Binary Tree Expression Parser

Group Project 3

Created by:

Isaac Hodge

Angel Hristov

Adrian Ronchetto

For the 3rd and final project, the group decided to create an infix expression parser using a binary expression tree. The work was split accordingly among the members of the team. Adrian Ronchetto created the Treebuilder and BTNode class. The code takes care of the precedence, parse the infix expression, and populates it into an expression tree. Isaac Hodge created the Evaluation class that took care of the calculations and the evaluation of the expression tree. He also worked on making sure the program is flexible and spaces between operands and operators are not a problem. Angel Hristov created the main code that reads from the input file and this report.

This project uses a hash map data structure to assign an order of precedence, stack data structure to convert the infix expression into postfix and binary tree data structure to populate the tree with operators as nodes and operands as leaf-nodes.

The system could be improved by organizing the code better and making it easier to read and understand. For example, our code’s infixToTree method ignores whitespace from the input expression so the code can be flexible. However, if we were to put it in a new method for removing whitespace, it could add a bit more clarity to the methods.

Test Case 1

Input:

1+2\*3

2+2^2\*3

1==2

1+3 > 2

(4>=4) && 0

(1+2)\*3

2%2+2^2-5\*(3^2)

Output:

7

14

0

1

0

9

-41

Operations Used: (Based on the operators in the input)

Test Case 2

Input:

24 -18\*2

57 || 0

27/2^3

((2+2)\*2)^2

6 > 5 && 4 > 5

3 <= 3

17/0

Expected Output:

-12

1

3

64

0

1

Exception in thread "main" java.lang.UnsupportedOperationException: Cannot divide by zero

Operations Used: (Based on the operators in the input)

UML Chart

Diagram

Description automatically generated