

# CSE222 – Homework5 – Question2

## Report

Nevzat Seferoglu

171044024

## Problem Solution Approach :

### \* Problem Definition :

Implement **ExpressionTree** class of arithmetic operations which extends the **BinaryTree** class implementation given in your Data Structures book.

Your **ExpressionTree** implementation must have the following:

A constructor to initialize the tree structure with the given expression string. The expression string will be given as a parameter to the constructor. The expressions will include integer operands and arithmetic operators. Operands and operators will be separated by spaces. The constructor will use the overridden **readBinaryTree** method. You should override it such that it will be able to create an expression tree by reading both prefix and postfix expressions.

The **binaryTree** implementation of the book includes a **preOrderTraverse** method. You will add a **postOrderTraverse** method to traverse the tree post order.

The **binaryTree** implementation of the book includes a **toString** method which creates a string of the tree structure in preorder. You will add a **toString2** method which will create a

string of the tree structure in post order. This method will use the **postOrderTaverse** method.

You will add **eval** method which evaluates the expression and returns the result as an integer.

### \* Approach :

Prefix and postfix expression are exploited in many fields of computer science. Both expressions can be converted the ' binary tree ' because nearly all operator can take two different operands and return the result of it. In this tree , operands correspond to leaf node. They cannot take any other operands and operators.


According to given input expression , constructing a tree differs according to type of expression. Traversing methods can also contain discrete tasks. Those traversing methods are explained detailed in Javadoc file of project.

In evaluation process , using postorder traverse is sensible. Because expression tree evaluation is going forward from bottom to top. Accessing the bottom operators and operands is needed to be evaluated first for reaching the top and finally result of it.













## Test Cases:

**\*\*Note :** There are separated test parts below. But prefix and postfix expression use all same function and constructor. It is only for making test result clear view.


### Prefix Input Constructors Test:

Test ID	Scenario	Test Data	Expected Result	Actual Result	Pass	Fail
T01	Creating tree with given prefix expression.	tree = + + 10 * 5 15 20	Construct a tree.	Expected result has occurred.		













### Methods Test for Prefix input:

Test ID	Scenario	Test Data	Expected Result	Actual Result	Pass	Fail
T01	-eval ( )	tree = + + 10 * 5 15 20	105	Expected result has occurred.		
T02	-toString()	tree = + + 10 * 5 15 20	+ + 10 * 5 15 20	Expected result has occurred.		
T03	-toString2()	tree = + + 10 * 5 15 20	10 5 15 * + 20 +	Expected result has occurred.		
T04	Creating new expression tree with constructor tree = / + + * - 3 5 3 7 11 6	tree = / + + * - 3 5 3 7 11 6	Construct a tree.	Expected result has occurred.		
T05	-eval()	tree = / + + * - 3 5 3 7 11 6	2	Expected result has occurred.		
T06	-toString()	tree = / + + * - 3 5 3 7 11 6	/ + + * - 3 5 3 7 11 6	Expected result has occurred.		
T07	-toString2()	tree = / + + * - 3 5 3 7 11 6	3 5 - 3 * 7 + 11 + 6 /	Expected result has occurred.		
T08	Creating Scanner with "- / * + 1 3 4 2 3" expression	scan object corresponding to "- / * + 1 3 4 2 3"	Scanner is created.	Expected result has occurred.		
T09	readExpressionTree( scan )	scan	Construct a tree and returned.	Expected result has occurred.		
T10	-eval()	tree = - / * + 1 3 4 2 3	5	Expected result has occurred.		
T11	-toString()	tree = - / * + 1 3 4 2 3	- / * + 1 3 4 2 3	Expected result has occurred.		
T12	-toString2()	tree = - / * + 1 3 4 2 3	1 3 + 4 * 2 / 3 -	Expected result has occurred.		


### Postfix Input Constructors Test:

Test ID	Scenario	Test Data	Expected Result	Actual Result	Pass	Fail
T01	Creating tree with given prefix expression.	tree = 10 5 15 * + 20 +	Construct a tree.	Expected result has occurred.		

### Methods Test for Postfix input:

Test ID	Scenario	Test Data	Expected Result	Actual Result	Pass	Fail
T01	-eval ( )	tree = 10 5 15 * + 20 +	105	Expected result has occurred.		
T02	-toString()	tree = 10 5 15 * + 20 +	++ 10 * 5 15 20	Expected result has occurred.		
T03	-toString2()	tree = 10 5 15 * + 20 +	10 5 15 * + 20 +	Expected result has occurred.		
T04	Creating new expression tree with constructor tree = 3 5 - 3 * 7 + 11 + 6 /	tree = 3 5 - 3 * 7 + 11 + 6 /	Construct a tree.	Expected result has occurred.		
T05	-eval()	tree = 3 5 - 3 * 7 + 11 + 6 /	2	Expected result has occurred.		
T06	-toString()	tree = 3 5 - 3 * 7 + 11 + 6 /	/ + + * - 3 5 3 7 11 6	Expected result has occurred.		
T07	-toString2()	tree = 3 5 - 3 * 7 + 11 + 6 /	3 5 - 3 * 7 + 11 + 6 /	Expected result has occurred.		
T08	Creating Scanner with "1 3 + 4 * 2 / 3 -" expression	scan object corresponding to "1 3 + 4 * 2 / 3 -"	Scanner is created.	Expected result has occurred.		
T09	readExpressionTree( scan )	scan	Construct a tree and returned.	Expected result has occurred.		
T10	-eval()	tree = 1 3 + 4 * 2 / 3 -	5	Expected result has occurred.		
T11	-toString()	tree = 1 3 + 4 * 2 / 3 -	- / * + 1 3 4 2 3	Expected result has occurred.		
T12	-toString2()	tree = 1 3 + 4 * 2 / 3 -	1 3 + 4 * 2 / 3 -	Expected result has occurred.		

### Exception Input Test:

Test ID	Scenario	Test Data	Expected Result	Actual Result	Pass	Fail
T01	Attempt to build tree with invalid input tree = - + - g t u 4 6	tree = - + - g t u 4 6	Invalid expression : - + - g t u 4 6	Expected result has occurred.		

### Class Diagram :

\*Class diagram is inserted to the homework file.\*

### Running Commend and Result:

\*Running command and result stage is inserted to the homework file.\*