Building Java Programs

Chapter 7: Arrays

Chapter outline

- Array basics
  - Declaring and initializing an array
  - Getting and setting values of elements of an array
  - Looping through arrays
- Array traversal algorithms
  - Printing an array's elements
  - Searching an array
  - Reversing an array
- Advanced array usage
  - Arrays as parameters to methods
  - String and Graphics methods that use arrays
  - The Arrays class
  - Command-line arguments
  - Shifting elements in an array

Array basics

Reading: 7.1

Why the problem is tough

- We appear to need each input value twice:
  - Once to compute the average
  - A second time to count how many were above average
- We could read each value into a variable...
  - However, we don't know how many variables to declare.
  - We don't know how many days' weather will be typed until the program is running.
- We need a way to declare many variables in one step.

Arrays

- array: A variable that stores many values of the same type.
  - element: One value in an array.
  - index: A 0-based integer used to access an element from an array.
- We usually draw an array as a row or column of boxes.
  - Example: An array of ten integers

<table>
<thead>
<tr>
<th>Index</th>
<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>value</td>
<td>12</td>
<td>49</td>
<td>2</td>
<td>36</td>
<td>5</td>
<td>17</td>
<td>6</td>
<td>84</td>
<td>72</td>
<td>3</td>
</tr>
</tbody>
</table>

Arrays

A problem we can't solve (yet)

Consider the following program (input underlined):

```
How many days' temperatures? 2
Day 1's high temp: 45
Day 2's high temp: 24
Day 3's high temp: 77
Day 4's high temp: 44
Day 5's high temp: 77
Day 6's high temp: 44
Day 7's high temp: 77
Average temp = 4.57142857142857
```

- We need the temperatures to compute the average, and again to tell how many were above average.
Array declaration

- Declaring/initializing an array:
  `<type>[:<name>] = new <type>[:<length>]();`
- The length of the array is specified between `[]` brackets.
- Example:
  ```java
  int[] numbers = new int[10];
  ```
- The array's length can be any expression.
  - Example:
    ```java
    int a = 2 * 3 + 1;
    int[] data = new int[a * 5 + 2];
    ```

Array auto-initialization

- When arrays are initially constructed, every element is automatically initialized to a "zero-equivalent" value.
  - `int`:
    ```java
    int[] intNumbers = new int[4];
    ```
  - `double`:
    ```java
    double[] doubleNumbers = new double[4];
    ```
  - `boolean`:
    ```java
    boolean[] booleanNumbers = new boolean[4];
    ```
  - `String` or `object`:
    ```java
    String[] stringNumbers = new String[4];
    ```

Accessing array elements

- Assigning a value to an array element:
  `<array name> :<index> = <value> ;`
- Example:
  ```java
  numbers[0] = 27;
  numbers[3] = 4;
  ```

Accessing array elements

- Using an array element's value in an expression:
  `<array name> :<index>`
- Using an array element's value in an expression:
  ```java
  System.out.println(numbers[0]);
  if (numbers[3] < 0) System.out.println("Element 3 is negative.");
  ```

Out-of-bounds indexes

- The indexes that are legal to access in an array are those in the range of 0 to the array's length - 1.
- Reading or writing any index outside this range will throw an `ArrayIndexOutOfBoundsException`.
- Example:
  ```java
  int[] data = new int[10];
  System.out.println(data[0]); // okay
  System.out.println(data[-1]); // exception
  System.out.println(data[10]); // okay
  System.out.println(data[5]); // exception
  ```

Arrays of other types

- Arrays can contain other types, such as `double`.
  ```java
  double[] results = new double[5];
  results[0] = 3.4;
  results[1] = -3.1;
  ```
- Example:
  ```java
  boolean[] tests = new boolean[4];
  tests[0] = true;
  ```
- Arrays can contain other types, such as `double`.
  ```java
  double[] results = new double[5];
  results[0] = 3.4;
  ```
- Example:
  ```java
  boolean[] tests = new boolean[4];
  tests[0] = true;
  ```
### Accessing array elements

