Team Project 3A

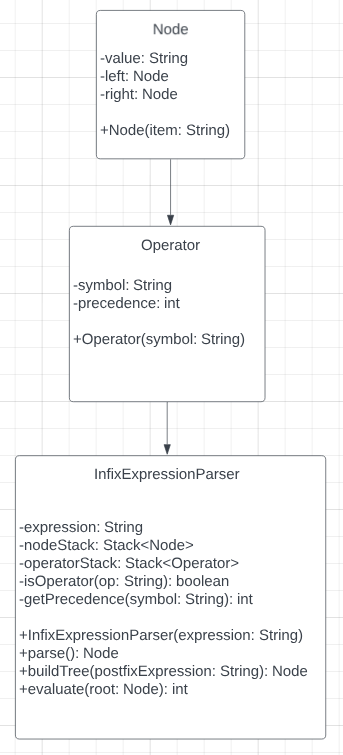
Binary Tree Infix Expression Parser

Andrew Lange, Mason Johnson, Steven Valet

**Design Explanation**

The system is separated into a InfixExpressionParser, Operator, and Main class. When the program is started, the Main class loads the file with the expressions on it. From there, it reads the current line and parses it into an array of characters. Then it removes all whitespace before continuing. Afterward, it runs through character-by-character and checks to see if the current character is a digit or operator, and pushes it to the stack accordingly, converting it from an infix expression to a postfix expression. When the time comes to build the tree, we once again remove all whitespace and create an Array of valid “tokens” to run through. These tokens are run through and are popped before creating a new Node with the operator token and setting its left and right children to those popped nodes. The new node is then pushed back to the stack, and once it is through all tokens, we begin to evaluate in a postfix manner. Once evaluated, we return the result. The process then repeats for every valid line of the text document.

UML Chart Diagram



**Test Cases**

Test case 1:

1+2\*3

Expected result:

Expression: 1+2\*3 Result: 7

Actual result:

Expression: 1+2\*3 Result: 7

Test case 2:

1==2

Expected result:

Expression: 1==2 Result: 0

Actual result:

Expression: 1==2 Result: 0

**Team Member Contributions**

**Andrew Lange -** InfixExpressionParser.java, UML Chart Diagram

**Mason Johnson -** Started report, Original Operator.java solution, Node.java

**Steven Valet -** Started repository, Main.java, Operator.java error correction, ParseInfixExpression.java error correction, code clean up

**Future Improvements**

There are a few things we could implement to improve this project. First would be error handling. We could handle a lot of different cases to ensure errors are accounted for. Another thing is optimization. We are sure there are different ways to make the code faster and more seamless. We could add a graphical user interface so the user has a more pleasing time inputting and retrieving the data. We could also add support for other operators such as unary and bitwise operators.