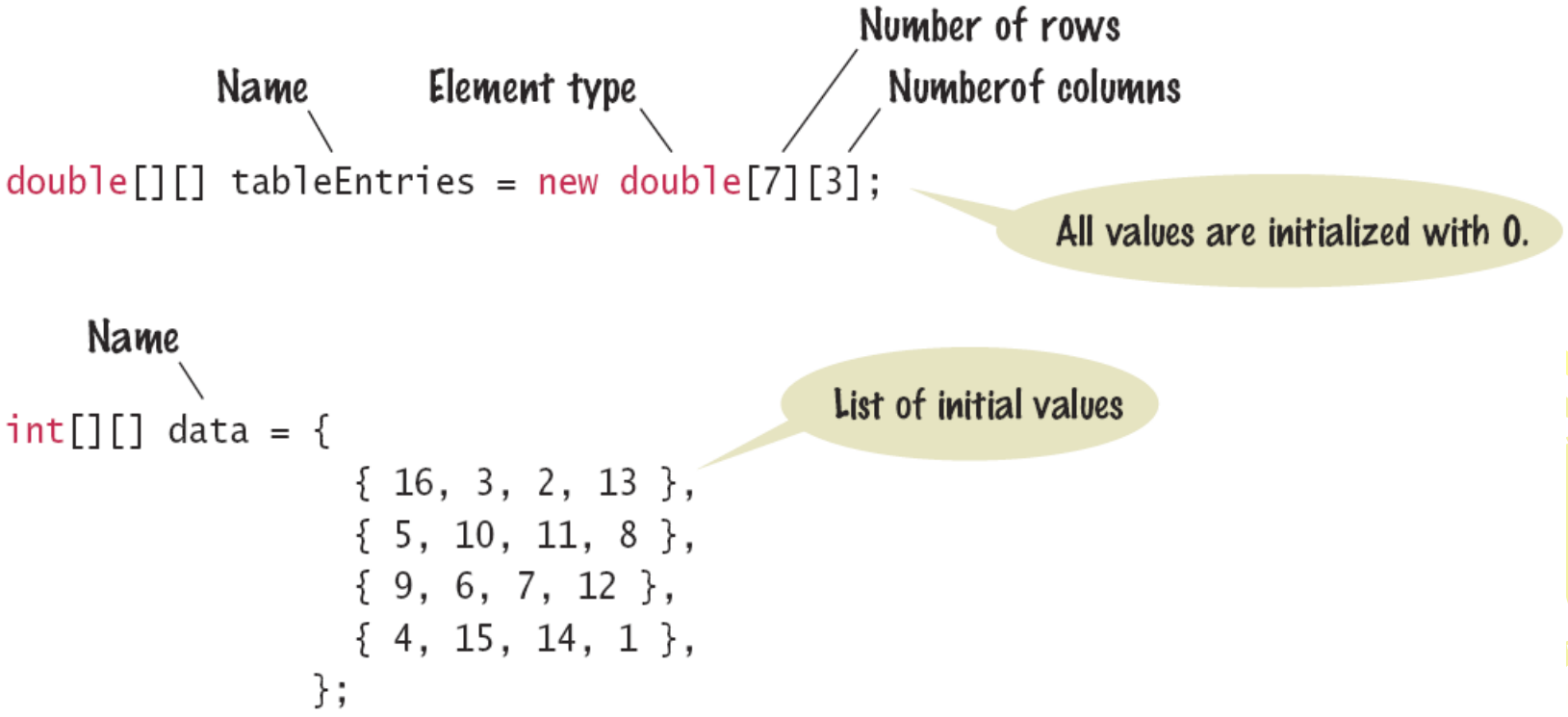
```
final int COUNTRIES = 7;
final int MEDALS = 3;
int[][] counts =
{
  { 1, 0, 1 },
  { 1, 1, 0 },
  { 0, 0, 1 },
  { 1, 0, 0 },
  { 0, 1, 1 },
  { 0, 1, 1 },
  { 1, 1, 0 }
};
```

Gold	Silver	Bronze
1	0	1
1	1	0
0	0	1
1	0	0
0	1	1
0	1	1
1	1	0

Note the use of two ‘levels’ of curly braces. Each row has braces with commas separating them.

