SE/CS 2S03: Principles of Programming

Due on December 13th

Dr. Jacques Carette

Idea

The goals of this assignment are:

1. learn about GUI programming and event handlers
2. get more familiar with OO programming with lots of classes.

The Task

You will implement a small calculator application: you can enter digits, add, subtract, multiply and divide numbers. The calculator should only display the input as entered, until the '=' key is entered: the result should then be computed and displayed. Provide parentheses as input as well.

For example, the key presses `1`, then `+` then `3` would display `1+3`. Pressing `=` would change the display to `1+3=4`.

You should also implement an `AC` (for *all clear*) button, to start new computations.

To make this more interesting, your calculator should have two modes: integer mode (as above), and floating-point mode. All input in integer mode should be interpreted as integers, and produce integers. All inputs in floating point mode should be interpreted as doubles, and produce doubles. For example, after entering `1/2` and pressing `=` in float mode should produce `1/2=0.5000000` (where you should print all results to 6 digits of accuracy).

There should be a 'backspace' key to correct mistakes.

The details

As there are a number of corner cases, here is the specification for those cases:

- If the input is syntactically incorrect, then simply display `syntax error`. In other words, pressing `=` when you have previous entered something (like) `1+)` would result in `1+)= syntax error`. Pressing 'backspace' would then revert the display back to the bad input.
- In integer mode, if the result is not an integer, let the result be `fraction`. Backspace should work as above. This includes division by zero.
- In floating point mode, division by zero should cause the result to display as `NaN`. infinities should also 'work'.
- Changing the mode only effects what `=` *does*. It otherwise has no effect.

For the implementation, you *will*:

- start from the UI code (provided as fora5.zip), minus the action code.
- remove '%' from the UI and replace it with 'B' (for backspace).

- remove '.' from the UI (replace with a blank space).
- use the 'parser' provided (see assignment page) [you may modify it],
- adapt the `Expr` class and its sub-classes to perform the computations. [Note: you will *not* need the printing code from those classes, but they are very useful for debugging.] You should add 2 new methods to each (sub)class, one for interpreting each mode.

Submission Requirements

- A *single* zip file containing all your java files, including all the ones provided (whether modified or not).

Marking Scheme

- Programs which do not compile will be given a mark of 0, no matter how *close* your code might be to the correct answer. **YOU** are responsible for ensuring that what you hand in can be compiled by the TA. The easiest way to ensure this is to put your assignment files (as extracted from the zip) onto a computer which has not been used to develop this assignment and compile. The computers in ITB 235 are perfect for such a test.
- The (modifications of the) UI part of the code will be worth 50%, the computation parts 50%.

Bonus

Each one of these will be worth extra marks:

- (easy) Make sure that in your code there is a *single* conditional on `mode`.
- (easy-medium) Improve the parser to implement operator precedence, and to give better error messages.
- (medium) Have *no* conditionals on mode. Hint: First class functions (see the rosettacode.org entry for first class functions, look up the Java 8 way of doing that on that page) and arrays.