- A longer example of accessing and changing elements:
  ```java
  int[] numbers = new int[8];
  numbers[1] = 4;
  numbers[4] = 99;
  numbers[7] = 2;
  int x = numbers[1];
  numbers[4] = 66;
  numbers[numbers[7]] = 11; // use numbers[7] as index
  x = 4
  
  0 1 2 3 4 5 6 7
  numbers 0 4 11 64 7 2
  ```

### Arrays and for loops

- Arrays are very commonly used with for loops that pass over each element and process it in some way:
  ```java
  // Example print each element of an array:
  for (int i = 0; i < 8; i++)
      System.out.println(numbers[i] + ":");
  System.out.println(); // end the line of output
  
  // Output: (when used on array from previous slide):
  0 4 11 64 0 0 2
  ```

### More arrays and for loops

- Sometimes we assign each array element a value in a for loop.
  ```java
  // Example:
  for (int i = 0; i < 15; i++)
      index 0 1 2 3 4 5 6 7
      value 0 2 4 6 8 10 12 14
  ```

- What values would be stored into the array after this code?
  ```java
  // Example:
  for (int i = 0; i < 15; i++)
      value 0 1 4 9 16 25 36 49
  ```

### The .length field

- An array’s `.length` field stores its number of elements.
  ```java
  // General syntax:
  <array name>.length
  ```

- It does not use parentheses like a String’s `.length()`.

- Example (using array from previous slide):
  ```java
  // Example:
  for (int i = 0; i < numbers.length; i++)
      System.out.println(numbers[i] + ":");
  ```

- Output:
  0 4 11 64 0 0 2

- What expression refers to the last element of the array? The middle element?

### Weather question

- Use an array to solve the weather problem:
  ```java
  // This program makes several day's temperatures from the user.
  // Precise data would be averaged and the day's high and low average.
  // Output: (in Celsius)
  ```

- Day 1’s high temp: 41
- Day 2’s high temp: 44
- Day 3’s high temp: 35
- Day 4’s high temp: 48
- Day 5’s high temp: 37
- Day 6’s high temp: 45
- Day 7’s high temp: 33
- Average temp = 44.57162807142857
- 4 days were above average.

### Weather answer

```java
// This program makes several day's temperatures from the user.
// Precise data would be averaged and the day's high and low average.
// Output: (in Celsius)
```
Arrays for counting and tallying
reading: 7.1

A multi-counter problem
• Problem: Examine a large integer and count the number of occurrences of every digit from 0 through 9.
  • Example: The number 320230107 contains:
    two 0s, one 1, three 2s, one 7, and one 9.
• We need to examine each digit of the large integer and count how many times we’ve seen that digit.
  • This will require counters for each of the values 0–9.
• We could declare 10 counter variables for this...
  • A better solution is to use an array of size 10.
  • The element at index i will store the counter for digit value i.

Creating an array of tallies
• The following code builds an array of digit counters:

```
int num = 23202107;
int[] count = new int[10];
while(num > 0)
{
    int digit = num % 10;
    count[digit]++;
    num = num / 10;
}

Index 0 1 2 3 4 5 6 7 8 9
value 7 1 3 3 0 0 0 0 1 0 1
```

Array histogram question
• Given a file of integer exam scores, such as:
  22
  73
  13

Write a program that will print a histogram of stars indicating the number of students who earned each unique exam score.

```
? ??
? ???
? ??
? ?
? ??

Variations:
• Make a curve that adds a fixed number of points to each score.
  (But don’t allow a curved score to exceed the max of 100.)
• Chart the data with a DrawingPanel.
```

Array histogram answer
// Read an input file of exam scores (integers) and display a // histogram (graph of the score distribution).
import java.io.*;
import java.util.*;
public class Histogram
{
    public static void main(String[] args)
    {
        // Open an input file
        // Print a blank line
        System.out.println();
        // Print the header
        System.out.println("Score | Frequency");
        // Print a blank line
        System.out.println();
        // Read the data
        // Create an array of frequency counters
        int[] frequency = new int[100];
        // Loop through the data
        for(int i = 0; i < frequency.length; i++)
            // Print a histogram
            System.out.println("Score: "+i+" Frequency: "+frequency[i]);
        // Print a blank line
        System.out.println();
    }
}

Array histogram solution 2
...
Why are arrays useful?

