Arrays

CSC 2014 – Java Bootcamp

Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Some slides in this presentation are adapted from the slides accompanying Java Software Solutions by Lewis & Loftus

Java Bootcamp  Dr. Papalaskari  Villanova University

Arrays

• An array is an ordered list of values:

```
0  1  2  3  4  5  6  7  8  9
7.9 8.7 9.4 8.2 6.7 9.8 8.7 8.1 7.4 9.1
```

This array holds 10 values of type double that are indexed from 0 to 9

CSC 1051 M.A. Papalaskari, Villanova University
Arrays - Overview

- **Declaration:**
  
  ```java
  double[] scores = new double[10];
  ```

  - The entire array has a single name
  - **element type**

- **Initialization:**

  ```java
  scores[0] = 7.9;
scores[1] = 8.7;
scores[2] = 9.4;
scores[3] = 8.2;
scores[4] = 6.7;
scores[5] = 9.8;
scores[6] = 8.7;
scores[7] = 8.1;
scores[8] = 7.4;
scores[9] = 9.1;
  ```

  - This array holds 10 values of type `double` that are indexed from 0 to 9
  - The size of the array is given by `scores.length = 10`

- **Instantiation:**

  - Declaration:

    ```java
    double[] scores
    ```

  - Initialization:

    ```java
    double[] scores = {7.9, 8.7, 9.4, 8.2, 6.7, 9.8, 8.7, 8.1, 7.4, 9.1};
    ```

  - Alternative declaration/instantiation/Initialization:

    ```java
    double[] scores = (7.9, 8.7, 9.4, 8.2, 6.7, 9.8, 8.7, 8.1, 7.4, 9.1);
    ```
Arrays - Instantiation

- Declaration:
  
  \[
  \text{double[]} \ text{scores} = \text{new} \text{ double}[10];
  \]

- Instantiation:
  
  The entire array has a single name.

  \[
  \text{scores[2]}
  \]

Arrays - Initialization

- Declaration:
  
  \[
  \text{double[]} \ text{scores} = \text{new} \text{ double}[10];
  \]

- Instantiation:
  
  The entire array has a single name.

  \[
  \text{scores[2]}
  \]

- Initialization:
  
  \[
  \begin{align*}
  \text{scores[0]} &= 7.9; \\
  \text{scores[1]} &= 8.7; \\
  \text{scores[2]} &= 9.4; \\
  \text{scores[3]} &= 8.2; \\
  \text{scores[4]} &= 6.7; \\
  \text{scores[5]} &= 9.8; \\
  \text{scores[6]} &= 8.7; \\
  \text{scores[7]} &= 8.1; \\
  \text{scores[8]} &= 7.4; \\
  \text{scores[9]} &= 9.1;
  \end{align*}
  \]

  Size of array:

  \[
  \text{scores.length} = 10
  \]
Declaring and instantiating Arrays

- Some more examples:

```java
int[] weights = new int[2000];
boolean[] flags;
flags = new boolean[20];
char[] codes = new char[1750];
double[] prices = new double[500];
```

Using Arrays

Array elements can be assigned a value, printed, or used in a calculation. Examples:

```java
System.out.println("Top = " + scores[5]);
mean = (scores[0] + scores[1])/2;
scores[3] = 7 + Math.random();
scores[scores.length - 1] = 9.0;
String answer = args[0];
```
Try this: Write some Java code to create an array

- declare and instantiate an array `ratings` that holds 5 values type `int`

```
ratings[0] 1 2 3 4
```

- declare and instantiate an array `vowel` to hold 5 values of type `char`, then initialize its values to the vowels 'a', 'e', 'i', 'o', 'u'

```
vowel[0] 'a' 'e' 'i' 'o' 'u'
vowel.length 5
```

What gets printed?

```
System.out.println(scores[8] + 1);
System.out.println(scores[1] + scores[2]);
System.out.println(scores[1 + 2]);
System.out.println(scores[scores.length - 2]);
```
Arrays

Java Bootcamp - Villanova University

Dr. Papalaskari

6

**Show how scores values change:**

```java
scores[4] = 1;
scores[5] = scores[0] + 1;
scores[scores.length - 2]) = 5.5;
```

**Processing Arrays using for-loops:**

1) draw a picture of the resulting array

```java
double[] mylist = new double[10];
for (int i = 0; i < 10; i++)
    mylist[i] = 0;

for (int i = 0; i < 10; i++)
    mylist[i] = i;
```
Processing Arrays using for-loops:
2) Reversing through...

```java
double[] tinyScores = new double[5];
for (int i = 4; i >= 0; i--)
    System.out.println(tinyScores[i]);
```

Output:
```
 0 1 2 3 4
7.9 8.7 9.4 8.2 6.7
```

Processing Arrays using for-loops:
3) write a for-loop to print the values in the `vowel` array (going forward)

```
0 1 2 3 4
'a' 'e' 'i' 'o' 'u'
```

