CSC205AB
Minilab – ExpressionTrees

You are given:

You are given a class called ExpTree, which implements an "expression tree." ExpTree already works and has the following:

- A constructor which accepts a String that holds a fully parenthesized arithmetic expression. It parses the String and uses 2 Stacks to build up the expression tree. It is commented enough that it should be understandable, although you won't need to understand it for this Minilab.
- A seeTree() that returns a String that uses \n's and \t's to show what the tree looks like on its side. If you "capture output" (in Textpad) and then print (on paper) what is outputted, then you can draw in the links to see what the tree looks like.
- An inOrder() method, which will traverse the tree in LNR order. It also inserts parentheses so that the String that it returns looks like the original parenthesized expression.

The seeTree() and inOrder() methods are implemented recursively, so that there are actually 2 methods for each public method.

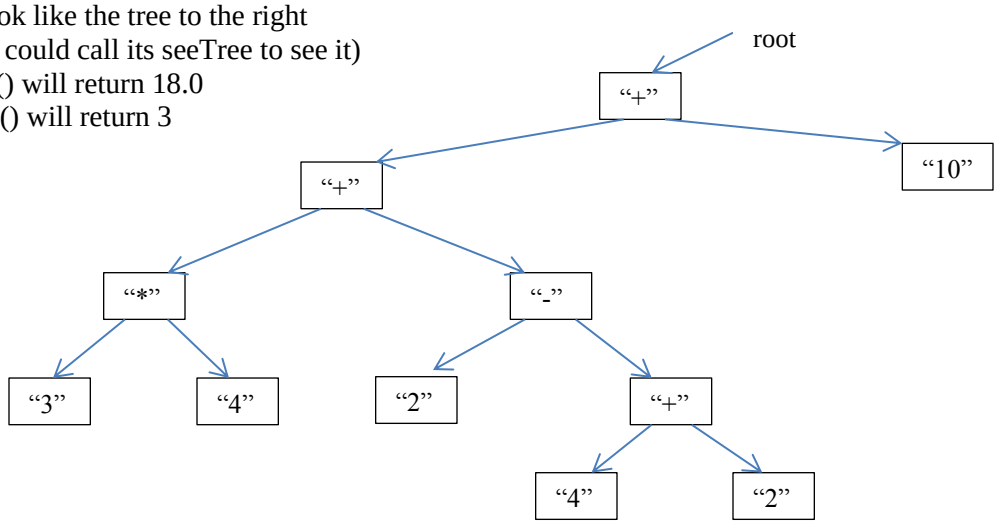
You should download the ExpTree class and add 2 (pairs of) methods:

You should download the ExpTree class and add 2 methods. They should implement their functionality recursively so they will actually be implemented as 2 pairs of methods. You can use the recursive methods that are already in ExpTree as guides. The public methods you should write are:

1) An evaluate() method, which returns the arithmetic evaluation of the MinilabExpTree (as a double). This should be done recursively, so you will need 2 methods to do it. In the case where it would result in division or mod by 0, it should throw a new ArithmeticException with a descriptive String. If the tree is empty, evaluate() should also throw a new IllegalStateException with a descriptive String.
2) A numPlus() method, which will traverse the tree and return how many plus signs ("+") are contained (as an int). This should be done recursively, so you will need 2 methods to do it.

For example:

- If a MinilabExpTree that is built using "(((3*4)+(2-(4+2)))+10)"
- it will look like the tree to the right
  (a driver could call its seeTree to see it)
- evaluate() will return 18.0
- numPlus() will return 3



Notes:

- if the expected result is 6.36 and your result is 6.359999999999…, that is should be OK (Java doing base10 arithmetic in a base2 number system). If HyperGrade complains, please let me know.
- To convert a String to a double, use Double.parseDouble(). For example,
  String numString = "3.14";
  Double.parseDouble(numString) would return an actual 3.14 as a double

Comments and formatting: Your program should have opening comments for the class and each method (and anywhere in the code that is "tricky." It should also have correct indenting and meaningful variable names.

Please submit: Your MinilabExpTree.java file on HyperGrade.