Building Java Programs

Chapter 7
Lecture 7-1: Arrays

reading: 7.1
THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:

"MY CODE'S COMPILING."

HEY! GET BACK TO WORK!

COMPILING!

OH. CARRY ON.
Can we solve this problem?

- Consider the following program (input underlined):

  How many days' temperatures? 7
  Day 1's high temp: 45
  Day 2's high temp: 44
  Day 3's high temp: 39
  Day 4's high temp: 48
  Day 5's high temp: 37
  Day 6's high temp: 46
  Day 7's high temp: 53
  Average temp = 44.6
  4 days were above average.
Why the problem is hard

• We need each input value twice:
  • to compute the average (a cumulative sum)
  • to count how many were above average

• We could read each value into a variable… but we:
  • don't know how many days are needed until the program runs
  • don't know how many variables to declare

• We need a way to declare many variables in one step.
Arrays

- **array**: object that stores many values of the same type.
  - **element**: One value in an array.
  - **index**: A 0-based integer to access an element from an array.

```
index 0 1 2 3 4 5 6 7 8 9
value 12 49 -2 26 5 17 -6 84 72 3

element 0 element 4 element 9
```
Array declaration

\[
\text{type[\[ \]] \ name = new \ type[\{length\}];}
\]

- Example:
  \[
  \text{int[] numbers = new int[10];}
  \]

<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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Array declaration, cont.

- The length can be any integer expression.

  ```java
  int x = 2 * 3 + 1;
  int[] data = new int[x % 5 + 2];
  ```

- Each element initially gets a "zero-equivalent" value.

<table>
<thead>
<tr>
<th>Type</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>0</td>
</tr>
<tr>
<td>double</td>
<td>0.0</td>
</tr>
<tr>
<td>boolean</td>
<td>false</td>
</tr>
<tr>
<td>String or other object</td>
<td>null</td>
</tr>
<tr>
<td></td>
<td>(means, &quot;no object&quot;)</td>
</tr>
</tbody>
</table>
Accessing elements

name[index]  // access
name[index] = value;  // modify

- Example:

```java
numbers[0] = 27;
numbers[3] = -6;
System.out.println(numbers[0]);
if (numbers[3] < 0) {
    System.out.println("Element 3 is negative.");
}
```

<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>27</td>
<td>0</td>
<td>0</td>
<td>-6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Accessing array elements

```java
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 99;
numbers[6] = 2;

int x = numbers[1];
numbers[x] = 42;
numbers[numbers[6]] = 11; // use numbers[6] as index
```

<table>
<thead>
<tr>
<th>inde</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>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>3</td>
<td>11</td>
<td>42</td>
<td>99</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

```
Arrays of other types

double[] results = new double[5];
results[2] = 3.4;
results[4] = -0.5;

boolean[] tests = new boolean[6];
tests[3] = true;
Out-of-bounds

- Legal indexes: between 0 and 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[9]);       // okay
  System.out.println(data[-1]);      // exception
  System.out.println(data[10]);      // exception
  ```

<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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Arrays and for loops

- It is common to use for loops to access array elements.

```java
for (int i = 0; i < 8; i++) {
    System.out.print(numbers[i] + " ");
}
System.out.println();  // output: 0 4 11 0 44 0 0 2
```

- Sometimes we assign each element a value in a loop.

```java
for (int i = 0; i < 8; i++) {
    numbers[i] = 2 * i;
}
```

<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>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>
The length field

- An array's length field stores its number of elements.

```
name.length

for (int i = 0; i < numbers.length; i++) {
    System.out.print(numbers[i] + " ");
}
// output: 0 2 4 6 8 10 12 14
```

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

- What expressions refer to:
  - The last element of any array?
  - The middle element?
Weather question

- Use an array to solve the weather problem:

  How many days' temperatures? 7
  Day 1's high temp: 45
  Day 2's high temp: 44
  Day 3's high temp: 39
  Day 4's high temp: 48
  Day 5's high temp: 37
  Day 6's high temp: 46
  Day 7's high temp: 53
  Average temp = 44.6
  4 days were above average.
Weather answer

// Reads temperatures from the user, computes average and # days above average.
import java.util.*;

public class Weather {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("How many days' temperatures? ");
        int days = console.nextInt();

        int[] temps = new int[days]; // array to store days' temperatures
        int sum = 0;

        for (int i = 0; i < days; i++) { // read/store each day's temperature
            System.out.print("Day "+(i + 1)+"'s high temp: ");
            temps[i] = console.nextInt();
            sum += temps[i];
        }

        double average = (double) sum / days; // see if each day is above average
        int count = 0;

        for (int i = 0; i < days; i++) {
            if (temps[i] > average) {
                count++;
            }
        }