4) write a for-loop to print the values in the `vowel` array (going backward)
Bounds Checking

**An array index must specify a valid element**

- Example: If an array `codes` holds 100 values, it can be indexed from 0 to 99. If the value of `count` is 100, then
  ```java
  System.out.println(codes[count]);
  ```
  causes an `ArrayIndexOutOfBoundsException`

- It’s common to introduce **off-by-one errors** when using arrays:
  ```java
  for (int index=0; index <= 100; index++)
      codes[index] = index*50 + epsilon;
  ```

Initializer Lists

- Alternative way to declare, instantiate, and initialize an array. For example:
  ```java
  int[] ratings = {4, 3, 3, 1, 4, 2, 1, 0, 3, 4};
  char[] grades = {'A', 'B', 'C', 'D', 'F'};
  ```

- **NOTE:**
  - the `new` operator is **not** used
  - size of array is determined by the number of items listed
  - can only be used in the array declaration
The “for-each” Loop

• A simple way of processing every array element:

```java
for (double score : scores)
    System.out.println(score);
```

**NOTE:**
• Only appropriate when processing all array elements starting at index 0
• It can't be used to set the array values

Another example

```java
String[] animals = {"dog", "cat", "mouse", "fox");
for (String word : animals)
    System.out.println("The " + word + " ate the cake");

for (String word : animals)
    for (String otherWord : animals)
        System.out.println("The " + word + " ate the " + otherWord);
```
Try this: Use the “for each” loop to scan through an array of `int` containing ratings (range: 0 - 4) and count up how many 4’s.

```java
int[] ratings = {4, 3, 3, 1, 4, 3, 1, 0, 3, 4};
```

Try this: Repeat, but now count up the 0’s, 1’s, … 4’s – Use a separate array for this

```java
int[] ratings = {4, 3, 3, 1, 4, 3, 1, 0, 3, 4};
```
More array examples (see textbook):

- BasicArray.java
- Primes.java
- ReverseOrder.java
- LetterCount.java

```java
//********************************************************************
// ReverseOrder.java          Author: Lewis/Loftus
// Demonstrates array index processing.
//********************************************************************
import java.util.Scanner;
public class ReverseOrder {
    public static void main (String[] args) {
        Scanner scan = new Scanner (System.in);
        double[] numbers = new double[10];
        System.out.println ("The size of the array: " + numbers.length);
        continue
    }
}
```
Arrays

Java Bootcamp - Villanova University

CSC 1051 M.A. Papalaskari, Villanova University

```java
continue
for (int index = 0; index < numbers.length; index++)
{
    System.out.print ("Enter number " + (index+1) + " : ");
    numbers[index] = scan.nextDouble();
}
System.out.println ("The numbers in reverse order:");
for (int index = numbers.length-1; index >= 0; index--)
    System.out.print (numbers[index] + "  ");
}
```

Sample Run

The size of the array: 10
Enter number 1: 18.36
Enter number 2: 48.9
Enter number 3: 53.5
Enter number 4: 29.06
Enter number 5: 72.404
Enter number 6: 34.8
Enter number 7: 63.41
Enter number 8: 45.55
Enter number 9: 69.0
Enter number 10: 99.18
The numbers in reverse order:
99.18 69.0 45.55 63.41 34.8 72.404 29.06 53.5 48.9 18.36

places numbers in an array, then prints them out **backward**

… alternatively, we could place the numbers in the array **backward** and then print them **forward**
Another example: Computing letter frequency counts

Sample run:

Enter a sentence:
In Casablanca, Humphrey Bogart never says "Play it again, Sam."

<table>
<thead>
<tr>
<th>Letter</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
</tr>
</tbody>
</table>

Let's write a program to do this:

```java
import java.util.Scanner;

public class LetterCount {
    public static void main (String[] args) {
        final int NUMCHARS = 26;
        Scanner scan = new Scanner (System.in);
        int[] upper = new int[NUMCHARS];
        int[] lower = new int[NUMCHARS];
        char current;  // the current character being processed
        int other = 0;  // counter for non-alphabets
        continue
    }
}
```
Arrays

continue
System.out.println("Enter a sentence:");
String line = scan.nextLine();

// Count the number of each letter occurrence
for (int ch = 0; ch < line.length(); ch++)
{
    current = line.charAt(ch);
    if (current >= 'A' && current <= 'Z')
        upper[current-'A']++;
    else
        if (current >= 'a' && current <= 'z')
            lower[current-'a']++;
        else
            other++;
}
// Print the results
System.out.println();
for (int letter=0; letter < upper.length; letter++)
{
    System.out.print((char) (letter + 'A') );
    System.out.print(":" + upper[letter]);
    System.out.print("\t\t" + (char) (letter + 'a') );
    System.out.println(":" + lower[letter]);
}
System.out.println();
System.out.println("" + other);
}

Sample Run

Enter a sentence:
In Casablanca, Humphrey Bogart never says "Play it again, Sam."

