Building Java Programs

Chapter 7
Lecture 7-1: Arrays

**reading: 7.1**
self-checks: #1-9
videos: Ch. 7 #4
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.

<table>
<thead>
<tr>
<th>index</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>49</td>
</tr>
<tr>
<td>2</td>
<td>-2</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>-6</td>
</tr>
<tr>
<td>7</td>
<td>84</td>
</tr>
<tr>
<td>8</td>
<td>72</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

*element 0* *element 4* *element 9*
Array declaration

```java
type[] name = new type[length];
```

- Example:
  ```java
  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 (means, &quot;no object&quot;)</td>
</tr>
</tbody>
</table>
### Accessing elements

```java
name[index] = value;    // modify
```

**Example:**

```java
define numbers as array of 4 integers

numbers[0] = 27;
numbers[3] = -6;

System.out.println("value of numbers [0]: ", 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>
```
Arrays of other types

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

<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>0.0</td>
<td>0.0</td>
<td>3.4</td>
<td>0.0</td>
<td>-0.5</td>
</tr>
</tbody>
</table>

boolean[] tests = new boolean[6];
tests[3] = true;

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>
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>
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>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>numbers value</td>
<td>0</td>
<td>4</td>
<td>11</td>
<td>42</td>
<td>99</td>
<td>0</td>
<td>2</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.

```java
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.
// 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[] temperatures = 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: ");
            temperatures[i] = console.nextInt();
            sum += temperatures[i];
        }
        double average = (double) sum / days;

        int count = 0; // see if each day is above average
        for (int i = 0; i < days; i++) {
            if (temperatures[i] > average) {
                count++;
            }
        }

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

reading: 7.1
self-checks: #8
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 229231007 contains: two 0s, one 1, three 2s, one 7, and one 9.

- We could declare 10 counter variables for this...

  ```c
  int counter0, counter1, counter2, counter3, counter4,
  counter5, counter6, counter7, counter8, counter9;
  ```

- Yuck!
A multi-counter problem

• A better solution is to use an array of size 10.
  • The element at index $i$ will store the counter for digit value $i$.
  • For integer value 229231007, our array should store:

  \[
  \begin{array}{cccccccccc}
  \text{index} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
  \hline
  \text{value} & 2 & 1 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
  \end{array}
  \]

• The index at which a value is stored has meaning.
  • Sometimes it doesn't matter.
  • What about the weather case?
Creating an array of tallies

```java
int num = 229231007;
int[] counts = new int[10];
while (num > 0) {
    // pluck off a digit and add to proper counter
    int digit = num % 10;
    counts[digit]++;
    num = num / 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>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Array histogram question

- Given a file of integer exam scores, such as:
  82
  66
  79
  63
  83

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

- 85: *****
- 86: ************
- 87: ***
- 88: *
- 91: ****
Histogram variations

- Curve the scores; add a fixed number to each score. (But don't allow a curved score to exceed the max of 101.)

- Chart the data with a DrawingPanel.
  - window is 100px tall
  - 2px between each bar
  - 10px tall bar for each student who earned that score
public static final int CURVE = 5;         // adjustment to each exam score

public class Histogram {              // Reads an input file of test scores (integers) and displays a
  public static final int CURVE = 5;  // graphical histogram of the score distribution.

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

public class Histogram {
    public static final int CURVE = 5;         // adjustment to each exam score

    public static void main(String[] args) throws FileNotFoundException {
        Scanner input = new Scanner(new File("midterm.txt"));
        int[] counts = new int[101];              // counters of test scores 0 - 100

        while (input.hasNextInt()) {              // read file into counts array
            int score = input.nextInt();
            score = Math.min(score + CURVE, 100);  // curve the exam score
            counts[score]++;                         // if score is 87, then counts[87]++
        }

        for (int i = 0; i < counts.length; i++) {  // print star histogram
            if (counts[i] > 0) {
                System.out.print(i + " : ");
                for (int j = 0; j < counts[i]; j++) {
                    System.out.print("*");
                }
                System.out.println();
            }
        }
    }
}
Array histogram solution 2

... 

// use a DrawingPanel to draw the histogram
DrawingPanel p = new DrawingPanel(counts.length * 3 + 6, 200);
Graphics g = p.getGraphics();
g.setColor(Color.BLACK);
for (int i = 0; i < counts.length; i++) {
    g.drawLine(i * 3 + 3, 175, i * 3 + 3, 175 - 5 * counts[i]);
}