- Arrays store a large amount of data in one variable.
  - Example: Read in a file of 1000 numbers, then print out the numbers in reverse order.
- Arrays help us group related data into elements.
  - Example: For a school exam, open a file of exam scores and count how many students got each score from 0 through 100.
- Arrays let us access data in random order.
  - Example: Read a file of weather data, store each month's weather stats as an element in a large array, and then examine the data to find overall weather statistics for the year.

Array initialization statement

- Quick array initialization, general syntax:
  - `type[] name = {value1, value2, ..., valueN};`
  - Example:
    ```java
    int numbers = {10, 44, -3, 5, 7, 16};
    ```
    - Index: 0 1 2 3 4 5 6
    - Value: 10 44 -3 5 7 16
  - This syntax is useful when you know in advance what the array's element values will be.
  - You don't explicitly specify the array's size in this syntax.
  - The Java compiler figures out the size by looking at the number of values written between `{| and |}.`

Array practice problem

- What element values are stored in the following array?

```java
int[] a = {2, 5, 1, 6, 14, 7, 8};
for (int i = 0; i < a.length; i++) {
    a[i] = a[i] - 1;
}

Index: 0 1 2 3 4 5 6
Value: 2 7 8 14 20 35 44
```

The Arrays class

- The `Arrays` class in package `java.util` has several useful static methods for manipulating arrays:

<table>
<thead>
<tr>
<th>Method name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>binarySearch(array, value)</code></td>
<td>returns the index of the given value in this array (less if not found)</td>
</tr>
<tr>
<td><code>equals(array1, array2)</code></td>
<td>returns true if the two given arrays contain exactly the same elements in the same order</td>
</tr>
<tr>
<td><code>fill(array, value)</code></td>
<td>sets every element in the array to the given value</td>
</tr>
<tr>
<td><code>sort(array)</code></td>
<td>arranges the elements in the array into ascending order</td>
</tr>
</tbody>
</table>
| `toString(array)` | returns a string representing the array, such as "[10, 20, 15]"

Arrays.toString

- The `Arrays.toString` method is useful when you want to print an array's elements.
  - `Arrays.toString` accepts an array as a parameter and returns the string representation, which you can then print.
  - Example:
    ```java
    int[] a = {2, 5, 1, 6, 14, 7, 8};
    for (int i = 0; i < a.length; i++) {
        a[i] = a[i] - 1;
    }
    System.out.println("a is " + Arrays.toString(a));
    ```
    - Output:
      ```java
      a is [2, 7, 8, 14, 20, 35, 44]
      ```

Traversal algorithms

- Reading: 7.1, 7.2, 4.4
### Array traversal

- **Traversal**: An examination of each element of an array.
  - Traversal algorithms often take the following form:
    ```java
    for (i = 0; i < array.length; i++) {
        do something with array[i];
    }
    ```
- Traversals are useful in many situations:
  - printing the elements of an array
  - searching an array for a specific value
  - rearranging the elements of an array
  - computing the sum, product, etc. of an array's elements

### Printing array elements

**Example** (print each element of an array on a line):
```java
int[] list = {4, 1, 5, 7};
for (int i = 0; i < list.length; i++) {
    System.out.println("* + list[i];
}
``` 
Output:
```
0: 4
1: 1
5: 7
``` 
- *How could we change the code to print the following?*
  ```
  4, 1, 5, 7
  ```
  [Default: use Arrays.toString()]

### Examining array elements

**Example** (find the largest even integer in an array):
```java
int[] list = {4, 1, 2, 7, 4, 1, 6};
int largestEven = 0;
for (int i = 0; i < list.length; i++) {
    if (list[i] % 2 == 0 && list[i] > largestEven) {
        largestEven = list[i];
    }
}
System.out.println("Largest even: " + largestEven);
``` 
Output:
```
Largest even: 6
``` 

