Colin Kincaid
CS 106A

Section #3
July 9, 2018

Section Handout #3: Strings and Files

Portions of this handout by Eric Roberts, Mehran Sahami, Marty Stepp, Patrick Young, and Jeremy Keeshin

1. Adding commas to numeric strings (Chapter 8, Exercise 13, page 290)

When large numbers are written out on paper, it is traditional—at least in the United States—to use commas to separate the digits into groups of three. For example, the number one million is usually written in the following form:

```
1,000,000
```

To make it easier for programmers to display numbers in this fashion, implement a method

```
private String addCommasToNumericString(String digits)
```

that takes a string of decimal digits representing a number and returns the string formed by inserting commas at every third position, starting on the right. For example, if you were to execute the main program

```
public void run() {
   String digits = readLine("Enter a numeric string: ");
   while (digits.length() > 0) {
      println(addCommasToNumericString(digits));
      digits = readLine("Enter a numeric string: ");
   }
}
```

your implementation of the **addCommasToNumericString** method should have the following behavior:

```
addCommasToNumericString("147")          returns          "147"
addCommasToNumericString("2014")         returns        "2,014"
addCommasToNumericString("37897987")     returns   "37,897,987"
```

2. Deleting characters from a string

Write a method

```
private String removeAllOccurrences(String str, char ch)
```

that removes all occurrences of the character **ch** from the string **str**. For example, your method should return the values shown:

```
removeAllOccurrences("This is a test", 't')  returns  "This is a es"
removeAllOccurrences("Summer is here!", 'e') returns  "Summr is hr!"
removeAllOccurrences("---0---", '-')         returns  "0"
```

3. Converting a string to alternating capital letters

Write a method

```
private String altCaps(String str)
```

which converts a string to alternating capital letters, meaning you alternate between uppercase and lowercase. This style of typing was, like, so fly on the internet in the late 90s. For example:

```
altCaps("hello")                    returns "hElLo"
altCaps("CS106A rocks my socks!")   returns "cS106a RoCkS mY sOcKs!"
```

Note that characters that are not letters are not changed and do not affect the alternating sequence of uppercase and lowercase letters.

4. Tracing method execution

For the program below, show what output is produced by the program when it runs, and draw the trace of what each variable contains as the program runs.

```
/*
 * File: ConsoleMystery.java
 * -------------------------
 * This program doesn't do anything useful and exists only to test
 * your understanding of method calls and parameter passing.
 */

import acm.program.*;

public class ConsoleMystery extends ConsoleProgram {

   public void run() {
      ghost(13);
   }


   private void ghost(int x) {
      int y = 0;
      for (int i = 1; i < x; i *= 2) {
         y = witch(y, skeleton(x, i));
      }

      println("ghost: x = " + x + ", y = " + y);
   }


   private int witch(int x, int y) {
      x = 10 * x + y;
      println("witch: x = " + x + ", y = " + y);
      return x;
   }


   private int skeleton(int x, int y) {
      return x / y % 2;
   }
}
```

5. Class Presidents

Somehow, the vote tallies for sophomore and junior class presidents got mixed up. The file format looks like the following:

```
Xavier s 25 Sophie j 12 Ashwin j 44 Isaac s 30 Tyra s 60 Russ
s 23 Aiko j 20
```

The file repeats the pattern **`name year votes`**. The year is either "s" for sophomore or "j" for junior.

Your program should read the file in, output the candidate in each presidential race with the most votes, and how many votes they got.

Specifically, if this file were named `candidates.txt`, your program should output the following:

```
Sophomore Class President: Tyra (60 votes)
Junior Class President: Ashwin (44 votes)
```

Recall the syntax for creating a Scanner from a file:

```
Scanner input = new Scanner(new File("candidates.txt"));
```

6. Pig Latin

Write a method named **`pigLatin`** that accepts as a parameter a Scanner representing an input file. Your method should, preserving line breaks, print out the input file's text in a simplified version of Pig Latin, a (silly) English variant where the first letter of each word is moved to the end. The rules for translating a word to Pig Latin are as follows:

<u>If the word starts with a vowel (a-e-i-o-u), simply append "yay" to the end of the word.</u>

English	Pig Latin
elephant	elephantyay
aardvark	aardvarkyay
easel	easelyay

<u>If the word starts with a consonant, move the consonant to the end, and append "ay".</u>

English	Pig Latin
switch	witchsay
welcome	elcomeway

As an overall example, if the input file `hamilton.txt` contains the following text:

```
i am not
throwing away my shot
```

We might call your method as follows:

```
Scanner input = new Scanner(new File("hamilton.txt"));
pigLatin(input);
```

In this case, your method should print the following console output:

```
iyay amyay otnay
hrowingtay awayyay ymay hotsay
```

To extend this program further, you can optionally implement the more advanced Pig Latin translation rules for words starting with constants. Specifically, if the word starts with a constant, move all constants up to the first vowel (not just the first consonant) to the end, and append "ay".

English	Pig Latin
switch	itchsway
string	ingstray

7. NegativeSum.

Write a method **`negativeSum`** that accepts a Scanner reading from a file containing a series of integers. Your method should print a message to the console indicating whether the sum starting from the first number is ever negative. You should also return true if a negative sum can be reached and false if not. For example, suppose the file contains the following text:

```
38 4 19 -27 -15 -3 4 19 38
```

Your method would consider the sum of just one number (38), the sum of the first two numbers (38 + 4), the sum of the first three numbers (38 + 4 + 19), and so on up to the end. None of these sums is negative, so the method should return false after printing the following message to the console: **`no negative sum`**

Suppose instead that the file contains the following numbers:

```
14 7 -10 9 -18 -10 17 42 98
```

In this case, the method finds that a negative sum (-8) is reached after adding 6 numbers together (14 + 7 + -10 + 9 + -18 + -10). It should return true, indicating that a negative sum can be reached, after printing the following message to the console: **`-8 after 6 steps`**