OOP Programing Code

March 22, 2021

1 Problem

To express an arithmetic expression, there are 5 following classes:

Exp: general arithmetic expression

BinExp: an arithmetic expression that contains one binary operators (+,-,*,/) and two operands. To construct a BinExp object, you must pass parameters: first operand, operator, second operand, respectively.

UnExp: an arithmetic expression that contains one unary operator (+,-) and one operand. To construct a UnExp object, you must pass the operator first.

IntLit: an arithmetic expression that contains one integer number

FloatLit: an arithmetic expression that contains one floating point number

Define these classes in Python (their parents, attributes, methods) such that their objects can response to eval() message by returning the value of the expression. For example, let object x express the arithmetic expression 3 + 4 * 2.0, x.eval() must return 11.0

Extend the contents of classes Exp, BinExp, UnExp, IntLit, FloatLit such that they can response to printPrefix() message to return the string corresponding to the expression in prefix format. Note that, unary operator +/- is printed as +./-. in prefix format and there is a space after each operator or operand. For example, when receiving message printPrefix(), the object expressing the expression -4 + 3 * 2 will return the string "+ -. 4 * 3 2 "

As in the previous question, when a task is added into expression classes, new methods are added into these classes. Please change the way these classes are implemented in such a way that these classes do not change their contents when new tasks are added into these classes:

- Define class Eval to calculate the value of an expression

- Define class PrintPrefix to return the string corresponding to the expression in prefix format

- Define class PrintPostfix to return the string corresponding to the expression in postfix format

Let x be an object expressing an expression, x.accept(Eval()) will return the value of the expression x, x.accept(PrintPrefix()) will return the expression in prefix format and x.accept(PrintPostfix()) will return the expression in postfix format.

Be careful that you should not allowed to use type(), isinstance() when implementing this exercise.

Tip: Use Visitor pattern.

In this exercise, we use:

x1 = IntLit(1)

x2 = FloatLit(2.0)

x3 = BinExp(x1,"+",x1)

x4 = UnExp("-",x1)

x5 = BinExp(x4,"+",BinExp(IntLit(4),"*",x2))