Colin Kincaid
CS 106A

Section #2
July 3, 2018

Section Handout #2—Intro to Java

Portions of this handout by Eric Roberts and Marty Stepp

1. Warmup: Evaluating Expressions

Evaluate the following; use the proper type (i.e. 7.0 for a double instead of 7).

```
a) 3 * 4 + 2 * 3
b) 2 * 19 % 5 – (11 * (5 / 2))
c) 2 + 2.0
d) (3/4) + (3/4)
e) (5 * 7.0 / 2 – 2.5) / 5 * 2
f) 55 + “abc”
g) “abc” + 1 + (1/2)
```

2. Warmup: Nested For Loops

What does the code below print out? (Note: unlike println, print does not add a newline).

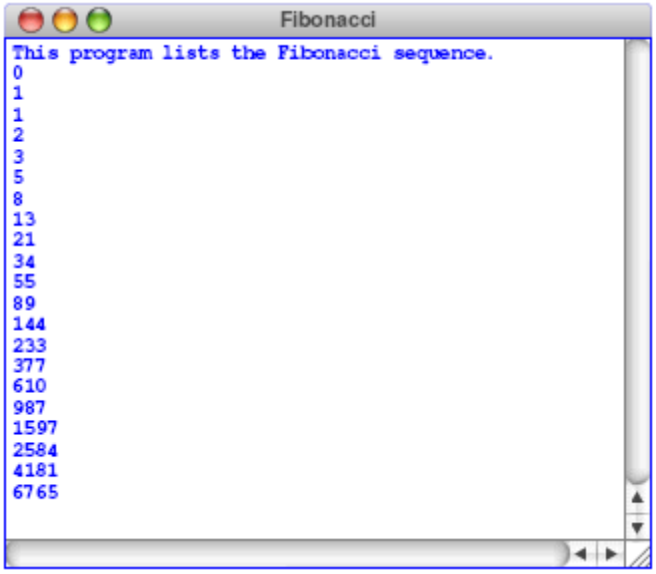
```
for (int i = 1; i <= 10; i++) {
    for (int j = 1; j <= 10 - i; j++) {
        print(" ");
    }
    for (int j = 1; j <= 2 * i - 1; j++) {
        print("*");
    }
    println();
}
```

3. The Fibonacci Sequence

In the 13th century, the Italian mathematician Leonardo Fibonacci—as a way to explain the geometric growth of a population of rabbits—devised a mathematical sequence that now bears his name. The first two terms in this sequence, **Fib**(0) and **Fib**(1), are 0 and 1, and every subsequent term is the sum of the preceding two. Thus, the first several terms in the Fibonacci sequence look like this:

Fib(0)	=	0	
Fib(1)	=	1	
Fib(2)	=	1	(0 + 1)
Fib(3)	=	2	(1 + 1)
Fib(4)	=	3	(1 + 2)
Fib(5)	=	5	(2 + 3)

Write a program that displays the terms in the Fibonacci sequence, starting with **Fib**(0) and continuing as long as the terms are less than 10,000. Thus, your program should produce the following sample run:



To do this, you should use a **while** loop, presumably with a header line that looks like this:

```
while (term < MAX_TERM_VALUE)
```

Note that the maximum term value is specified using a named constant.

4. Calculating Lines

Write an interactive console program that calculates *y* coordinates on a line. First, it prompts the user for a slope, *m*, and an intercept term, *b* (remember that a line has an equation of the form $y = mx+b$). Then, the program prompts the user for *x* values until the user enters the **`SENTINEL`** (the value of which is specified using a named constant). For each entered number, print the *y* value on that line for that entered *x* value. Here is a sample run of the program, with **`SENTINEL = -1`** (user input is underlined):

```
This program calculates y coordinates for a line.
Enter slope (m): 2
Enter intercept (b): 4
Enter x: 5
f(5) = 14
Enter x: 1
f(1) = 6
Enter x: -1
```

Your program should work properly regardless of the value of **`SENTINEL`**.

5. Extra: Piglet

Write the code for a simple 1-player dice game called "Piglet" (based on the game "Pig"). The player's goal is to accumulate as many points as possible without rolling a 1. Each turn, the player rolls the die; if they roll a 1, the game ends and they get a score of 0. Otherwise, they add this number to their running total score. They then choose whether to roll again, or end the game with their current point total. Two sample games are shown below.

```
Welcome to Piglet!
You rolled a 5!
Roll again? yes
You rolled a 4!
Roll again? yes
You rolled a 1!
You got 0 points.
```

```
Welcome to Piglet!
You rolled a 6!
Roll again? yes
You rolled a 2!
Roll again? yes
You rolled a 2!
Roll again? no
You got 10 points.
```

Tip: use the `readBoolean` method to prompt the user with a yes/no question.
Tip: you will need to use `RandomGenerator`, covered in the textbook and in lecture on July 9th.