### String traversal

**Strings** are like arrays of characters; they also use 0-based indexes.
- We can write algorithms to traverse strings to compute information:
  ```java
  // string contains a sequence of lowercase letters / 0-9
  // x is a valid character
  int stringLength = s.length(); 
  // s = a 26-letter / 10-digit string
  for (int i = 0; i < stringLength; i++) {
      char c = s.charAt(i);
      if (c >= 'a' && c <= 'z') {
          // lowercase letter
          // we could also check for uppercase
          // or use a 26-letter alphabet
      } 
      System.out.println("Character at index " + i + " is " + c);
  }
  ``` 
- *Example* contains 3 rows and a 9-week attendance record:
  ```
  student1 student2 student3 student4 student5 student6 student7 student8 student9
  week0 1 0 0 1 0 1 1 0 0
  week1 1 0 0 0 1 0 0 0 1
  week2 1 0 1 0 1 0 0 0 0
  week3 1 0 1 0 1 0 0 0 1
  week4 1 0 1 0 1 0 0 0 0
  week5 1 0 1 0 1 0 0 0 1
  ``` 
- *Example* contains 3 sections and a 9-week attendance record:
  ```
  section1 section2 section3
  week0 1 1 1
  week1 1 1 1
  week2 1 1 1
  week3 1 1 1
  week4 1 1 1
  week5 1 1 1
  ``` 

### Section attendance problem

- Consider an input file containing course attendance data.
  - Each line represents a section, and each section has 9 students.
  - Every 9 numbers represent 1 week’s attendance record.
  - 1 indicates the student attended, 0 means the student did not attend.

### Arrays as parameters

**Reading**: 7.1, 3.3
Array transformations

- Problems like this are examples where we use data in one form to compute new data in another form.
  - We sometimes call this transforming the data.
  - Often each transformation is done into its own array.
- Many transformation problems require a mapping between the original data and array indexes.
  - Sometimes the mapping is a fairly easy.
    - (if the input value is the integer, store it into array index i)
- Sometimes the mapping relates to the position of the data.
  - (above the 19 value we read into index i)
- Sometimes the mapping is done explicitly.
  - (count occurrences of "O" into index 0)

Section attendance problem

- Write a program that reads the preceding section data file and produces output such as the following:
  
  ```
  Section #1:
  Sections attended: [9, 6, 7, 6, 5]
  Student scores: [20, 19, 18, 17, 16]
  Section grades: [100.0, 100.0, 100.0, 100.0, 100.0]
  Section #2:
  Sections attended: [4, 7, 5, 5, 4]
  Student scores: [40, 20, 14, 14, 13]
  Section grades: [50.0, 70.0, 75.0, 80.0, 90.0]
  Section #3:
  Sections attended: [5, 6, 5, 7, 6]
  Student scores: [50, 60, 65, 90, 80]
  Section grades: [70.0, 90.0, 90.0, 100.0, 100.0]
  ```

- Assume the input file exists and is valid.

Section attendance solution

```java
// This program makes a file representing which students attended
// which sections and prints the final output of the problem

import java.io.*;
import java.util.*;

public class Attendance {
    public static void main(String[] args) {
        // Read attendance data from the file Attendance.txt
        // and process it to generate the output
        //...
    }
}
```

Section attendance solution 2

```java
// This program makes a file representing which students
// which sections and prints the final output of the problem

import java.io.*;
import java.util.*;

public class Attendance {
    public static void main(String[] args) {
        // Read attendance data from the file Attendance.txt
        // and process it to generate the output
        //...
    }
}
```

Arrays as parameters

- An array can be passed as a parameter.
  - Syntax (declaration):
    ```java
    public static <type> <name> <type> <name>;
    ```
  - Examples:
    ```java
    public static double average(int[] numbers) {
        //...
    }
    ```
  - Syntax (call):
    ```java
    <method name> <array name>;
    ```
  - Examples:
    ```java
    int[] scores = [15, 17, 12, 15, 10];
    double avg = average(scores);
    ```

Array parameter example

```java
// This program makes a file representing which students
// which sections and prints the final output of the problem

import java.io.*;
import java.util.*;

public static void main(String[] args) {
    // Read attendance data from the file Attendance.txt
    // and process it to generate the output
    //...
}
```

```java
// This program makes a file representing which students
// which sections and prints the final output of the problem

import java.io.*;
import java.util.*;

public class Attendance {
    public static void main(String[] args) {
        // Read attendance data from the file Attendance.txt
        // and process it to generate the output
        //...
    }
}
```

```java
// This program makes a file representing which students
// which sections and prints the final output of the problem

import java.io.*;
import java.util.*;

public class Attendance {
    public static void main(String[] args) {
        // Read attendance data from the file Attendance.txt
        // and process it to generate the output
        //...
    }
}
```

Output:

```
Max = 167
```
Arrays as parameters, contd.

