Chris Piech
CS 106A

Section #4
February 5, 2018

Section Handout #4: String Processing

Portions of this handout by Eric Roberts , 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
```

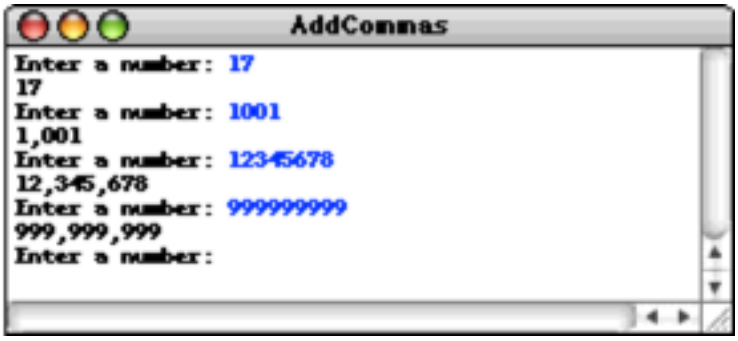
To make it easier for programmers to display numbers in this fashion, implement a method

```
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() {
   while (true) {
      String digits = readLine("Enter a numeric string: ");
      if (digits.length() == 0) break;
      println(addCommasToNumericString(digits));
   }
}
```

your implementation of the `addCommasToNumericString` method should be able to produce the following sample run:



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. Separating Digits and Letters

Write a method

```
private String separateDigitsAndLetters(String str)
```

that takes as input a string and returns all the numbers in the string in their original order of apperance, followed by all the letters in the string, also in their original order of appearance. Any other characters present in the original string should be ignored. For example, your method should return the values shown:

```
separateDigitsAndLetters("a1b2c3d4")      returns "1234abcd"
separateDigitsAndLetters("abc1def2g")     returns "12abcdefg"
separateDigitsAndLetters("abc1??def2g!") returns "12abcdefg"
separateDigitsAndLetters("abcdefg")       returns "abcdefg"
```

4. Pig Latin

Write a method

```
private String pigLatin(String word)
```

that converts a single lowercase word to a simplified version of Pig Latin, a (silly) variant of English 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>

```
pigLatin("elephant")                 returns "elephantyay"
pigLatin("aardvark")                 returns "aardvarkyay"
```

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

```
pigLatin("switch")                   returns "itchsway"
pigLatin("string")                   returns "ingstray"
```

You should assume that **word** has length >= 0 and is comprised of only lowercase letters.

5. Tracing method execution - Graphics

For the program below, draw the output produced on the canvas when the program runs. Note the x, y, width, height and color of any shapes on the canvas, as well as whether or not they are filled.

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

import acm.program.*;
import acm.graphics.*;
import java.awt.*;

public class GraphicsMystery extends GraphicsProgram {
   public void run() {
      double size = 100;
      int x = mystery(50, 50);
      int y = mystery(x, -25);
      GRect r = new GRect(size, size/3);
      paintShop(r);
      add(r, x, y);
   }

   private void paintShop(GRect r) {
      r.setFilled(true);
      r.setColor(Color.BLUE);
   }

   private int mystery(int var, int delta) {
      unknown(var);
      for (int i = 0; i < 4; i++) {
         if (i * 100 <= var) {
            var += delta;
         }
      }
      return var;
   }

   private void unknown(int var) {
      var += 100;
   }
}
```

6. Tracing method execution - Console

For the program below, show what output is produced by the program when it 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;
   }
}
```