Nick Troccoli
CS 106A

Section #3
July 10, 2017

Solution to Section #3

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

1. Adding commas to numeric strings

```
private String addCommasToNumericString(String digits) {
   String result = "";
   int len = digits.length();
   int nDigits = 0;
   for (int i = len - 1; i >= 0; i--) {
      result = digits.charAt(i) + result;
      nDigits++;
      if (((nDigits % 3) == 0) && (i > 0)) {
         result = "," + result;
      }
   }
   return result;
}
```

2. Deleting characters from a string

```
private String removeAllOccurrences(String str, char ch) {
   String result = "";
   for (int i = 0; i < str.length(); i++) {
      if (str.charAt(i) != ch) {
         result += str.charAt(i);
      }
   }
   return result;
}
```

A slightly different approach that involves a **`while`** loop instead of a **`for`** loop:

```
private String removeAllOccurrences(String str, char ch) {
   int pos = 0;
   while (pos >= 0) {
      str = str.substring(0, pos) + str.substring(pos + 1);
      pos = str.indexOf(ch);
   }
   return str;
}
```

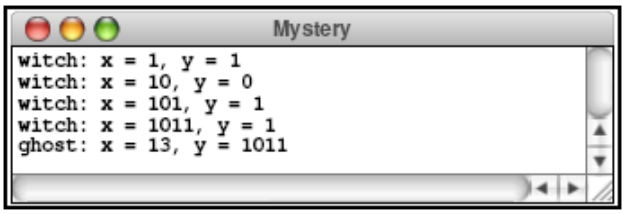
I

3. Converting a string to alternating capital letters

```
private String altCaps(String str) {
    String result = "";
    int counter = 0;
    for(int i = 0; i < str.length(); i++) {
       if (Character.isLetter(str.charAt(i))) counter++;

       if ((counter % 2) == 0) {
          result += Character.toUpperCase(str.charAt(i));
       }else{
          result += Character.toLowerCase(str.charAt(i));
       }
    }
    return result;
 }
```

4. Tracing method execution



5. Class Presidents

```
/*
 * File: ClassPresidents.java
 * --------------------
 * Tallies up the votes for each candidate for the junior
 * and senior classes, and outputs the winners.
 */

import java.io.*;
import java.util.*;

import acm.program.*;

public class ClassPresidents extends ConsoleProgram {

   public void run() {
      try {
         Scanner input = new Scanner(new
File("src/class_presidents.txt"));
         classPresidents(input);
         input.close();
      } catch (FileNotFoundException e) {
         println("File not found.");
      }
   }

   private void classPresidents(Scanner input) {
      String sophomorePresident = "";
      int maxSophomoreVotes = 0;
      String juniorPresident = "";
      int maxJuniorVotes = 0;

      while (input.hasNext()) {
         String name = input.next();
         String classLetter = input.next();
         int votes = input.nextInt();
         if (classLetter.equals("s")) {
            if (votes > maxSophomoreVotes) {
               sophomorePresident = name;
               maxSophomoreVotes = votes;
            }
         } else if (classLetter.equals("j")) {
            if (votes > maxJuniorVotes) {
               juniorPresident = name;
               maxJuniorVotes = votes;
            }
         }
      }
      println("Sophomore Class President: " + sophomorePresident + " ("
            + maxSophomoreVotes + " votes)");

      println("Junior Class President: " + juniorPresident + " (" +
            maxJuniorVotes + " votes)");
   }

}
```

6. **Pig Latin.**

```
 // Given a text file, this method outputs the file as simplified Pig
Latin.
 private void pigLatin(Scanner input) {
   while (input.hasNextLine()) {
     String line = input.nextLine();
     Scanner lineScanner = new Scanner(line);
     while (lineScanner.hasNext()) {
       String word = lineScanner.next();
       if (isVowel(word.charAt(0))) {
         print(word + "yay ");
       } else {
         print(word.substring(1) + word.charAt(0) + "ay ");
       }
     }
     lineScanner.close();
     println();
   }
 }

 // Given a text file, this method outputs the file as full Pig Latin.
 private void fullPigLatin(Scanner input) {
   while (input.hasNextLine()) {
     String line = input.nextLine();
     Scanner lineScanner = new Scanner(line);
     while (lineScanner.hasNext()) {
       String word = lineScanner.next();
       if (isVowel(word.charAt(0))) {
         print(word + "yay ");
       } else {
         // Find the start of the word beyond the first consonants
         int startIndex = 0;
         while (startIndex < word.length() &&
             !isVowel(word.charAt(startIndex))) {
           startIndex++;
         }
         print(word.substring(startIndex) + word.substring(0,
startIndex));
         print("ay ");
       }
     }
     lineScanner.close();
     println();
   }
 }

 // This method returns whether the letter is aeiou (case insensitive)
 private boolean isVowel(char letter) {
   char lowerCaseLetter = Character.toLowerCase(letter);
   return lowerCaseLetter == 'a' || lowerCaseLetter == 'e' ||
   lowerCaseLetter == 'i' || lowerCaseLetter == 'o' ||
   lowerCaseLetter == 'u';
 }
```

7. Negative Sum.

```
private boolean negativeSum(Scanner input) {
    int sum = 0;
    int count = 0;
    while (input.hasNextInt()) {
        int next = input.nextInt();
        sum += next;
        count++;
        if (sum < 0) {
            println(sum + " after " + count + " steps");
            return true;
        }
    }

    println("No negative sum");
    return false;
}
```

Style Focus for Section 3

Common Programming Idioms: A programming *idiom* is a commonly used expression or pattern, like using ++ to increment a variable, or the loop-and-a-half. In this section we went over a common pattern of iterating through a string and building up a new result string. It is good to familiarize yourself with common programming idioms because you will see them appear in others' code, and it will make your own code better.