Lab 5

Due 25 February

Handout 5
CSC 062: Spring, 2007
13 February

Calculator

1 Lab

Your next program is to implement a calculator application. The calculator should take expressions input in postfix notation and display the results of the computation. Thus to calculate `3+5`, click on `3, enter, 5, +`. The answer should then be displayed. To calculate `3+5*7` (which is written `3 5 7 * +` in postfix notation), click on `3, enter, 5, enter, 7, *, +`.

Aside from the main class, the program should include four other classes - one, `State`, representing the state of the calculator, and three listener classes: `DigitButtonListener`, `OpButtonListener`, and `MiscButtonListener`, which each implement the `ActionListener` interface.

A state object represents the memory of the calculator. You may think of it as representing the printed circuit in the caculator. Its purpose is to keep track of the current state of the computation. For instance, it has to keep track of whether the user is entering a number, and what the number is so far. It also must keep track of the stack used in the computation and update the display window.

Objects of class `DigitButtonListener` and `OpButtonListener` respond to events generated by buttons which represent digits and operations, respectively. Each digit key has its own specialized `DigitButtonListener` which is responsible for knowing which number key it is listening to. Similarly each of the operation keys has its own specialized `OpButtonListener` which is responsible for knowing which number key it is listening to. When the user clicks on a button with a digit on it, its corresponding listener is responsible for informing the state what number has been clicked on so the state can use it to build the number being typed in. When the user clicks on an operation button (+, -, *, or /), the corresponding listener is responsible for informing the state what operation is to be performed next so that the state can perform the operation.

Notice that we only create "smart" buttons for digits and operations. This is because there are several buttons of each kind. The "unique" buttons: `Enter`, `pop`, and `clear`, are handled directly by the `MiscButtonListener`. They simply send a message to the state corresponding to the operation.

To help you get started I have provided you with the code for the main `Calculator` class. This relieves you of the burden of laying out the buttons and display of the calculator. I have also included the code for `MiscButtonListener` as an inner class of `Calculator`, and the code for `OpButtonListener` as a separate class. I have also included the skeleton of code for the `State` class to indicate what methods it should support. You are to write the code for `DigitButtonListener` and fill in the bodies of the constructor and other methods of `State`. Note that there is a call of the constructor of `DigitButtonListener` in the constructor of `Calculator`. Your class should support that constructor. Note also that the state is responsible for updating the display when necessary.

You might find it easier to write the code first under the assumption that the user must push the "enter" button after punching in each number. However, for full credit, it should also handle the case where the user punches in a number followed immediately by an operation. The result should be equivalent to sticking in an intervening "enter". That is punching in `5, enter, 7, +` should give the same results as `5, enter, 7, enter, +`.

Once you have the rest of the calculator working properly I would like you to add one or more new buttons. Possibilities include a squaring button, factorial, or x^y. You should add the new button so that it looks nice, add an appropriate listener (probably created from `OpButtonListener`), and make sure that the state knows how to handle that operation.

If you are interested in extra credit, I suggest that you add a decimal point button so that your calculator can handle floating point numbers. Be sure to check that the user does not input an illegal number (e.g., with more than one decimal point). Possibilities for new buttons with this include square root, trig functions (e.g., sin, cos, tan, etc.), inverse trig functions, x^y, e^x, factorial, inverses, or logarithms. See the class `Math` in package `java.lang` for a list of available mathematical functions in Java.

2 Collaboration

I encourage you to work together with a partner and use "pair programming" in writing this program. If you decide to do this, you must follow the following rules:

- Begin by reading the article "All I Really Need to Know about Pair Programming I Learned In Kindergarten" by Williams and Kessler. A preliminary version is available at
  - http://collaboration.csc.ncsu.edu/laurie/Papers/Kindergarten.PDF

  but you may prefer the published version as it appeared in the Communications of ACM (CACM). The college has an electronic subscription to CACM, so it is available for free from on campus. Go to http://portal.acm.org/dl.cfm and search for the article.
- Agree with your partner to follow the rules for pair programming. In particular, all code must be worked on together, and you must trade roles of "driving" and "reviewing" at least every half hour.
- Turn in one copy of your program with both of your names on the folder and in the comments for each class.

3 What to hand in

Your final program will be due on Sunday, February 25 at 11:59 p.m. As usual place all of your code in a folder and drop it off in the cs 062 dropbox folder.