Introduction to Arrays

CSC 1051 – Data Structures and Algorithms I

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

Course website:
www.csc.villanova.edu/~map/1051/

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

- Arrays are objects that help us organize large amounts of information

- Chapter 8 focuses on:
  - array declaration and use
  - bounds checking and capacity
  - arrays that store object references
  - variable length parameter lists
  - multidimensional arrays
  - polygons and polylines
  - mouse events and keyboard events
Arrays

- An array is an ordered list of values:

The entire array has a single name
Each value has a numeric index

scores

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>79</td>
<td>87</td>
<td>94</td>
<td>82</td>
<td>67</td>
<td>98</td>
<td>87</td>
<td>81</td>
<td>74</td>
<td>91</td>
</tr>
</tbody>
</table>

An array of size N is indexed from zero to N-1

This array holds 10 values that are indexed from 0 to 9
Arrays

- A particular value in an array is referenced using the array name followed by the index in brackets.

- For example, the expression `scores[2]` refers to the value 94 (the 3rd value in the array).

- That expression represents a place to store a single integer and can be used wherever an integer variable can be used.
Arrays

• For example, an array element can be assigned a value, printed, or used in a calculation:

```java
scores[2] = 89;
scores[first] = scores[first] + 2;
mean = (scores[0] + scores[1])/2;
System.out.println("Top = " + scores[5]);
pick = scores[rand.nextInt(11)];
```
Arrays

• The values held in an array are called *array elements*
• An array stores multiple values of the same type – the *element type*
• The element type can be a primitive type or an object reference
• Therefore, we can create an array of integers, an array of *Char* objects, an array of *String* objects, an array of *Coin* objects, etc.
Arrays

- In Java, the array itself is an object that must be instantiated
- Another way to depict the `scores` array:

The name of the array is an object reference variable.
Declaring Arrays

• The **scores** array could be declared as follows:

```java
int[] scores = new int[10];
```

• The type of the variable **scores** is **int[]** (an array of integers)

• Note that the array type does not specify its size, but each object of that type has a specific size

• The reference variable **scores** is set to a new array object that can hold 10 integers
Declaring Arrays

• Some other examples of array declarations:

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

• The for-each version of the `for` loop can be used when processing array elements:

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

• This is only appropriate when processing all array elements starting at index 0

• It can't be used to set the array values

• See BasicArray.java
public class BasicArray
{
    public static void main (String[] args)
    {
        final int LIMIT = 15, MULTIPLE = 10;

        int[] list = new int[LIMIT];

        // Initialize the array values
        for (int index = 0; index < LIMIT; index++)
            list[index] = index * MULTIPLE;

        list[5] = 999;  // change one array value

        // Print the array values
        for (int value : list)
            System.out.print (value + " ");
    }
}
public class BasicArray
{
    public static void main (String[] args)
    {
        final int LIMIT = 15, MULTIPLE = 10;

        int[] list = new int[LIMIT];

        // Initialize the array values
        for (int index = 0; index < LIMIT; index++)
            list[index] = index * MULTIPLE;

        list[5] = 999;  // change one array value

        // Print the array values
        for (int value : list)
            System.out.print (value + " ");
    }
}
Basic Array Example

The array is created with 15 elements, indexed from 0 to 14.

After three iterations of the first loop:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
<td>110</td>
<td>120</td>
<td>130</td>
<td>140</td>
</tr>
</tbody>
</table>

After completing the first loop:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
<td>110</td>
<td>120</td>
<td>130</td>
<td>140</td>
</tr>
</tbody>
</table>

After changing the value of list[5]:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>999</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
<td>110</td>
<td>120</td>
<td>130</td>
<td>140</td>
</tr>
</tbody>
</table>

CSC 1051 M.A. Papalaskari, Villanova University
Bounds Checking

• Once an array is created, it has a fixed size

• An index used in an array reference must specify a valid element

• That is, the index value must be in range 0 to N-1

• The Java interpreter throws an `ArrayIndexOutOfBoundsException` if an array index is out of bounds

• This is called automatic bounds checking
Bounds Checking

• For example, if the array `codes` can hold 100 values, it can be indexed from 0 to 99.

• If the value of `count` is 100, then the following reference will cause an exception to be thrown:

  ```java
  System.out.println(codes[count]);
  ```