        // report results
        System.out.printf("Average temp = %.1f\n", average);
        System.out.println(count + " days above average");
    }
}
Quick array initialization

type[] name = {value, value, ... value};

- Example:
  int[] numbers = {12, 49, -2, 26, 5, 17, -6};

  \[\begin{array}{cccccccc}
  index & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
  value & 12 & 49 & -2 & 26 & 5 & 17 & -6 \\
  \end{array}\]

- Useful when you know what the array's elements will be
- The compiler figures out the size by counting the values
Limitations of arrays

- You cannot resize an existing array:
  ```java
  int[] a = new int[4];
  a.length = 10; // error
  ```

- You cannot compare arrays with `==` or `equals`:
  ```java
  int[] a1 = {42, -7, 1, 15};
  int[] a2 = {42, -7, 1, 15};
  if (a1 == a2) { ... } // false!
  if (a1.equals(a2)) { ... } // false!
  ```

- An array does not know how to print itself:
  ```java
  int[] a1 = {42, -7, 1, 15};
  System.out.println(a1); // [I@98f8c4]
# The Arrays class

- **Class Arrays** in package `java.util` has 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 a <em>sorted</em> array (or &lt; 0 if not found)</td>
</tr>
<tr>
<td><code>copyOf(array, length)</code></td>
<td>returns a new copy of an array</td>
</tr>
<tr>
<td><code>equals(array1, array2)</code></td>
<td>returns <code>true</code> if the two arrays contain same elements in the same order</td>
</tr>
<tr>
<td><code>fill(array, value)</code></td>
<td>sets every element to the given value</td>
</tr>
<tr>
<td><code>sort(array)</code></td>
<td>arranges the elements into sorted order</td>
</tr>
<tr>
<td><code>toString(array)</code></td>
<td>returns a string representing the array, such as &quot;[10, 30, -25, 17]&quot;</td>
</tr>
</tbody>
</table>

- **Syntax:** `Arrays.methodName(parameters)`

---

Copyrigh 2010 by Pearson Education
Arrays.toString

- Arrays.toString accepts an array as a parameter and returns a String representation of its elements.

```java
int[] e = {0, 2, 4, 6, 8};
System.out.println("e is " + Arrays.toString(e));
```

Output:
e is [0, 14, 4, 6, 8]

- Must import java.util.*;
Weather question 2

- Modify the weather program to print the following output:

How many days' temperatures? 7
Day 1's high temp: 45
Day 2's high temp: 44
Day 3's high temp: 39
Day 4's high temp: 48
Day 5's high temp: 37
Day 6's high temp: 46
Day 7's high temp: 53
Average temp = 44.6
4 days were above average.

Temperatures: [45, 44, 39, 48, 37, 46, 53]
Two coldest days: 37, 39
Two hottest days: 53, 48
Weather answer 2

// Reads temperatures from the user, computes average and # days above average.
import java.util.*;

public class Weather2 {
    public static void main(String[] args) {
        ...  
        int[] temps = new int[days];             // array to store days' temperatures
        ...  (same as Weather program)

        // report results
        System.out.printf("Average temp = %.1f\n", average);
        System.out.println(count + " days above average");

        System.out.println("Temperatures: " + Arrays.toString(temps));
        Arrays.sort(temps);
        System.out.println("Two coldest days: " + temps[0] + ", " + temps[1]);
        System.out.println("Two hottest days: " + temps[temps.length - 1] + ", " + temps[temps.length - 2]);
    }
}
"Array mystery" problem

- **traversal**: An examination of each element of an array.

What element values are stored in the following array?

```java
int[] a = {1, 7, 5, 6, 4, 14, 11};
for (int i = 0; i < a.length - 1; i++) {
    if (a[i] > a[i + 1]) {
        a[i + 1] = a[i + 1] * 2;
    }
}
```

<table>
<thead>
<tr>
<th>index</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>22</td>
</tr>
</tbody>
</table>
Why arrays are useful

- Arrays store a large amount of data accessible from one variable.

- Arrays help us group related data into elements.

- Arrays let us access data in random order.
  - Cassette tape vs. DVD