CS 474: Object Oriented Programming Languages and Environments
Fall 2018

Second Smallalk project

Due time: **9:00 pm on Wednesday 10/24/2018**

You are required to implement an *Programmable Calculator (PC)* in Cincom Smalltalk. The PC is similar to a traditional calculator; however, it also lets users specify sequences of arithmetic operations on 4 registers, denoted by *x, y, w* and *z*. To use the PC a user would typically enter a sequence of arithmetic expressions in an appropriate text editor. Each expression appears in a separate line in the text editor. Lines are numbered from 1 onward. Next the user would manually enter initial values for *x, y, w* and *z* in appropriate input fields. (The default values for these fields would be zero.) Finally, the user would execute the sequence of expressions, resulting in modified values being displayed for *x, y, w* and *z*. The order of execution of expressions is sequential from the first expression (on Line 1), except when an expression with the question mark is encountered. (See below.)

The syntax of each arithmetic expression conforms to the following four patterns, where *id* can be one of *x, y, w* and *z* and *op* can be one of +, -, *, ** and /:

1. *id = id op id.*
2. *id = id op constant.*
3. *id = constant.*
4. *id ? go int.*

The first two forms above supports addition, subtraction, multiplication, division and exponentiation of two numeric operands. The operands must be of integral or floating point type (i.e., no fractions or complex numbers.) The third form stores a numeric constant in one of the registers. The fourth form allows control transfers to arbitrary lines in the expression sequence entered by the user. If value associated with the *id* appearing on the left of the questioni mark is different from zero, the expression located at the line denoted the *int* operand is executed instead of the next expression. Expression execution ends when the last expression in the sequence has been evaluated. However, to prevent infinite execution loops, you should halt expression execution also after 100 expressions have been evaluated.

(The periods at the end of each expression type should make it easier for you to parse the sequence of expressions extracted from the text editor widget.) Here is an example of a simple PC instruction sequence. At the end of this execution *x, y, w* and *z* should hold the values 4, 2, 0 and 16 respectively.

```
1:   x = 3.
2:   y = 2.
3:   w = 2.
4:   z = x ** y.
5:   w = w - 1.
6:   x = x + 1.
7:   w ? 4.
8:   x = x - 1.
```

Your GUI must include at least the following widgets:

1. A text editor field for entering and editing the sequence of expressions. Each line in the editor field should be numbered, to make it easy for the user to see the position of each in the list.
2. Two *action buttons* for starting the execution of the expression sequence in the editor field. The sequence could be executed in normal mode (the entire sequence of expressions) or debugging mode (one expression at a time) depending on the button being selected.

3. Four widgets for entering and displaying the values of x, y and z before and after the execution of the expressions.

You may assume that expression sequences in the PC by PC users are syntactically correct. (You are not required to perform error diagnosis or correction). Also, you must use inheritance and at least one abstract superclass in your implementation. One good way for complying with this requirement is to define an abstract superclass for expressions with concrete subclasses for each particular type of expression. Make sure that an abstract superclass you define includes at least one concrete method, and that this method is actually executed during normal operation of the PC.

You must work alone on this project. Your project code should be in a special package called CS474. Save all your code by filing out that package in the file xxx.st, where xxx denotes your last name. Submit the file by clicking on the link provided with this assignment. If you wish to submit multiple files, you may create a *zip* archive and submit a file named xxx.zip where again xxx denotes your last name. No late submissions will be accepted.