CS 474: Object Oriented Programming Languages and Environments
Fall 2021

Second Ruby project

Due time: **11:59 pm on Sunday 10/24/2021**

You are required to implement a *Programmable Calculator (PC)* in Ruby. 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 a file named "pc_input.txt". Each expression appears in a separate line in the input file. Lines are numbered from 1 onward. The initial value of registers *x, y, w* and *z* is zero.

Once an input file is ready for execution, the PC user would start the PC and enter one of two commands, either 'r' (run) or 's' step. Both commands read the input file. The 'r' command executes all the lines in the input until the last line, printing the value of the registers and the index position of the instruction after each instruction is executed. Command 's' executes just one instruction at a time, printing the instruction being executed and the values of the registers after each instruction. (The user will hit command 's' repeatedly to advance the execution of the instructions.) Finally, command 'x' exits the PC. 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. 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 question 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 a total of 100 expressions have been evaluated.

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.
```

You may assume that expression sequences in the PC by PC users are syntactically correct. (You are not required to perform error checking or correction). Also, you must use inheritance and at least one abstract superclass (à la Ruby) 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, not to be redefined in the subclasses, 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 zip archive bearing your first and last names (e.g., jane_doe.zip). Save all your code in the archive and submit the archive by clicking on the link provided with this assignment.