

Section Handout #2—Simple Java

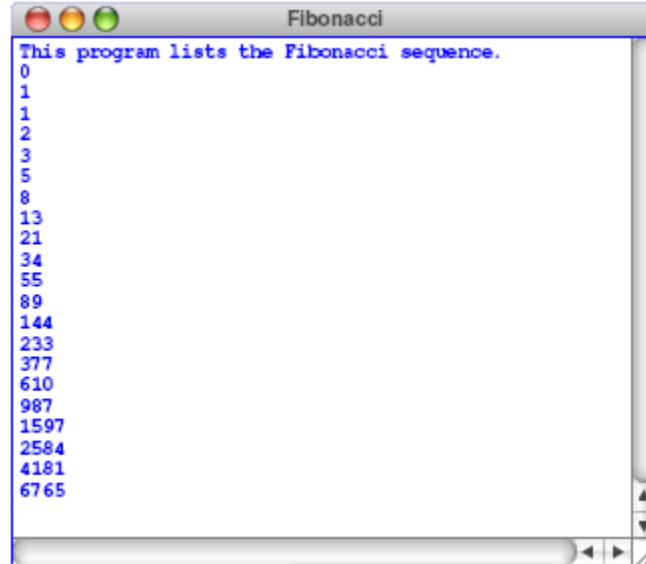
Portions of this handout by Eric Roberts

1. 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:

$$\begin{aligned}\mathbf{Fib}(0) &= 0 \\ \mathbf{Fib}(1) &= 1 \\ \mathbf{Fib}(2) &= 1 \quad (0 + 1) \\ \mathbf{Fib}(3) &= 2 \quad (1 + 1) \\ \mathbf{Fib}(4) &= 3 \quad (1 + 2) \\ \mathbf{Fib}(5) &= 5 \quad (2 + 3)\end{aligned}$$

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 or equal to 10,000. Thus, your program should produce the following sample run:



```
Fibonacci
This program lists the Fibonacci sequence.
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
```

This program should continue as long as the value of the term is less than or equal to the maximum value. 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. Your program should work properly regardless of the value of **MAX_TERM_VALUE**.

2. 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
```

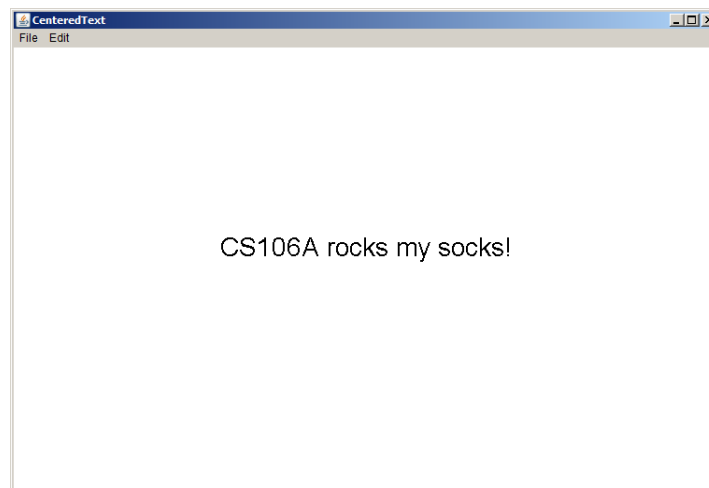
Your program should work properly regardless of the value of **SENTINEL**.

3. Drawing Centered Text

Your job is to write a **GraphicsProgram** that displays the text message (i.e., **GLabel**):

CS106A rocks my socks!

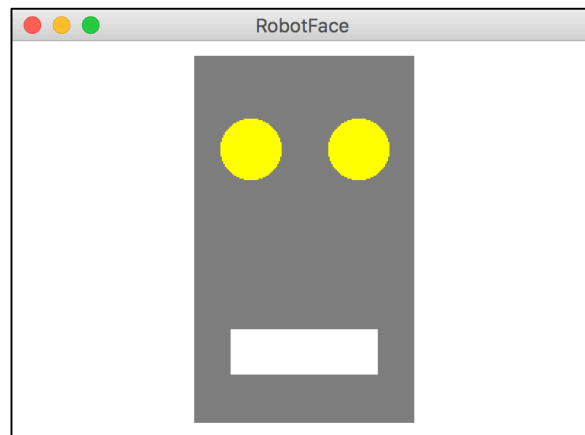
The text should be displayed in SansSerif 28-point font, and centered horizontally and vertically in the middle of the graphics window, looking something like this:



You can find the width of a label by calling `label.getWidth()` and the height it extends above the baseline by calling `label.getAscent()`. If you want to center a label, you need to shift its origin by half of these distances in each direction.

4. Drawing a face

Your job is to draw a robot-looking face like the one shown in the following sample run:



This simple face consists of four parts—a head, two eyes, and a mouth—which are arranged as follows:

- *The head.* The head is a big rectangle whose dimensions are given by the named constants **HEAD_WIDTH** and **HEAD_HEIGHT**. The head is gray.
- *The eyes.* The eyes should be circles whose radius in pixels is given by the named constant **EYE_RADIUS**. The centers of the eyes should be set horizontally a quarter of the width of the head in from either edge, and one quarter of the distance down from the top of the head. The eyes are yellow.
- *The mouth.* The mouth should be centered with respect to the head in the x -dimension and one quarter of the distance up from the bottom of the head in the y -dimension. The dimensions of the mouth are given by the named constants **MOUTH_WIDTH** and **MOUTH_HEIGHT**. The mouth is white.

Finally, the robot face should be centered in the graphics window.