Syntax: Two-Dimensional Arrays

- **The name of the array continues to be a reference to the array contents**
  - Use new or fully initialize the array

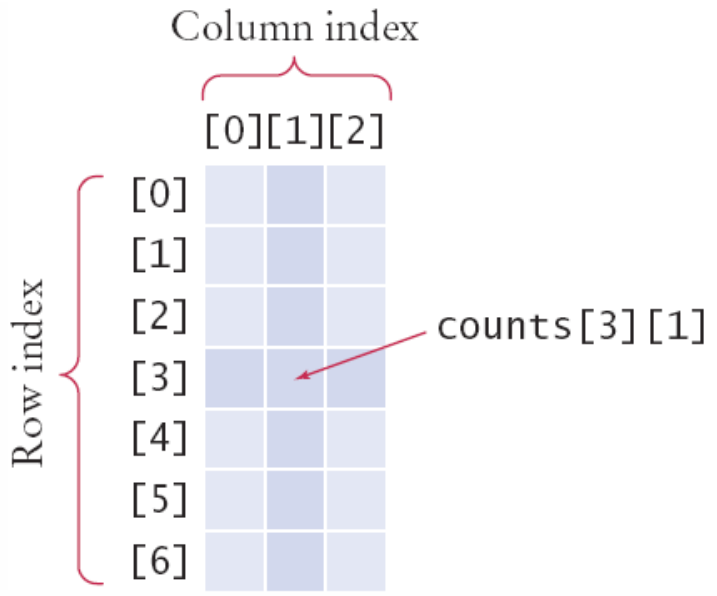


Accessing Elements

- **Use two index values**

Row then Column

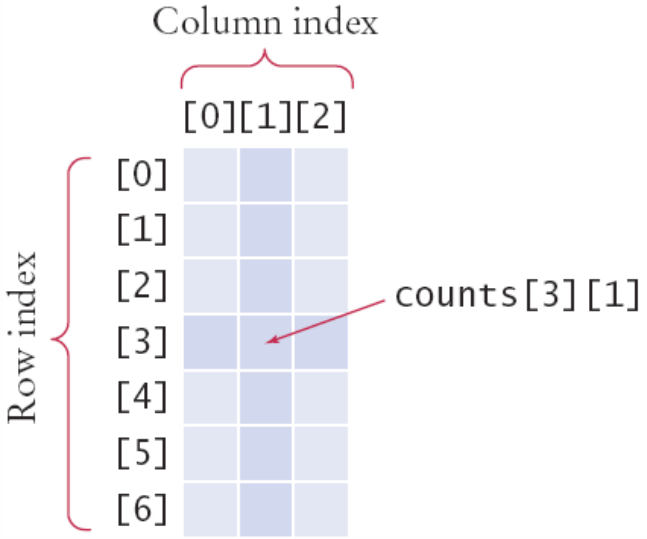
```
int value = counts[3][1];
```

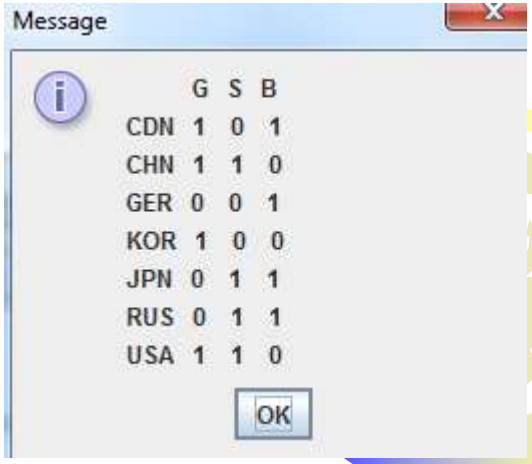


Printing a Two-Dimensional Array

- Create a String to store all values
- Use nested for loops to get all values
- Outer row (i), inner column (j)

```
// Calculate the rows and columns in 2D array
final int COUNTRIES = medalCounts.length;
final int MEDALS = medalCounts[0].length;

/***** Print medal table *****/
String medalCount = "";

// Add medal types
medalCount += "            G    S    B\n";

// Process the "i"th row
for (int i = 0; i < COUNTRIES; i++) {

   // Add country name
   medalCount += countries[i] + "    ";

   // Process the "j"th column in the "i"th row
   for (int j = 0; j < MEDALS; j++) {
      medalCount += medalCounts[i][j] + "    ";
   }
   // Start a new line at the end of the row
   medalCount += "\n";
}
JOptionPane.showMessageDialog(null, medalCount);
```

