Building Java Programs

Chapter 7
Lecture 7-3: Arrays for Tallying; Text Processing

reading: 7.6, 4.3

A multi-counter problem

- Problem: Write a method `mostFrequentDigit` that returns the digit value that occurs most frequently in a number.

  - Example: The number 669260267 contains:
                one 0, two 2s, four 6es, one 7, and one 9.
    `mostFrequentDigit(669260267)` returns 6.

  - If there is a tie, return the digit with the lower value.
    `mostFrequentDigit(57135203)`   returns 3.

A multi-counter problem

- We could declare 10 counter variables …

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

- A better solution is to use an array of size 10.
  - The element at index i will store the counter for digit value i.
  - Example for 669260267:

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

Creating an array of tallies

```
// n is an int, e.g. 669260267
int[] counts = new int[10];
while (n > 0) {
    // pluck off a digit and add to proper counter
    int digit = n % 10;
    counts[digit]++;
    n = n / 10;
}
```

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

Tally solution

```java
// Returns the digit value that occurs most frequently in n.
// Breaks ties by choosing the smaller value.
public static int mostFrequentDigit(int n) {
    int[] counts = new int[10];
    while (n > 0) {
        int digit = n % 10;   // pluck off a digit and tally it
        counts[digit]++;
        n = n / 10;
    }

    // find the most frequently occurring digit
    int bestIndex = 0;
    for (int i = 1; i < counts.length; i++) {
        if (counts[i] > counts[bestIndex]) {
            bestIndex = i;
        }
    }

    return bestIndex;
}
```

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

```java
// Reads a file of test scores and shows a 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("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();
            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();
            }
        }
    }
}
```
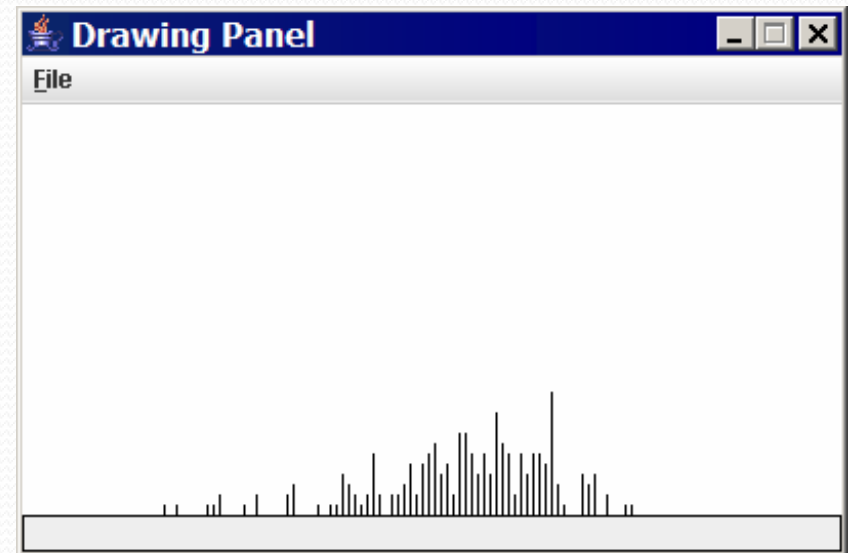
Histogram exercise variation

- 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`.

Histogram: Solution

```
...

// 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]);
}

...
```

Text processing

reading: 4.3

String traversals

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

index	0	1	2	3	4	5	6
value	'l'	'e'	't'	't'	'e'	'r'	's'

- We can write algorithms to traverse strings to compute information.

- What useful information might the following string have?

  `"IDRIRRIDRRIDMIDIRRRIRIRIIDIDDRDDRRDIDIID"`

Down with the Marty Party!

```
// string stores voters' votes
// (R)EPUBLICAN, (D)EMOCRAT, (I)NDEPENDENT, (M)ARTY
String votes = "IDRIRRIDRRIDMIDIRRRIRIRIIDIDDRDDRRDIDIID";
int[] counts = new int[4]; // R -> 0, D -> 1, I -> 2, M -> 3
for (int i = 0; i < votes.length(); i++) {
    char c = votes.charAt(i);
    if (c == 'R') {
        counts[0]++;
    } else if (c == 'D') {
        counts[1]++;
    } else if (c == 'I') {
        counts[2]++;
    } else {   // c == 'M'
        counts[3]++;
    }
}
System.out.println(Arrays.toString(counts));
```

Output:

```
[13, 12, 14, 1]
```

Section attendance question

- Read a file of section attendance (*see next slide*):

```
yynyyynayayynyyyayanyyyaynayyayyanayyyanyayna
ayyanyyyyayanaayyanayyyananayayaynyayayynynya
yyayaynyyayyanynnyyyayyanayaynannnyyayyayayny
```

- And produce the following output:

```
Section 1
Student points: [20, 16, 17, 14, 11]
Student grades: [100.0, 80.0, 85.0, 70.0, 55.0]

Section 2
Student points: [16, 19, 14, 14, 8]
Student grades: [80.0, 95.0, 70.0, 70.0, 40.0]

Section 3
Student points: [16, 15, 16, 18, 14]
Student grades: [80.0, 75.0, 80.0, 90.0, 70.0]
```

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

Section input file

student		123451234512345123451234512345123451234512345
week		1 2 3 4 5 6 7 8 9
section	1	yynyyynayayynyyyayanyyyaynayyayyanayyyanyayna
section	2	ayyanyyyyayanaayyanayyyananayayaynyayayynynya
section	3	yyayaynyyayyanynnyyyayyanayaynannnyyayyayayny

- Each line represents a section.
- A line consists of 9 weeks' worth of data.
  - Each week has 5 characters because there are 5 students.
- Within each week, each character represents one student.
  - `a` means the student was absent   (+0 points)
  - `n` means they attended but didn't do the problems (+1 points)
  - `y` means they attended and did the problems   (+3 points)

Section attendance answer 1

```java
// 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"));
        int section = 1;
        while (input.hasNextLine()) {
            // process one section
            String line = input.nextLine();
            int[] points = countPoints(line);
            double[] grades = computeGrades(points);
            results(section, points, grades);
            section++;
        }
    }

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

    ...
```

Section attendance answer 2

```java
    ...
    // Computes the points earned for each student for a particular section.
    public static int[] countPoints(String line) {
        int[] points = new int[5];
        for (int i = 0; i < line.length(); i++) {
            int earned = 0;
            char c = line.charAt(i);
            if (c == 'y') {   // c == 'y' or 'n' or 'a'
                earned = 3;
            } else if (c == 'n') {
                earned = 1;
            }
            int student = i % 5;
            points[student] = Math.min(20, points[student] + earned);
        }
        return points;
    }

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

Data transformations

- In many problems we transform data between forms.
  - Example: digits $\rightarrow$ count of each digit $\rightarrow$ most frequent digit
  - Often each transformation is computed/stored as an array.
  - For structure, a transformation is often put in its own method.

- Sometimes we map between data and array indexes.

  - 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 `'J'` at index 0, count `'X'` at index 1)