Building Java Programs

Chapter 7: Arrays

Lecture 7-3: More text processing, file output

Remember: `charAt` method

- Strings are internally represented as `char` arrays

- String traversals are a common form of data manipulation
  - There is no [] notation for Strings.
  - There is no Scanner for breaking apart a String.
  - We use the `charAt` method

charAt exercise

- Write a method named `count` which accepts a `String` and a `char` as parameters. The method should return the number of times the `char` appears in the `String`.
  - Example:

```
int hCount = count("Oscar the grouch", 'h');
// hCount is 2
```

- Could we also re-write the `replace` method for `Strings`?
  - We can't directly access the array of `chars`
  - How can we build a `String`?

```
String newVerse = replace("eat apples and bananas",
                          'a', 'o');
// newVerse is "eot opples ond bononos"
```

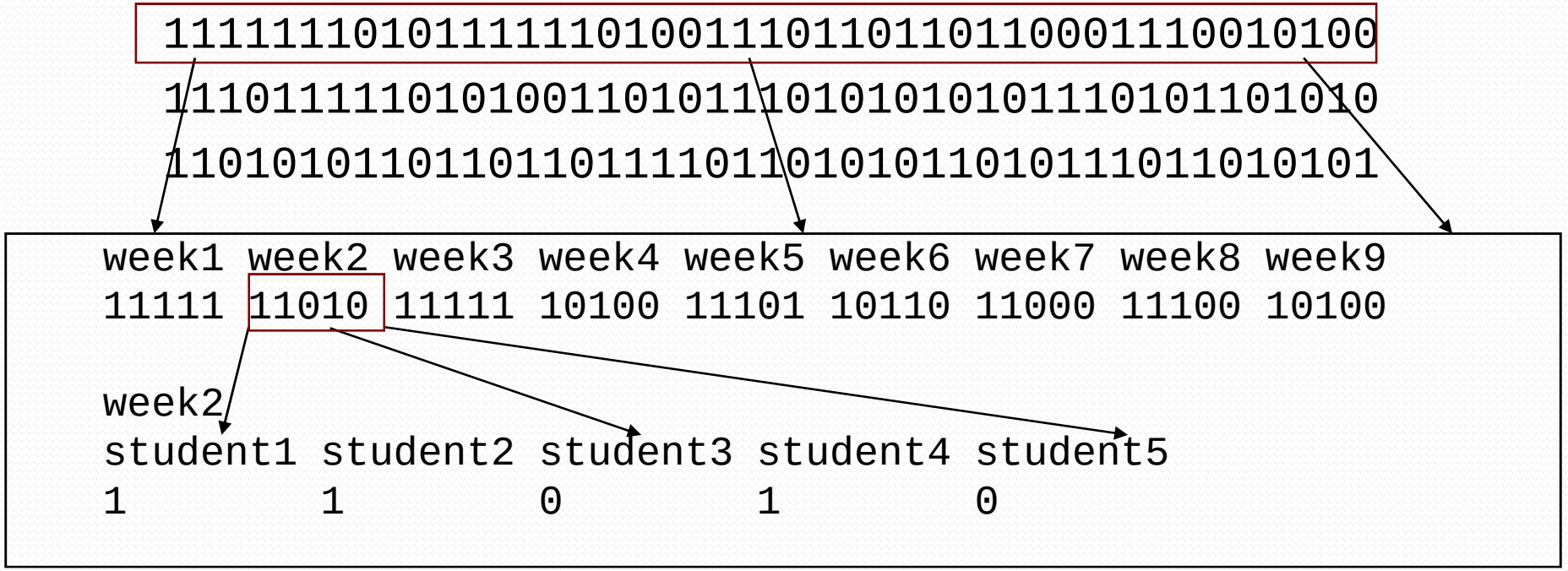
charAt exercise solutions

```
public static int count(String s, char ch) {
    int count = 0;
    for(int i = 0; i < s.length(); i++) {
        if(s.charAt(i) == ch) {
            count++;
        }
    }
    return count;
}

public static String replace(String s, char c1, char c2) {
    String result = "";
    for(int i = 0; i < s.length(); i++) {
        if(s.charAt(i) == c1) {
            result = result + c2;
        } else {
            result = result + s.charAt(i);
        }
    }
    return result;
}
```

Section attendance problem

- Consider an input file of course attendance data:



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

Section attendance problem

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

```
Section #1:
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]

Section #2:
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]

Section #3:
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]
```

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 `'0'` at index 1)

Plan of attack

- This is a complex problem, so let's break it down!
  - We'll start by writing everything in main.
    - Let's just get the section headings, first.
    - Then we can compute sections attended, etc, one at a time.
  - Eventually, the methods we need should be clear.

- Our goal: make main a good program summary.

Section attendance 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();
    }

    ...
```

Section attendance answer 2

```
...
// 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;
}
}
```

File input/output

reading: 6.4 - 6.5

Prompting for a file name

- We can ask the user to tell us the file to read.
  - The file name might have spaces: use `nextLine()`

```
// prompt for the file name
Scanner console = new Scanner(System.in);
System.out.print("Type a file name to use: ");
String filename = console.nextLine();

Scanner input = new Scanner(new File(filename));
```

- What if the user types a file name that does not exist?

Fixing file-not-found issues

- `File` objects have an `exists` method we can use:

```
Scanner console = new Scanner(System.in);
System.out.print("Type a file name to use: ");
String filename = console.nextLine();
File file = new File(filename);

while (!file.exists()) {
    System.out.print("File not found! Try again: ");
    String filename = console.nextLine();
    file = new File(filename);
}
Scanner input = new Scanner(file);  // open the file
```

Output:

```
Type a file name to use: hourz.text
File not found! Try again: h0urz.txt
File not found! Try again: hours.txt
```

Output to files

- **`PrintStream`**: An object in the `java.io` package that lets you print output to a destination such as a file.
  - `System.out` is also a `PrintStream`.
  - Any methods you have used on `System.out` (such as `print`, `println`) will work on every `PrintStream`.

- Do not open a file for reading (`Scanner`) and writing (`PrintStream`) at the same time.
  - You could overwrite your input file by accident!
  - The result can be an empty file (size 0 bytes).

Printing to files, example

- Printing into an output file, general syntax:

```
PrintStream <name> =
            new PrintStream(new File("<file name>"));
  ...
```

  - If the given file does not exist, it is created.
  - If the given file already exists, it is overwritten.

```
PrintStream output = new PrintStream(new File("output.txt"));
output.println("Hello, file!");
output.println("This is a second line of output.");
```

  - Can use similar ideas about prompting for file names here.

PrintStream question

- Modify our previous Sections program to use a `PrintStream` to output to the file `section_output.txt`.

```
Section #1:
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]

Section #2:
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]

Section #3:
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]
```

PrintStream answer

```
// Section attendance
// This version uses a PrintStream for output.

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"));
        PrintStream out = new PrintStream(new File("section_output.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, out);
        }
    }

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