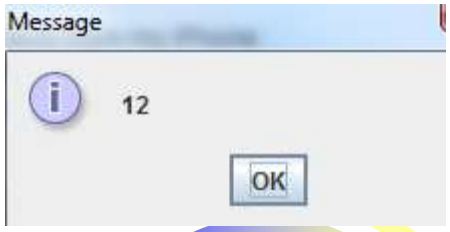




Adding All Values in Two-Dimensional Array

- **To add all of the values in the two-dimensional array**
  - Loop through all of the rows
  - Loop through all of the columns
    - Get the data value at each row/column intersection

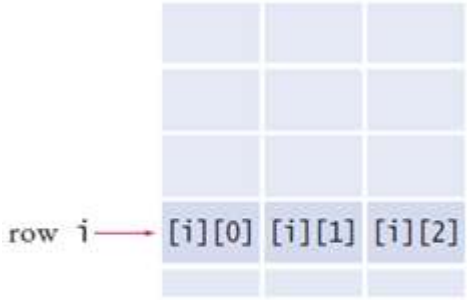
```
// Calculate the rows and columns in 2D array
final int COUNTRIES = medalCounts.length;
final int MEDALS = medalCounts[0].length;

/***** Get the total number of medals *****/
int total = 0;

for (int i = 0; i < COUNTRIES; i++) {
   for (int j = 0; j < MEDALS; j++) {
      total += medalCounts[i][j];
   }
}
JOptionPane.showMessageDialog(null, total);
```



Add Row Totals (Columns of Each Row) in Two-Dimensional Array

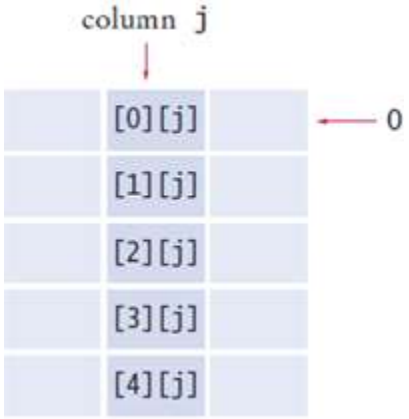
- Loop through the rows, then columns and keep a total of the row values

```
// Calculate the rows and columns in 2D array
final int COUNTRIES = medalCounts.length;
final int MEDALS = medalCounts[0].length;

/***** Print medal table *****/
String medalCount = "";

// Add medal types
medalCount += "           G   S   B     T\n";

// Process the "i"th row
for (int i = 0; i < COUNTRIES; i++) {

   // Add country name
   medalCount += countries[i] + "   ";

   // For each row, reset the medalTotal
   int medalTotal = 0;

   // Process the "j"th column in the "i"th row
   for (int j = 0; j < MEDALS; j++) {
      medalCount += medalCounts[i][j] + "    ";
      // Maintain running total of individual row total
      medalTotal += medalCounts[i][j];
   }
   medalCount += medalTotal;
   // Start a new line at the end of the row
   medalCount += "\n";
}
JOptionPane.showMessageDialog(null, medalCount);
```



Add Column Totals (Rows of Each Column in Two-Dimensional Array

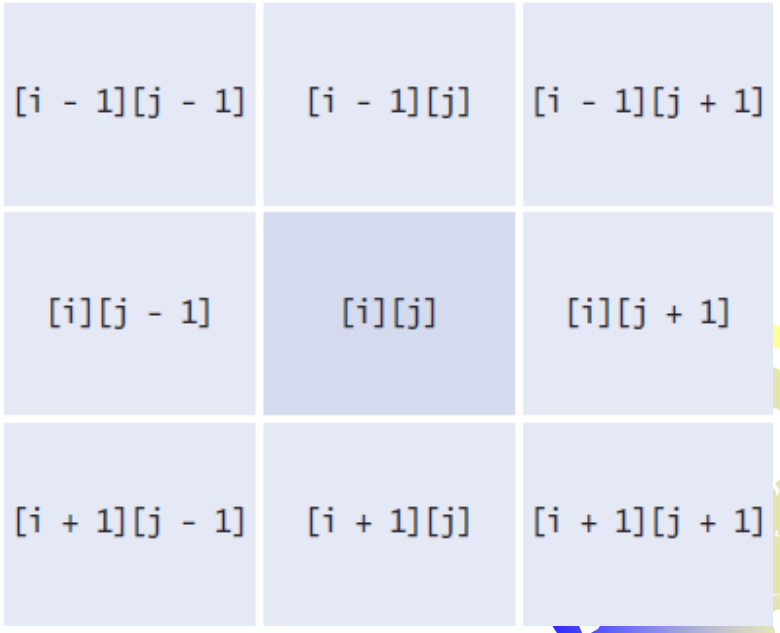
- Loop through the columns, then rows and keep a total of the column values




```
// Process the "j"th column
for (int j = 0; j < MEDALS; j++) {
   int columnTotal = 0;
   // Process the "j"th column in the "i"th row
   for (int i = 0; i < COUNTRIES; i++) {
      columnTotal += medalCounts[i][j];
   }
   medalCount += "    " + columnTotal;
}
```

