Nick Troccoli
CS 106A

Section #2
July 3, 2017

Solutions for Section #2

Portions of this handout by Mehran Sahami, Eric Roberts, and Marty Stepp

1. The Fibonacci sequence

```
/*
 * File: Fibonacci.java
 * --------------------
 * This program lists the terms in the Fibonacci sequence up to
 * a constant MAX_TERM_VALUE, which is the largest Fibonacci term
 * the program will display.
 */

import acm.program.*;

public class Fibonacci extends ConsoleProgram {

   /* Defines the largest term to be displayed */
   private static final int MAX_TERM_VALUE = 10000;

   public void run() {
      println("This program lists the Fibonacci sequence.");
      int t1 = 0;
      int t2 = 1;
      while (t1 < MAX_TERM_VALUE) {
         println(t1);
         int t3 = t1 + t2;
         t1 = t2;
         t2 = t3;
      }
   }

}
```

2 . ASCII Art

```
/*
 * File: AsciiFigure.java
 * ----------------------
 * Draws a figure with SIZE number of lines, where each line contains
 * 8 more stars than the line before it. Lines are padded with
 * forward and backward slashes.
 */

import acm.program.*;

public class AsciiFigure extends ConsoleProgram {
    private static final int SIZE = 5;           // number of lines
    private static final int STAR_INCREMENT = 8; // stars added/line

    public void run() {
        int maxLineWidth = STAR_INCREMENT * (SIZE - 1);
        for (int line = 0; line < SIZE; line++) {
            int numStarsOnLine = STAR_INCREMENT * line;
            int numSlashes = (maxLineWidth - numStarsOnLine) / 2;
            for (int i = 0; i < numSlashes; i++) {
                print("/");
            }
            for (int i = 0; i < numStarsOnLine; i++) {
                print("*");
            }
            for (int i = 0; i < numSlashes; i++) {
                print("\\");
            }
            println();
        }
    }

}
```

3. Piglet

```
/*
 * File: Piglet.java
 * --------------------
 * This program plays the 1-player dice game "Piglet".  Each turn,
 * the player rolls a dice – if it's a 1, then the game is over and
 * they get a score of 0.  Otherwise, the value is added to their
 * total and the player chooses whether or not to roll again.  The
 * player tries to get the highest score possible.
 */

import acm.program.*;
import acm.util.*;       // for RandomGenerator

public class Piglet extends ConsoleProgram {
    public void run() {
        println("Welcome to Piglet!");
        int sum = 0;
        boolean rollAgain = true;

        int die = RandomGenerator.getInstance().nextInt(1, 6);
        println("You rolled a " + die + "!");

        // Loop until we roll a 1 or player doesn't want to re-roll.
        while (die != 1 && rollAgain) {
            sum += die;
            rollAgain = readBoolean("Roll again?", "yes", "no");
            if (rollAgain) {
                die = RandomGenerator.getInstance().nextInt(1, 6);
                println("You rolled a " + die + "!");
            }
        }
        if (die == 1) {
            sum = 0;
        }
        println("You got " + sum + " points.");
    }
}
```

4. CalculateLine

```
/*
 * File: CalculateLine.java
 * --------------------
 * Reads in a line equation from a user and, for every x entered
 * until -1, outputs the corresponding y value.
 */

import acm.program.*;

public class CalculateLine extends ConsoleProgram {
     public void run() {
           println("This program calculates y coordinates for a line.");
           int slope = readInt("Enter slope (m): ");
           int intercept = readInt("Enter intercept (b): ");

           int x = readInt("Enter x: ");
           while (x != -1) {
                 int y = slope * x + intercept;
                 println("f(" + x + ") = " + y);
                 x = readInt("Enter x: ");
           }
     }
}
```

5. Evaluating Expressions

	Expression	Value
a)	3 * 4 + 2 * 3	18
b)	2 + 19 % 5 – (11 * (5 / 2))	-16
c)	2 + 2.0	4.0
d)	(3/4) + (3/4)	0
e)	(5 * 7.0 / 2 – 2.5) / 5 * 2	6.0
f)	55 + “abc”	“55abc”
g)	“abc” + 1 + (1/2)	“abc10”

6. Nested For Loops

The code prints out the following output:

```
           *
          ***
         *****
        *******
       *********
      ***********
     *************
    ***************
   *****************
  *******************
```

Style Focus for Section 2:

Always Use Constants: Our code should never contain "magic numbers," meaning numbers we use in our code that don't have a clear meaning. For example don't just have "7," say ***`STAR_INCREMENT`***. Instead of "1000," we write **`MAX_TERM_VALUE`**. Well-named constants make it clear what the purpose of the variable is, and also help reduce errors. For instance, if someone wants to change the **`MAX_TERM_VALUE`**, they can modify its value everywhere in the program by only changing it once. If we just wrote "1000," they would have to go searching through the code to find all the places we use this value. The only numbers we don't need to turn into constants are the numbers 0, 1 and sometimes 2.

There Are Many Ways To Solve the Same Problem: There are many ways to decompose the same problem. When you write your own programs try and consider the many ways to solve the problem, and the trade-offs and benefits of each solution.