• 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;
  ```
Bounds Checking

• Each array object has a public constant called `length` that stores the size of the array

• It is referenced using the array name:

  `scores.length`

• Note that `length` holds the number of elements, not the largest index

• See [ReverseOrder.java](#)
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

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
Counting letters - Algorithm

Uses two arrays: upper and lower with 26 elements each, to count the number of upper- and lowercase letters, respectively.

1. declare and instantiate upper and lower with 26 elements of type int
2. prompt user to "Enter a sentence:" and scan input into a String line
3. for each character current in line do the following:
   - if (current is between ‘A’ and 'Z')
     • position $\leftarrow$ current - 'A' // distance from ‘A’
     • upper[position] $\leftarrow$ upper[position] + 1
   - if (current is between ‘a’ and ‘z')
     • position $\leftarrow$ current – ‘a’
     • lower[position] $\leftarrow$ upper[position] + 1

1. CSC 1051 M.A. Papalaskari, Villanova University
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-alphabetics

    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++;}

continue
// 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 ("Non-alphabetic characters: " + other);
### Sample Run

Enter a sentence:

In Casablanca, Humphrey Bogart never says "Play it again, Sam."

<table>
<thead>
<tr>
<th>Letter</th>
<th>Count</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>
<tr>
<td>F</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>2</td>
</tr>
<tr>
<td>H</td>
<td>1</td>
</tr>
<tr>
<td>h</td>
<td>1</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>i</td>
<td>2</td>
</tr>
<tr>
<td>J</td>
<td>0</td>
</tr>
<tr>
<td>j</td>
<td>0</td>
</tr>
<tr>
<td>K</td>
<td>0</td>
</tr>
<tr>
<td>k</td>
<td>0</td>
</tr>
<tr>
<td>L</td>
<td>0</td>
</tr>
<tr>
<td>l</td>
<td>2</td>
</tr>
<tr>
<td>M</td>
<td>0</td>
</tr>
<tr>
<td>m</td>
<td>2</td>
</tr>
<tr>
<td>N</td>
<td>0</td>
</tr>
<tr>
<td>n</td>
<td>4</td>
</tr>
<tr>
<td>O</td>
<td>0</td>
</tr>
<tr>
<td>o</td>
<td>1</td>
</tr>
<tr>
<td>P</td>
<td>1</td>
</tr>
<tr>
<td>p</td>
<td>1</td>
</tr>
<tr>
<td>Q</td>
<td>0</td>
</tr>
<tr>
<td>q</td>
<td>0</td>
</tr>
<tr>
<td>R</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>3</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
</tr>
<tr>
<td>s</td>
<td>3</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
</tr>
<tr>
<td>t</td>
<td>2</td>
</tr>
<tr>
<td>U</td>
<td>0</td>
</tr>
<tr>
<td>u</td>
<td>1</td>
</tr>
<tr>
<td>V</td>
<td>0</td>
</tr>
<tr>
<td>v</td>
<td>1</td>
</tr>
<tr>
<td>W</td>
<td>0</td>
</tr>
<tr>
<td>w</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>3</td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
</tr>
<tr>
<td>z</td>
<td>0</td>
</tr>
</tbody>
</table>

Non-alphabetic characters: 14

### Sample Run (continued)

<table>
<thead>
<tr>
<th>Letter</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>3</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
</tr>
<tr>
<td>s</td>
<td>3</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
</tr>
<tr>
<td>t</td>
<td>2</td>
</tr>
<tr>
<td>U</td>
<td>0</td>
</tr>
<tr>
<td>u</td>
<td>1</td>
</tr>
<tr>
<td>V</td>
<td>0</td>
</tr>
<tr>
<td>v</td>
<td>1</td>
</tr>
<tr>
<td>W</td>
<td>0</td>
</tr>
<tr>
<td>w</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>3</td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
</tr>
<tr>
<td>z</td>
<td>0</td>
</tr>
</tbody>
</table>

Non-alphabetic characters: 14
Alternate Array Syntax

• The brackets of the array type can be associated with the element type or with the name of the array

• Therefore the following two declarations are equivalent:

  ```
  double[] prices;
  double prices[];
  ```

• The first format generally is more readable and should be used
Initializer Lists

• An *initializer list* can be used to instantiate and fill an array in one step

• The values are delimited by braces and separated by commas

• Examples:

```c
int[] units = {147, 323, 89, 933, 540, 269, 97, 114, 298, 476};

char[] grades = {'A', 'B', 'C', 'D', 'F'};
```
Initializer Lists

• Note that when an initializer list is used:
  – the new operator is not used
  – no size value is specified

• The size of the array is determined by the number of items in the list

• An initializer list can be used only in the array declaration

• See Primes.java
public class Primes
{
    // Stores some prime numbers in an array and prints them.
    public static void main (String[] args)
    {
        int[] primeNums = {2, 3, 5, 7, 11, 13, 17, 19};
        System.out.println ("Array length: " + primeNums.length);
        System.out.println ("The first few prime numbers are: ");
        for (int prime : primeNums)
            System.out.print (prime + " ");
    }
}
public class Primes
{
    // Stores some prime numbers in an array and prints them.
    public static void main (String[] args)
    {
        int[] primeNums = {2, 3, 5, 7, 11, 13, 17, 19};

        System.out.println ("Array length: " + primeNums.length);

        System.out.println ("The first few prime numbers are:");

        for (int prime : primeNums)
            System.out.print (prime + "  ");
    }
}
Arrays as Parameters

• An entire array can be passed as a parameter to a method (just like any other object).

• Exercise: Write a method that increments the value of each element in an array.
Homework

- Review Sections 8.1, 8.2
- Read Section 8.3 to prepare for next class

Exercises

- From textbook: EX8.6 - EX8.9
- Handout with more questions