

Topic 23

arrays - part 3 (tallying, text processing)

"42 million of *anything* is a lot."

-Doug Burger

(commenting on the number of transistors in the Pentium IV processor)



► What is output when method clicker2 is called?

```
public static void clicker2() {  
    int[] values = {1, 2};  
    arrayManip(values);  
    System.out.print(Arrays.toString(values));  
}
```

```
public static void arrayManip(int[] values) {  
    values[1] += 2;  
    values[0] -= 2;  
    System.out.print(Arrays.toString(values));  
    values = new int[3];  
    System.out.print(Arrays.toString(values));  
}
```

- A. [1, 2] [0, 0, 0] [1, 2]
- B. [1, 2] [1, 2] [1, 2]
- C. [-1, 4] [0, 0, 0] [0, 0, 0]
- D. [-1, 4] [0, 0, 0] [1, 2]
- E. [-1, 4] [0, 0, 0] [-1, 4]

A multi-counter problem

- ▶ Problem: Write a method `mostFrequentDigit` that returns the digit 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;
```

- ▶ But 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:

<i>index</i>	0	1	2	3	4	5	6	7	8	9
<i>value</i>	1	0	2	0	0	0	4	1	0	0

- How do we build such an array? And how does it help?

Creating an array of tallies

```
// assume n = 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

<i>value</i>	1	0	2	0	0	0	4	1	0	0
--------------	---	---	---	---	---	---	---	---	---	---

Tally solution

```
// 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;  
}
```

Tally Problem

- ▶ Write a method to pick random numbers from 0 to 99.
- ▶ A parameters specifies the number of random numbers to pick
- ▶ The method returns the difference between the number of times the most and least picked number
- ▶ With 1,000,000 numbers what do you expect the difference to be?

A. 0 B. 1 - 10 C. 11 - 100
D. 101 - 1000 E. more than 1001

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 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();
            }
        }
    }
}
```

Text processing

reading: 4.3

Type char

- **char** : A primitive type representing single characters.
 - A `String` is stored internally as an array of `char`

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

<i>index</i>	0	1	2	3	4	5
<i>value</i>	'A'	'l'	'i'	' '	'G'	'.'

- It is legal to have variables, parameters, returns of type `char`
 - surrounded with apostrophes: `'a'` or `'4'` or `'\n'` or `'\''`

```
char letter = 'T';
```

```
System.out.println(letter);
```

```
// T
```

```
System.out.println(letter + "exas!");
```

```
// Texas!
```

The charAt method

- The chars in a String can be accessed using the charAt method.
 - accepts an int index parameter and returns the char at that index

```
String food = "cookie";  
char firstLetter = food.charAt(0);    // 'c'  
System.out.println(firstLetter + " is for " + food);
```

- You can use a for loop to print or examine each character.

```
String major = "CS!";  
for (int i = 0; i < major.length(); i++) {  
    char c = major.charAt(i);  
    System.out.println(c);  
}  
// output:  
// C  
// S  
// !
```

Comparing char values

- ▶ You can compare chars with `==`, `!=`, and other operators:

```
String word = console.next();
char last = word.charAt(word.length() - 1);
if (last == 's') {
    System.out.println(word + " is
plural.");
}
```

```
// prints the alphabet
```

```
for (char c = 'a'; c <= 'z'; c++) {
    System.out.print(c);
}
```

char vs. int

- ▶ Each `char` is mapped to an integer value internally
 - Called an **ASCII value**

'A' is 65

'B' is 66

' ' is 32

'a' is 97

'b' is 98

'*' is 42

- Mixing `char` and `int` causes automatic conversion to `int`.

'a' + 10 is 107,
is 130

'A' + 'A'

- To convert an `int` into the equivalent `char`, type-cast it.

(char) ('a' + 2) is 'c'

char VS. String

- ▶ "h" is a String, but 'h' is a char (they are different)

- ▶ A String is an object; it contains methods.

