Building Java Programs

Chapter 7: Arrays

Lecture 7-1: Array basics,
arrays for counting and tallying

Lecture outline

- array basics
  - declaring and initializing an array
  - getting and setting values of elements of an array
  - arrays for counting and tallying

Array basics

reading: 7.1

self-checks: #1-9

A problem we can't solve (yet)

- 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 = 44.57142857142857
4 days were above average.
```

Why the problem is tough

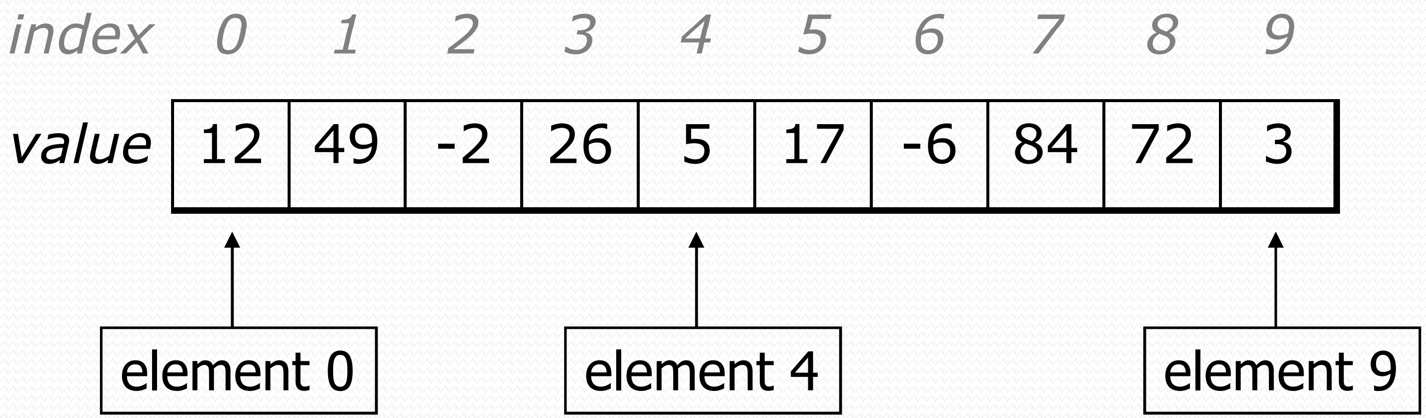
- We need each input value twice:
  - to compute the average (a cumulative sum)
  - to count how many were above average

- 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.



Array declaration

<*type*> `[]` **<*name*>** `= new` **<*type*>** `[` **<*length*>** `];`

- Example:

```
int[] numbers = new int[10];
```

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

Array declaration, cont.

- The length can be any integer expression.

```
int x = 2 * 3 + 1;
int[] data = new int[x % 5 + 2];
```

- Each element initially gets a "zero-equivalent" value.

`int`:	`0`
`double`:	`0.0`
`boolean`:	`false`
object (e.g. `String`):	`null` ("no object")

Assigning array elements

<array name> [***<index>***] = ***<value>*** ;

- Example:

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

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
value	**27**	0	0	**-6**	0	0	0	0	0	0

Accessing array elements

***<array name>* [*<index>*]**

- Example:

```
System.out.println(numbers[0]);
if (numbers[3] < 0) {
    System.out.println("Element 3 is negative.");
}
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
value	**27**	0	0	**-6**	0	0	0	0	0	0

Arrays of other types

- Arrays can contain other types, such as `double`.

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

index	*0*	*1*	*2*	*3*	*4*
value	0.0	0.0	**3.4**	0.0	**-0.5**

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

index	*0*	*1*	*2*	*3*	*4*	*5*
value	false	false	false	**true**	false	false

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:

```
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
```

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

Accessing array elements

- A longer example:

```
int[] numbers = new int[8];
numbers[1] = 4;
numbers[4] = 99;
numbers[7] = 2;

int x = numbers[1];
numbers[x] = 44;
numbers[numbers[7]] = 11;   // numbers[7] as index
```

x | 4 |

numbers								
	0	4	11	0	44	0	0	2

Arrays and for loops

- It's common to use for loops to access array elements.

```
for (int i = 0; i < 8; i++) {
    System.out.print(numbers[i] + " ");
}
System.out.println();
```

  - Output (when used on array from previous slide):

```
0 4 11 0 44 0 0 2
```

Arrays and for loops, cont.

- Sometimes we assign each element a value in a loop.

```
for (int i = 0; i < 8; i++) {
    numbers[i] = 2 * i;
}
```

  - Contents of array?

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

The .length field

- An array's `length` field stores its number of elements.
  - A field is a piece of data stored in an object (see Ch. 8)

- General syntax:

  <array name>`.length`

- It does *not* use parentheses like a String's `.length()`.
  - It is a field, not a method.

- What expressions refer to:
  - The last element of an array?
  - The middle element?

.length field example

- The length field is convenient for establishing loop bounds:

```
for (int i = 0; i < numbers.length; i++) {
    System.out.print(numbers[i] + " ");
}
```

  - Output:
    `0 2 4 6 8 10 12 14`

- Why do we care, since we gave the array a specific length?
  - What if we change the length later?
  - Arrays as parameters?

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 = 44.57142857142857
4 days were above average.
```

Weather answer

```
// This program reads several days' temperatures from the user
// and computes the average and how many days were 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.println("Average temp = " + 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...

```
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*.

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

- Note: 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
```

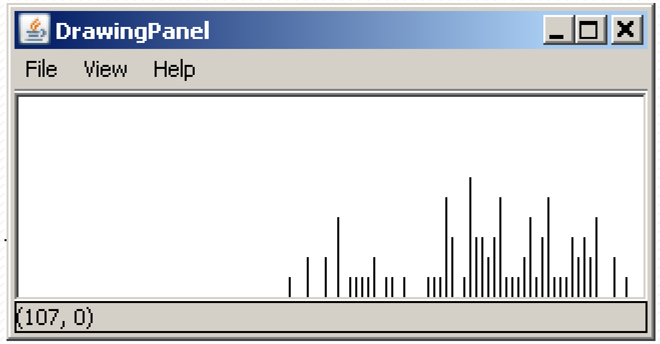
Write a program that will print a histogram of stars indicating the number of students who earned each unique exam score.

```
85: *****
86: ************
87: ***
88: *
91: ****
```

Histogram variations

- Curve 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`.
  - window is 100px tall
  - 2px between each bar
  - 10px for each score



Array histogram answer

```
// 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]);
        }
    }
}
```