Calculator

Assignment 5
Due Monday, February 27, 2012
CSC 062: Spring, 2012

Calculator

Your next program is to implement a postfix 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, *, +`. (Note that the only thing pressing `enter` does is to signal the end of the number being input.)

A version of this program running as an applet can be found at
`http://www.cs.pomona.edu/classes/cs062/assignments/CalculatorApplet/CalculatorApplet.html`.

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.

State

A state object represents the memory of the calculator. You may think of it as representing the printed circuit and memory 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.

The most important feature of the `State` is its stack, which represents those numbers stored on the computer. (You may use `java.util.Stack` rather than writing your own.) When the user types a number, it is stored on the top of the stack. If the enter key is pressed and then a digit, a new number is stored on top of the stack, with the previous value sitting below the new one on the stack. If there are at least two elements on the stack and the user pushes on an operation key then the top two elements are popped off, the operation is performed on them (make sure you do them in the right order!) and then the answer is pushed back on the stack. For example, to calculare 23 - 17, the user presses `2 3 Enter 1 7 -`, and the answer of 6 will end up on top of the stack and in the display. (The display always shows the number on top of the stack.)

If the user makes a mistake, e.g., dividing by 0 or trying to perform an arithmetic computation when there are fewer than two elements on the stack, then the display should show `Error`. You can detect divide by zero yourself or let the system throw an `ArithmeticException` when a divide by zero occurs. Either way, you must handle the error (i.e., your program should not crash – and no "ugly red stuff") and have the display show `Error`.

If the stack is empty, the display should show `0`. It is not considered an error if the user pushes the `Pop` button when there is nothing on the stack. Just ignore the key press.

Listeners

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. (See the code in the `Calculator` class to see how listeners are added to buttons.) 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`, `Exch` (to exchange the top two elements on the stack) and `Clear`, are handled directly by the `MiscButtonListener`. They simply send a message to the state corresponding to the operation.

Getting Started

To help you get started I have provided you with the code for the main `Calculator` class. This relieves you of the burden of layout of the buttons and display of the calculator. I have also included the code for `MiscButtonListener` as an inner class of `Calculator` (it is nested at the bottom of that class), 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` object is responsible for updating the display when necessary. All startup code is available in `/common/cs/cs062/assignments/assignment5`.

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, +`. (Think about what pressing the enter key actually does. What is the difference between pressing `3 7` and `3 Enter 7`? It makes a difference.)

The Next Step

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 (created from `OpButtonListener` if you like), 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 two decimal points). 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.

Grading

You will be graded based on the following criteria:

criterion	points
Digits and "Enter" handled appropriately	3
Arithmetic operations handled correctly	3
Appropriately handle errors and show message	2
Misc keys handled correctly	3
Digit listener	2
Extra method/calculator functionality	2
appropriate comments (including JavaDoc)	3
style and formatting	2

What to hand in

As usual, export the entire folder from Eclipse to your desktop, change the name of the folder to "Lastname-Assignment5" (where you should replace *Lastname* by your last name). Make sure the folder contains both your .java and .class files. Then drag it into the dropoff folder in `/common/cs/cs062/dropbox`, where an alias for the cs062 folder should be sitting on your desktop already. Be sure that your code is clear, formatted properly and commented appropriately (using Javadoc... see the first assignment for details on what's expected for comments).