```
String s = "h";  
s = s.toUpperCase();           // "H"  
int len = s.length();         // 1  
char first = s.charAt(0);      // 'H'
```

- ▶ A char is primitive; you can't call methods on it.

```
char c = 'h';  
c = c.toUpperCase();           // ERROR  
s = s.charAt(0).toUpperCase(); // ERROR
```

- What is `s + 1` ? What is `c + 1` ?
- What is `s + s` ? What is `c + c` ?

String traversals

- ▶ We can write algorithms to traverse strings to compute information.
- ▶ What useful information might the following string have?

"GDRGRRGDRRGDLGDGRRRRGRGRGGDGDDRDRRDGDGGD"

Data takes many forms

```
// string stores voters' votes
// (R)EPUBLICAN, (D)EMOCRAT, (G)REEN, (L)IBERTARIAN
String votes =
"GDRGRRGDRRGDLGDGRRRRGRGRGGDGDDRDDRRDGDGGD";
int[] counts = new int[4]; // R -> 0, D -> 1, G -> 2, L -> 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 == 'B') {
        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*):

```
yyynyynayayynyayanyaynayyayyanayyyanyayna  
ayyanayyyayanaayyanayyyananayayaynyayayyny  
yyayaynyyayyanynnyyyayyanayaynannnyyayyayny
```

- And produce the following output:

```
Section 1
```

```
Student points: [20, 17, 19, 16, 13]
```

```
Student grades: [100.0, 85.0, 95.0, 80.0, 65.0]
```

```
Section 2
```

```
Student points: [17, 20, 16, 16, 10]
```

```
Student grades: [85.0, 100.0, 80.0, 80.0, 50.0]
```

```
Section 3
```

```
Student points: [17, 18, 17, 20, 16]
```

```
Student grades: [85.0, 90.0, 85.0, 100.0, 80.0]
```

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

Section input file

student		1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5												
week		1	2	3	4	5	6	7	8	9																																	
section	1	y	y	n	y	y	n	a	y	a	y	n	y	y	y	a	y	a	n	y	y	y	a	y	n	a	y	y	a	y	n	a											
section	2	a	y	y	a	n	y	y	y	y	a	y	a	n	a	a	y	y	a	n	a	y	y	y	a	n	a	n	a	y	a	y	n	y	n	y	a						
section	3	y	y	a	y	a	y	n	y	y	a	y	y	a	n	y	n	n	y	y	y	a	y	y	a	n	a	y	a	y	n	a	n	n	n	y	y	a	y	a	y	n	y

- 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 (+2 points)
 - y means they attended and did the problems (+3 points)

Section attendance answer

```
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()) {
            String line = input.nextLine();           // process one section
            int[] points = new int[5];
            for (int i = 0; i < line.length(); i++) {
                int student = i % 5;
                int earned = 0;
                if (line.charAt(i) == 'y') {           // c == 'y' or 'n'
                    earned = 3;
                } else if (line.charAt(i) == 'n') {
                    earned = 2;
                }
                points[student] = Math.min(20, points[student] + earned);
            }

            double[] grades = new double[5];
            for (int i = 0; i < points.length; i++) {
                grades[i] = 100.0 * points[i] / 20.0;
            }

            System.out.println("Section " + section);
            System.out.println("Student points: " + Arrays.toString(points));
            System.out.println("Student grades: " + Arrays.toString(grades));
            System.out.println();
            section++;
        }
    }
}
```

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)
- ▶ *Exercise:* Modify the Sections program to use static methods that use arrays as parameters and returns.

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 Sections2 {
    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();
    }

    ...
}
```

Array param/return answer

...

// 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 student = i % 5;  
        int earned = 0;  
        if (line.charAt(i) == 'y') {           // c == 'y'   or   c == 'n'  
            earned = 3;  
        } else if (line.charAt(i) == 'n') {  
            earned = 2;  
        }  
        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[5];  
    for (int i = 0; i < points.length; i++) {  
        grades[i] = 100.0 * points[i] / 20.0;  
    }  
    return grades;  
}
```