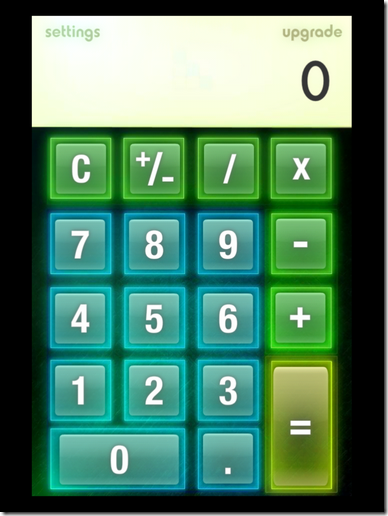
**Lab 2: Java Calculator**

**Purpose**

1. Get practice using a Java GUI layout manager
2. More practice with ActionListeners

**Goal**

For this lab, you are to create a calculator application. The button layout should mimic that in the picture below. You are NOT required to get an exact representation of the calculator. The image is only supposed to show button location, function, and relative size.



You will need to use a layout manager (such as GridBagLayout) so that your calculator will have the following behaviors:

1. As the user re-sizes the calculator, the buttons must stretch to appropriately fill the calculator.
2. The display (where the numbers go) should not stretch vertically.
3. OPTIONAL: Specify a min and max size so the buttons are always visible, but you can’t make the calculator too large.

The only interface to the calculator is the mouse (**not** the keyboard). You should not be able to type into the display, and you do not need keyboard shortcuts for the buttons.

I recommend the following classes; however, you may do whatever classes make sense to you:

1. Screen (subclassing JTextField). The screen kept track of the current number being entered, and the result following a calculation. I had the following methods:
   1. add digit – called when a numeric button was pressed
   2. get value – returns the current value on the display
   3. set value – sets the value following a computation
2. Number button (subclassing JButton). The number buttons behave the same except for the “value” of each button, so I chose to make a subclass.
3. Calculator (subclassing JPanel). This class contained all the buttons, the screen, and the layout. It contained the action listeners for the non-numeric buttons.

**Thought process**

You can think of this lab as containing two problems: how to lay out the UI and how to get correct behavior. You can think of the two problems separately (although some decisions in one area will affect decisions in the other).

When you start thinking about correct behavior, I’d suggest you experiment with the “standard” Windows calculator. However, Microsoft in their infinite wisdom replaced the perfectly good Windows 7 calculator with a Windows Store app that can, at best, be described as “flaky”. I created my own calculator that you can install and run from here: <http://unix.cset.oit.edu/~philip.howard/calc/publish.htm> If you have Windows Defender (or some other virus software), it may object twice before the calculator comes up (may have to click on “More info” before the “Run anyway” button shows up). Just click “run anyway” or whatever to get it to run. I will demo this process in lab.

Questions:

1. What, exactly, do the operation (+ - X /) buttons do?

They save the current text as a double for later use and change a variable to match what was used.

1. What, exactly, does the equals button do?

It retrieves a saved value, finds out what the last operator used was, and performs the correct calculation on it with the current value shown on the calculator.

1. What, exactly, does the +/- button do?

It effectively multiplies the current value shown on the calculator by -1.

1. What information do you need to maintain in order to get this behavior? (Hint: you need more than just the information on the screen).

You need to maintain the last operator used and the last value that was on the calculator.

1. Where should that information be stored?

I stored it in m\_screen because both my OperatorButtons and NumberButtons had access to it as well as my CalcIF.

**Grading**

Points will be awarded for the following categories:

1. Proper layout 40 pts:
   1. The layout mimics the picture in this write-up (layout - not other appearance features).
   2. The layout behaves appropriately as the window is resized
2. Proper behavior 40 pts:
   1. Each button behaves the same way as the equivalent button on my example.
   2. Calculations are done correctly.
   3. No surprises.
3. Appearance 10 pts:
   1. Up to 10% extra points (in this category) are available if you go beyond the standard appearance of the Java controls. Be creative, but don’t have your creativity detract from usability.
   2. Up to 20% extra points (in this category) if you add a menu that you can use to change the appearance of the calculator. Examples: various skins or add a scientific mode.
4. Coding style 10 pts:
   1. Code has proper header comments.
   2. Code is easy to understand. When in doubt, explain in comments.
   3. Good programming practices were followed. In particular, appropriate code re-use. You should not need to create a different action listener for each button, but neither should you create a do-all action listener with all kinds of convoluted logic to make it work for any and every button press.