A: 0 a: 10
B: 1 b: 1
C: 1 c: 1
D: 0 d: 0
E: 0 e: 3
F: 0 f: 0
G: 0 g: 2
H: 1 h: 1
I: 1 i: 2
J: 0 j: 0
K: 0 k: 0
L: 0 l: 2
M: 0 m: 2
N: 0 n: 4
O: 0 o: 1
P: 1 p: 1
Q: 0 q: 0

Continue

Sample Run (continued)
R: 0 r: 3
S: 1 s: 3
T: 0 t: 2
U: 0 u: 1
V: 0 v: 1
W: 0 w: 0
X: 0 x: 0
Y: 0 y: 3
Z: 0 z: 0

Non-alphabetic characters: 14
What does it mean to “copy an array”? 

• Suppose we have two arrays:

```java
int[] a = {147, 323, 89, 933};
int[] b = {100, 200, 300, 400};
```

**Copying elements vs. copying array variables:**

```java
for (int i=0; i<a.length; i++)
    a[i] = b[i];
```

```java
a = b;
```

Afterwards, what is the effect of the following?

```java
a[1] = 1111;
b[2] = 2222;
```

What does it mean to “copy an array”?

1) Copying elements:

<table>
<thead>
<tr>
<th>a</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>147</td>
<td>323</td>
<td>89</td>
<td>933</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>b</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
<td>200</td>
<td>300</td>
<td>400</td>
</tr>
</tbody>
</table>

What changes?

```java
for (int i=0; i<a.length; i++)
    a[i] = b[i];
```

```java
a[1] = 1111;
b[2] = 2222;
```
2) Copying array variables:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>147</td>
<td>323</td>
<td>89</td>
<td>933</td>
</tr>
<tr>
<td>b</td>
<td>100</td>
<td>200</td>
<td>300</td>
<td>400</td>
</tr>
</tbody>
</table>

What changes?

```java
a = b;
```

```java
a[1] = 1111;
b[2] = 2222;
```

Two-Dimensional Arrays

- A one-dimensional array stores a list of elements
- A two-dimensional array can be thought of as a table of elements, with rows and columns
Two-Dimensional Arrays

• To be precise, in Java a two-dimensional array is an array of arrays

• A two-dimensional array is declared by specifying the size of each dimension separately:

\[
\text{int[][] table = new int[12][50];}
\]

• A array element is referenced using two index values:

\[
\text{value = table[3][6]}
\]

• The array stored in one row can be specified using one index

\[
\begin{align*}
\text{0} & \quad 1 & \quad 2 & \quad 3 & \quad 4 & \quad 5 & \quad 6 & \quad 7 & \quad 8 & \quad 9 \\
7.9 & \quad 8.7 & \quad 9.4 & \quad 8.2 & \quad 6.7 & \quad 9.8 & \quad 8.7 & \quad 8.1 & \quad 7.4 & \quad 9.1 \\
9.3 & \quad 5.8 & \quad 6.9 & \quad 5.5 & \quad 9.0 & \quad 8.3 & \quad 7.7 & \quad 9.2 & \quad 9.8 & \quad 8.2 \\
8.9 & \quad 8.0 & \quad 8.4 & \quad 6.2 & \quad 7.7 & \quad 7.3 & \quad 9.6 & \quad 6.1 & \quad 7.8 & \quad 7.3
\end{align*}
\]
Two-Dimensional Arrays

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>int[][]</td>
<td>2D array of integers, or array of integer arrays</td>
</tr>
<tr>
<td>table[5]</td>
<td>int[]</td>
<td>array of integers</td>
</tr>
<tr>
<td>table[5][12]</td>
<td>int</td>
<td>integer</td>
</tr>
</tbody>
</table>

- See TwoDArray.java
- See SodaSurvey.java

```java
import java.util.*;

public class SodaSurvey {
    public static void main(String[] args) {
        int[][] survey = new int[5][12];
        for (int row=0; row < survey.length; row++)
        for (int col=0; col < survey[row].length; col++)
            survey[row][col] = row * 10 + col;
        // Print the survey
        for (int row=0; row < survey.length; row++)
        for (int col=0; col < survey[row].length; col++)
            System.out.println(survey[row][col] + "\t");
    }
}
```
Multidimensional Arrays

- An array can have many dimensions – if it has more than one dimension, it is called a **multidimensional array**

- Each dimension subdivides the previous one into the specified number of elements

- Each dimension has its own length constant

- Because each dimension is an array of array references, the arrays within one dimension can be of different lengths
  - these are sometimes called **ragged arrays**