Project 3A

Binary Tree Infix Expression Parser

A logo with a black background

Description automatically generated

CPT-287-R82-Intro to Data Struct With Java-Summer 2024

Developers

Mitchell Jones

Published: 07/23/2024

System Design

Explanation of the system

Expressions are parsed into infix binary expression trees. Expressions are inputted using a text file, and are then parsed into stacks of nodes. One stack of operator nodes, and another stack of operand nodes, evaluate the expression into a tree. In order operation is then used to evaluate through the tree and return the solution. Multiple expressions can be evaluated using the text file.

Data structures

Stacks: Used for storing operator and operand nodes during parsing and tree building.

Role: Manages the order of operations and creates the binary infix expression tree.

StringBuilder: Used for building multi-digit numbers and multi-character operators.

Role: Allows efficient concatenation of digits when parsing numbers.

Main Structure: The Binary Tree Infix Expression Parser is broken up into three main parts. The first main part of the system is the Node class. This handles both numbers and operators, as well as subtree building. Binary tree infix parsing is the next main portion of the program. This builds out the Binary Tree, by parsing the expression into nodes, and creating subtrees based on precedence. The final part of the code evaluates through the binary tree using in order traversal, returning the solution to the expression.

UML Diagram

A screenshot of a computer

Description automatically generated

Test Cases

Case One Case Two

*expressions.txt expressions.txt*

A screenshot of a computer

Description automatically generated A number and symbols on a black background

Description automatically generated

*Terminal Results*  *Terminal Results*

A screenshot of a computer program

Description automatically generated A screenshot of a computer

Description automatically generated

Contributions

Mitchell Jones

Implemented Node class, as well as the main parsing algorithm, and the recursive in order evaluation. Created the input file system, and error handling. Responsible for designing, testing, documentation and implementing the entirety of the project.

Future Improvements

The Binary Tree Infix Parser can evaluate a large majority of basic expressions. It however it cant evaluate anywhere near as many expressions as a calculator. There are many more advanced math operations that could be implemented into the system. Trigonometric functions, or even decimal number handling could be added in. Adding in a dynamic way to visually build out the expression tree in the consol would also be good to show the user. The last major improvement that could be made in the future is incorrect parentheses handling, among other error handling.

A black background with red text

Description automatically generated