Locating Neighboring Elements

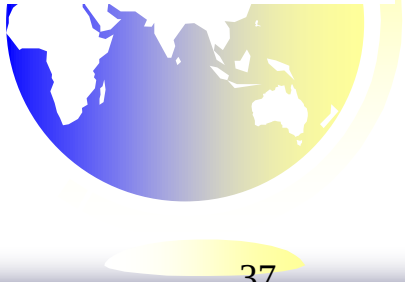
- **Some programs that work with two-dimensional arrays need to locate elements that are adjacent to an element**
  - Very common in gaming
  - You are always at loc (i, j)
  - Be careful of edges
    - No negative indexes
    - Not "off the board"



Two-Dimensional Array Parameters

- **Working with two-dimensional arrays and methods is the same as one dimensional arrays**
  - Use [][] when passing the array

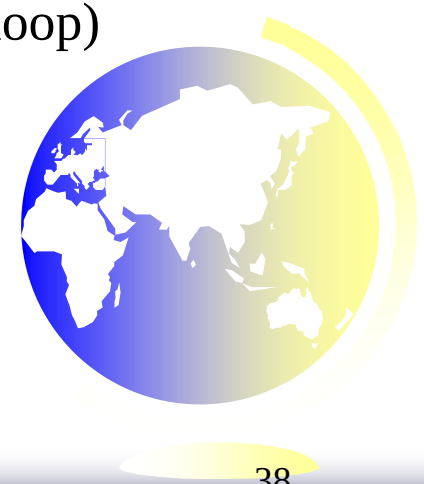
```
public static void main(String[] args) {
   String[] countries = getCountries();
   int[][] medalCounts = getMedalCounts();
   printMedalCount(countries, medalCounts);
}

// Methods
private static String[] getCountries() {}
private static int[][] getMedalCounts() {}
private static void printMedalCount(String[] countries, int[][] medalCounts) {}
```



Two-Dimensional Arrays with Variable Row Lengths

- **When declaring a two-dimensional array, you can specify the size of both elements**
  - Example: int[][] myArray = new int[3][3];
- **You may need an array that has a triangular, or different shape other than a matrix**
- **To do this, you must:**
  - Indicate you will manually set each row by leaving the second index empty
    - int[][] myArray = new int[3][]
  - Allocate each row separately (manually or through a loop)

```
for (int x = 0; x < myArray.length; x++) {
        myArray[x] = new int[x + 1];
}
```



More than Two-Dimensional Arrays

- **You can declare arrays with more than two dimensions**
  - Example: int[][][] rubiksCube = new int[3][3][3]
- **Each array element is specified by three index values (rubiksCube[i][j][k])**
  - Example: rubiksCube[0][0][1]
- **Dimensions beyond three are possible**
  - Becomes logically difficult to follow
  - Question if you really need to do this!

Content

- Introduction
- Arrays
- Sample problem
- Two-Dimensional Arrays
- Case study

Grade Summary Case Study

- **You have been provided with the following table of data**

Name	Test 1	Test 2	Test 3
Alex	87	96	70
Brittany	68	87	90
Bryan	94	100	90
Carolyn	100	81	82
David	83	65	85
Joe	78	87	65
Kelly	85	75	83
Lisa	91	94	100
Mary	76	72	84
Sam	87	93	73

Grade Summary Case Study

- **Based on the table provided, write Java methods that perform the following actions:**
  - Print the number of students in the gradebook
  - Print the number of tests entered in the gradebook
  - Print the number of grades entered in the gradebook
  - Print a list of all student names and their grades
    - Include the average grade earned
  - Print the minimum grade entered
  - Print the maximum grade entered
  - Print the grades of a student based on a name

Questions?