**CPT-287 Team Project 3A**

**Binary Tree Infix Expression Parser**

**-Team Members-**

Tu Nguyen

Kyle Molitor

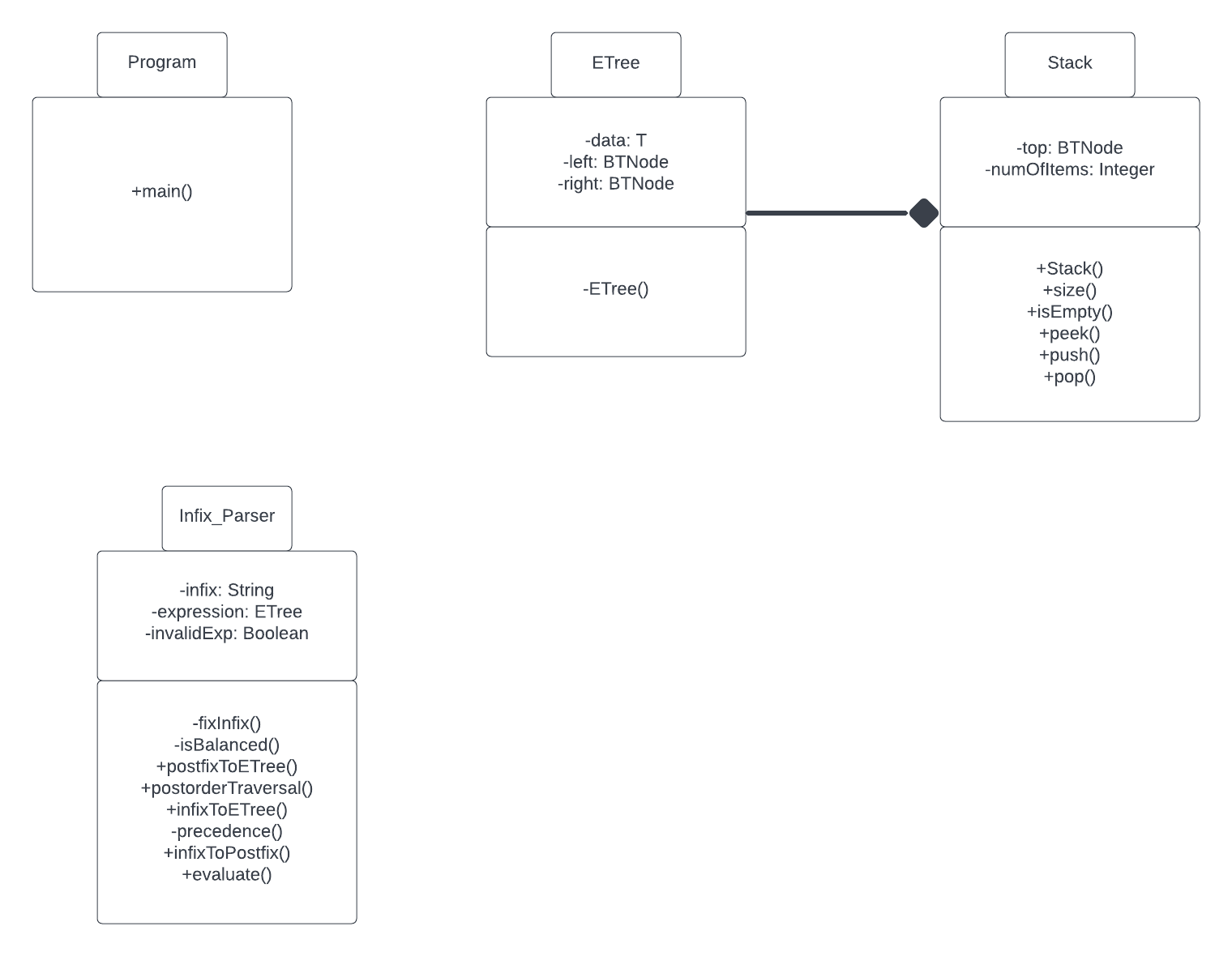
Josh Vander Veld

**-System Design-**

*Data Structures*

This system implements a Binary Tree and built in Java stack data structure. The Binary Tree is used to store expressions to be parsed. The stack data structure is used to traverse and parse the expressions into binary tree nodes. The Infix Parser class is used to parse any given expression read into a binary tree from an input file. Infix parser contains functions for parsing the input expression such as, isBalanced, evaluate, toString ext. The program class contains all the driver code to read in an input file and output the result.

*UML Class Diagram*

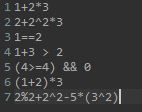


**-Test Cases-**

***Test Case 1:***

*Input File*

Input file contains expressions.



*Output Console*

Program successfully reads and gives correct answer.



Expected output:



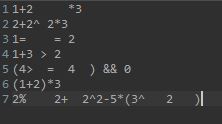
Actual output:



***Test Case 2:***

*User Input*

Input file expressions contain inconsistent spaces.



*Output Console*

Program successfully reads and corrects spacing.



Expected output:



Actual output:



**-Team Contributions-**

**Tu**: Contributed to the Program class for reading an input file and outputting results to console. Contributed to Infix\_Parser class. Responsible for writing the constructors, tree nodes, and the following methods, fixInFix, toString, IsBalanced, postfixToETree, postorderTraversal, deepCopy, infixToETree, and infixToPostfix method.

**Kyle**: Contributed to Program class for reading input from a text file. Contributed to the Infix\_Parser class. Wrote the evaluate method and modified the precedence function to support all operators needed in the assignment. Responsible for creating and writing this report as well as creating both test cases. Created and maintained GitHub project repository.

**Josh**: Created UML Class Diagram.

**-Future System Improvements-**

This system could be improved to support more operators and cases. Add more operators such as log, ln, or trig function and value such as I, e, pi. The infix parser could contain more methods and more support for all variations of expressions such as converting to and from infix, postfix, prefix expressions.

We could also add functionality for more advanced expressions with non-number values such as PI, sine, cosine, tangent.