Arrays are objects.
- When passed as parameters, they are passed by reference.
- Changes made in the method will also be seen by the caller.

Example:
```java
public static void main(String[] args) {
    int[] arr = {1, 2, 3, 4, 5};
    System.out.println(arr[0] + arr[1] + arr[2]);
}
```

Output:
```
6
```

Output parameters

- output parameter: An array or object passed as a parameter that has its content altered by the method.
- We can pass an array to a method and the method can change its content in useful ways for us.

Example:
The methods `Arrays.fill` and `Arrays.sort`.
```java
int[] nums = {1, 2, 3, 4, 5};
Arrays.sort(nums); // modify contents of nums
Arrays.fill(nums, 2); // replace contents of nums
```

Arrays as return values

- An array can also be returned from a method.
- Syntax (declaration):
  ```java
  public static <type> <name>(<parameters>) { ...
  }
  ```
- Example:
  ```java
  public static int[] readAllIntegers(Scanner scanner) {
      // read all integers from scanner
  }
  ```

Array return example

- Example: (digit-counting problem from an earlier slide):
  ```java
  public static int[] countDigits(int n) {
      int counts = new int[10];
      while (n > 0) {
          counts[n % 10]++;
      }
      return counts;
  }
  ```

Output:
```
[0, 1, 1, 0, 0, 0, 0, 0, 0, 0]
```

Array parameter questions

- Write a method named `average` that accepts an array of integers as its parameter and returns the average of the values in the array.
- Write a method named `contains` that accepts an array of integers and a target integer value as its parameters and returns whether the array contains the target value as one of its elements.
- Write a method named `roundAll` that accepts an array of doubles as its parameter and modifies each array element by rounding it to the nearest whole number.
- Improve the previous `Histogram` and `Sections` programs by making them use parameterized methods.
Array parameter answers

```java
public static double average(int[] numbers)
{
    for (int i = 0; i < numbers.length; i++)
        return (double) sum / numbers.length;
}

public static boolean contains(int[] values, int target)
{
    for (int i = 0; i < values.length; i++)
        if (values[i] == target)
            return true;

    return false;
}

public static void round(double[] array)
{
    array[i] = Math.round(array[i]);
}
```

Concept of an array rotation

- Imagine we want to 'rotate' the elements of an array; that is, to shift them left by one index.
- The element that used to be at index 0 will move to the last slot in the array.
- For example, [3, 8, 5, 7, 5] becomes [8, 5, 7, 5, 3].

Before: Index 0 1 2 3 4
value 3 8 5 7 5

After: Index 0 1 2 3 4
value 8 9 7 5 3

- Shifting elements is useful when inserting and removing values from arrays after they have already been filled with data.

Shifting elements left

- A left shift of the elements of an array:

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>3</td>
<td>9</td>
<td>9</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>

- Let's write the code to do the left shift.
- Can we generalize it so that it will work on an array of any size?
- Can we write a right shift as well?

Shifting practice problem

- Write a method `insertionOrder` that accepts as sorted array `a` of integers and an integer value `n` as parameters, and inserts `n` into a while maintaining sorted order.

In other words, assume that the element values in `a` occur in sorted ascending order, and insert the new value `n` into the array at the appropriate index, shifting to make room if necessary. The last element in the array will be last after the insertion.

- Example: calling `insertionOrder` on array `[1, 3, 7, 10, 12, 14, 22, 27, 34]` and value `11` produces `[1, 3, 7, 11, 10, 12, 14, 22, 27, 34]` and value `15` produces `[1, 3, 7, 10, 12, 14, 15, 22, 27, 34]`.