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:

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
value	2	1	3	0	0	0	0	1	0	1

- 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

```
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;
}
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
value	2	1	3	0	0	0	0	1	0	1

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: ****
```

Array histogram answer

```
// Reads an input file of test scores (integers) and displays a
// text histogram of the score distribution.
import java.io.*;
import java.util.*;

public class Histogram {

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

        while (input.hasNextInt()) {     // read file into counts array
            int score = input.nextInt();
            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 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.

```
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};
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*
value	12	49	-2	26	5	17	-6

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

Mini-exercise

- Improve the following code (it can be replaced by 1 line):

```
int[] ns = new int[4];
ns[1] = 10;
ns[2] = 25;
ns[3] = 50;
```

(This is slightly a trick question. But only slightly.)

Mini-exercise - solution

- Improve the following code (it can be replaced by 1 line):

```
int[] ns = new int[4];
ns[1] = 10;
ns[2] = 25;
ns[3] = 50;
```

```
// new code:
int[] ns = {0, 10, 25, 50};
```

"Array mystery" problem

- What element values are stored in the following array?

```
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;
    }
}
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*
value	1	7	10	12	8	14	22

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 `String`s representing DNA strands.

- `String`s are represented internally as arrays of `char`.

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

index	*0*	*1*	*2*	*3*	*4*	*5*
value	`'A'`	`'l'`	`'i'`	`' '`	`'G'`	`'.'`

Recall: type `char`

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

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

```
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.

```
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.

```
// (M)cCain, (O)bama, (I)ndependent
String voteText = "MOOOOOOMMMMMOOOOOOMOMMIMOMMIMOMMIO";
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.

```
int[] e = {0, 2, 4, 6, 8};
e[1] = e[3] + e[4];
System.out.println("e is " + Arrays.toString(e));
```

Output:

```
e is [0, 14, 4, 6, 8]
```

- Must `import java.util.*;`

Text processing answer

```
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));;
}
```

The `Arrays` class

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

Method name	**Description**
`binarySearch(`**array**`, `**value**`)`	returns the index of the given value in a sorted array (< 0 if not found)
`equals(`**array1**`, `**array2**`)`	returns `true` if the two arrays contain the same elements in the same order
`fill(`**array**`, `**value**`)`	sets every element in the array to have the given value
`sort(`**array**`)`	arranges the elements in the array into ascending order
`toString(`**array**`)`	returns a string representing the array, such as `"[10, 30, 17]"`

Arrays as parameters

- [Section 7.1 of the text]

- Declaration:

  ```
  public static type methodName(type[] name) {
  ```

  - Example:

    ```
    public static double average(int[] numbers) {
    ```

- Call:

  ```
  methodName(arrayName);
  ```

  - Example:

    ```
    int[] scores = {13, 17, 12, 15, 11};
    double avg = average(scores);
    ```

Array parameter example

```
public static void main(String[] args) {
    int[] iq = {126, 84, 149, 167, 95};
    double avg = average(iq);
    System.out.println("Average = " + avg);
}

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

Output:

```
Average = 124.2
```

Mini-exercise

Modify the ‘average’ method to find the max element instead (assume the array is non-empty)

```
  public static void main(String[] args) {
    int[] iq = {126, 84, 149, 167, 95};
    double avg = average(iq);
    System.out.println("Average = " + avg);
}

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

Mini-exercise - answer

```
  public static void main(String[] args) {
    int[] iq = {126, 84, 149, 167, 95};
    int m = max(iq);
    System.out.println(”max = " + m);
}

public static int max(int[] array) {
    int maxSoFar = array[0];
    for (int i = 1; i < array.length; i++) {
        if (array[i]>maxSoFar) {
            maxSoFar = array[i];
        }
    }
    return maxSoFar;
}
```

Output:

```
Max = 167
```

Arrays as return (declaring)

```
public static type[] methodName(parameters) {
```

- Example:

```
public static int[] countDigits(int n) {
    int[] counts = new int[10];
    while (n > 0) {
        int digit = n % 10;
        n = n / 10;
        counts[digit]++;
    }
    return counts;
}
```

Arrays as return (calling)

type`[]` **name** `=` **methodName**`(`**parameters**`);`

- Example:

```
public static void main(String[] args) {
    int[] tally = countDigits(229231007);
    System.out.println(Arrays.toString(tally));
}
```

