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
Outline

Declaring and Using Arrays
Arrays of Objects
Variable Length Parameter Lists
Two-Dimensional Arrays
Polygons and Polylines
Mouse Events and Key Events
Arrays

- An array is an ordered list of values:

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

  scores

  0 1 2 3 4 5 6 7 8 9
  79 87 94 82 67 98 87 81 74 91

  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

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

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:

```
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 characters, 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
{
    //                                                                           
    //  Creates an array, fills it with various integer values,                 
    //  modifies one value, then prints them out.                              
    //                                                                           
    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:

0  0
1 10
2 20
3
4
5
6
7
8
9
10
11
12
13
14

After completing the first loop:

0  0
1 10
2 20
3 30
4 40
5 50
6 60
7 70
8 80
9 90
10 100
11 110
12 120
13 130
14 140

After changing the value of list[5]:

0  0
1 10
2 20
3 30
4 40
5 999
6 60
7 70
8 80
9 90
10 100
11 110
12 120
13 130
14 140

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

• See LetterCount.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] + "  ");
    }
```java
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] + "  ");
```
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 ← current - 'A' // distance from ‘A’
     • upper[position] ← upper[position] + 1
   - if (current is between ‘a’ and ‘z’)
     • position ← current – ‘a’
     • lower[position] ← 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
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."

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
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
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: 2 3 5 7 11 13 17 19");
    }
}
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

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