Project 2 Report

Infix-Expression-Parser Using Stacks

A logo with a black background

Description automatically generated

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

Developers

Mitchell Jones

Published: 07/12/2024

System Design

Explanation of the system

The Infix Expression Parser is designed to read mathematical expressions in infix notation from a file, parse them, and evaluate the results. It handles various operators with different precedencies and supports parentheses for grouping.

Data structures

Stacks: Used for storing operators and operands during parsing and evaluation.

Role: Manages the order of operations and handles nested expressions.

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

Role: Allows efficient concatenation of digits when parsing numbers.

Main Structure: The infix-expression-parser uses two different stacks, and two different string builders to parse the input expressions. The main algorithm scans through every character of the expression string. It uses a string builder to create multi-digit characters and adds all operands to an Integer Stack. Non-digit characters are created into an instance of the Operator class. Each operator has a symbol, precedence, and operation. A string builder is used to create Multi-Character operators ( >=, <=, ==, !=, &&, || ). The main algorithm handles order of operations, logical operators, and parentheses.

UML Diagram

A diagram of a computer

Description automatically generated

Test Cases

Case One Case Two

*expressions.txt expressions.txt*

A black background with white text

Description automatically generated A number and symbols on a black background

Description automatically generated

*Terminal Results*  *Terminal Results*

A black screen with white text

Description automatically generated A black screen with white text

Description automatically generated

Contributions

Mitchell Jones

Implemented main parsing algorithm using stacks. Developed the operator class and precedence logic. Created the input file system, and error handling. Responsible for designing, testing, documentation and implementing the entirety of the project.

Future Improvements

There are a few areas where this infix-expression-parser could be improved. The first is implementing more complex math operations. Operations like Sin, Cos, Tan, Log, etc. The project currently does not handle decimals either; adding support for float datatypes would add more use cases for the program. The error handling in this program is minimal and could definitely be improved. It would also be nice to have a GUI to easily parse and save expressions. Overall, my infix-expression-parser is a very solid program, but there is countless room to improve or redesign for more functionality.

A black background with red text

Description automatically generated