Output:

```
[2, 1, 3, 1, 0, 0, 0, 1, 0, 1]
```

Section attendance question

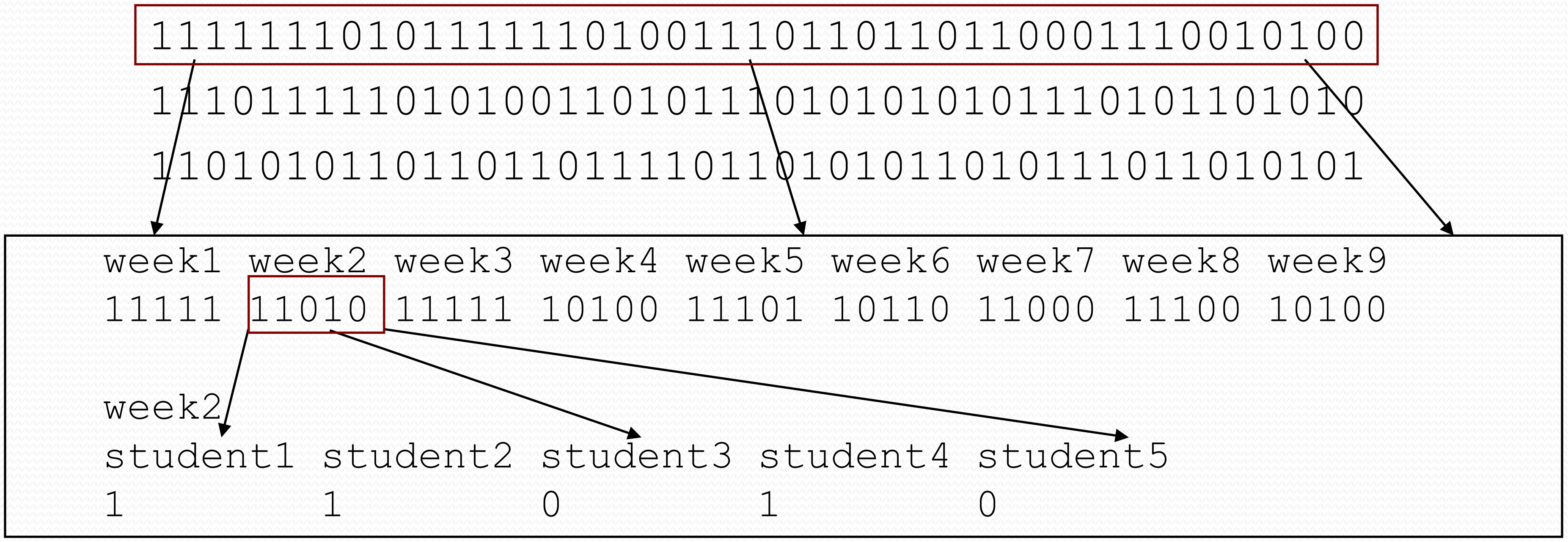
- Write a program that reads a data file of section attendance and produces the following output:

```
Sections attended: [9, 6, 7, 4, 3]
Student scores: [20, 18, 20, 12, 9]
Student grades: [100.0, 90.0, 100.0, 60.0, 45.0]

Sections attended: [6, 7, 5, 6, 4]
Student scores: [18, 20, 15, 18, 12]
Student grades: [90.0, 100.0, 75.0, 90.0, 60.0]

Sections attended: [5, 6, 5, 7, 6]
Student scores: [15, 18, 15, 20, 18]
Student grades: [75.0, 90.0, 75.0, 100.0, 90.0]
```

  - Students earn 3 points for each section attended up to 20.

Section input file

- The input file contains section attendance data:



```
111111101011111101001110110110110001110010100
111011111010100110101110101010101110101101010
110101011011011011110110101011010111011010101
```

```
week1 week2 week3 week4 week5 week6 week7 week8 week9
11111 11010 11111 10100 11101 10110 11000 11100 10100

week2
student1 student2 student3 student4 student5
1        1        0        1        0
```

- Each line represents a section (5 students, 9 weeks).
    - 1 means the student attended; 0 not.

Data transformations

- In this problem we go from 0s and 1s to student grades
  - This is called *transforming* the data.
  - Often each transformation is stored in its own array.

- We must map between the data and array indexes.

  Examples:

  - by position (store the *i* th value we read at index *i*)
  - tally (if input value is *i*, store it at array index *i*)
  - explicit mapping (count `'M'` at index 0, count `'O'` at index 1)

Array param/return answer

```
// This program reads a file representing which students attended
// which discussion sections and produces output of the students'
// section attendance and scores.

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

public class Sections {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner input = new Scanner(new File("sections.txt"));
        while (input.hasNextLine()) {
            // process one section
            String line = input.nextLine();
            int[] attended = countAttended(line);
            int[] points = computePoints(attended);
            double[] grades = computeGrades(points);
            results(attended, points, grades);
        }
    }

    // Produces all output about a particular section.
    public static void results(int[] attended, int[] points, double[] grades) {
        System.out.println("Sections attended: " + Arrays.toString(attended));
        System.out.println("Sections scores: " + Arrays.toString(points));
        System.out.println("Sections grades: " + Arrays.toString(grades));
        System.out.println();
    }

    ...
```

Array param/return answer

```
    ...
    // Counts the sections attended by each student for a particular section.
    public static int[] countAttended(String line) {
        int[] attended = new int[5];
        for (int i = 0; i < line.length(); i++) {
            char c = line.charAt(i);
            // c == '1'  or  c == '0'
            if (c == '1') {
                // student attended their section
                attended[i % 5]++;
            }
        }
        return attended;
    }

    // Computes the points earned for each student for a particular section.
    public static int[] computePoints(int[] attended) {
        int[] points = new int[5];
        for (int i = 0; i < attended.length; i++) {
            points[i] = Math.min(20, 3 * attended[i]);
        }
        return points;
    }

    // Computes the percentage for each student for a particular section.
    public static double[] computeGrades(int[] points) {
        double[] grades = new double[5];
        for (int i = 0; i < points.length; i++) {
            grades[i] = 100.0 * points[i] / 20.0;
        }
        return grades;
    }
}
```