Building Java Programs

Chapter 7
Lecture 7-2: Tallying and Traversing Arrays

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

<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>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

• 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>1</td>
<td>0</td>
<td>1</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
// Reads an input file of test scores (integers) and displays a 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]);
}
Array traversals, text processing

reading: 7.1, 4.4

self-check: Ch. 7 #8, Ch. 4 #19-23
Array traversals

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

  ```java
  for (int i = 0; i < array.length; i++) {
    do something with array[i];
  }
  ```

- **Examples**:
  - printing the elements
  - searching for a specific value
  - rearranging the elements
  - computing the sum, product, etc.
Quick array initialization

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

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

<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>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>12</td>
<td>49</td>
<td>-2</td>
<td>26</td>
<td>5</td>
<td>17</td>
<td>-6</td>
</tr>
</tbody>
</table>

- Useful when you know what the array's elements will be
- The compiler figures out the size by counting the values
"Array mystery" problem

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>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>1</td>
<td>7</td>
<td>10</td>
<td>12</td>
<td>8</td>
<td>14</td>
<td>22</td>
</tr>
</tbody>
</table>
Text processing

- **text processing**: Examining, editing, formatting text.
  - Often involves for loops to examine each letter of a String.
    - Count the number of times the letter 's' occurs in a file.
    - Find which letter is most common in a file.
    - Count A, C, T and Gs in Strings representing DNA strands.

- Strings are represented internally as arrays of `char`.

```java
String str = "Ali G."
```

```
<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>'A'</td>
<td>'l'</td>
<td>'i'</td>
<td></td>
<td></td>
<td>'G'</td>
</tr>
</tbody>
</table>
```
Recall: type `char`

- **`char`**: A primitive type representing a single character.
  - Values are surrounded with apostrophes: 'a' or '4' or '

- Access a string's characters with its `charAt` method.

```java
String word = console.next();
char firstLetter = word.charAt(0);
if (firstLetter == 'c') {
    System.out.println("That's good enough for me!");
}
```

- Use `for` loops to examine each character.

```java
String coolMajor = "CSE";
for (int i = 0; i < coolMajor.length(); i++) {
    System.out.println(coolMajor.charAt(i));
}
```
Text processing question

- Write a method `tallyVotes` that accepts a `String` parameter and prints the number of McCain, Obama and independent voters.

```java
// (M)cCain, (O)bama, (I)ndependent
String voteText = "M000000M000000O000000MOMMIMOMMIMOMMIO";
tallyVotes(voteText);
```

- **Output:**
  Votes: [16, 14, 3]
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.*;
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>binarySearch(array, value)</td>
<td>returns the index of the given value in a sorted array (&lt; 0 if not found)</td>
</tr>
<tr>
<td>equals(array1, array2)</td>
<td>returns true if the two arrays contain the same elements in the same order</td>
</tr>
<tr>
<td>fill(array, value)</td>
<td>sets every element in the array to have the given value</td>
</tr>
<tr>
<td>sort(array)</td>
<td>arranges the elements in the array into ascending order</td>
</tr>
<tr>
<td>toString(array)</td>
<td>returns a string representing the array, such as &quot;[10, 30, 17]&quot;</td>
</tr>
</tbody>
</table>
public static int[] tallyVotes(String votes) {
    int[] tallies = new int[3]; // M -> 0, O -> 1, I -> 2

    for(int i = 0; i < votes.length(); i++) {
        if(votes.charAt(i) == 'M') {
            tallies[0]++;
        } else if(votes.charAt(i) == 'O') {
            tallies[1]++;
        } else { // votes.charAt(i) == 'I'
            tallies[2]++;
        }
    }
    System.out.println("Votes: " + Arrays.toString(tally));
}