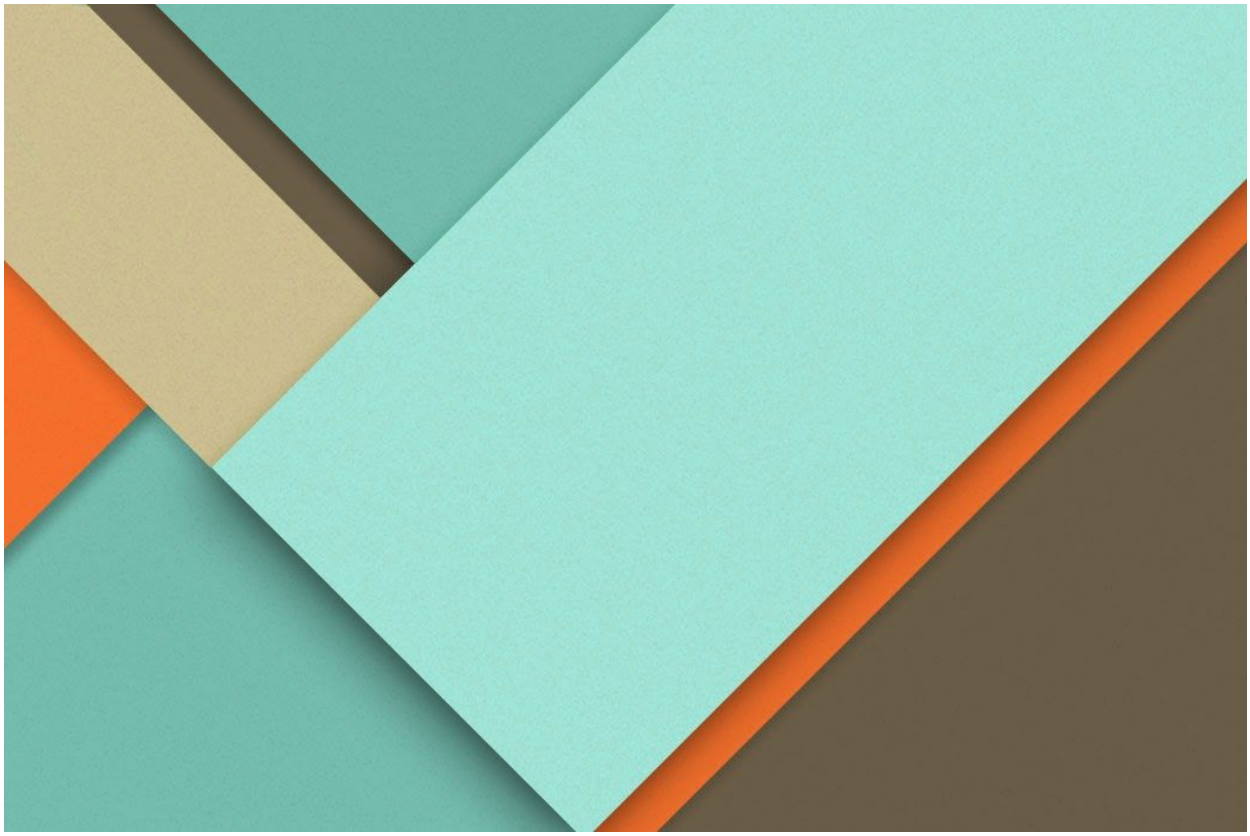


Team Project 3

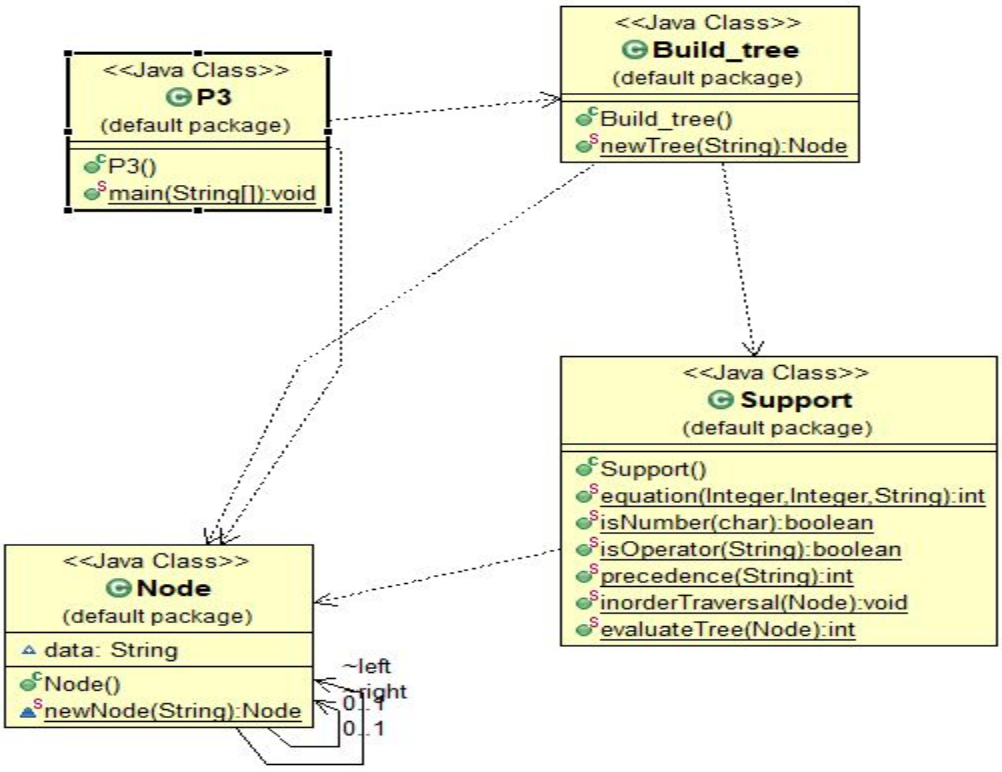
02.16.2021

Dalton Vining
Samuel Maynard
Shane Callaway

How we designed the system:

The system was based loosely off of the design used in Team Project 2 as the end result was attempting to solve the same problem through different methods. To this end we opted to reuse the Precedence, Equation, isNumber, and isOperator methods created by various team members. Samuel worked majorly on the algorithm to convert a input equation string into a tree, which was later refined through the combined efforts of Samuel and Shane. This method takes each character of the input string and, going through them one by one determines if it is an operator or number, and using two stacks for operators and Nodes builds a tree based on operator precedence. An open and closing parentheses was used to mark the start and end points of the equation. After the tree is finished being built the root node, which will be the only node remaining in the stack, is applied to a method that recursively goes through the tree and returns the result of each operator in order. The to_Tree method then returns the root node, for any further needed applications.

UML Class diagram:



Two test cases:

Test case one: 1 + 2 * 3, expected output: 7, actual output: 7

Test case two: (1 + 2) * 3, expected output: 9, actual output: 9

First, the equations are read from the file and converted into an array of strings. Then the expression is passed to the build_tree class and is converted into a tree with the help of the node class. Once the tree is created, a traversal method in the support class is called to output the expression by traversing the tree. After, the expression is evaluated. Parenthesis are treated almost as operators themselves, forcing new precedence. This is critical for these two examples, which are identical aside from parenthesis. In test case one, the * is evaluated first, and then the +. Because of the parenthesis, in test case two, the + is evaluated before the *. Once the evaluation is complete, the result is output to the console.

How we contributed:

Samuel developed the main method to turn an input string into a tree, which was then worked on, improved and cleaned up by the group. Samuel also implemented the file reading system that runs the project. Methods from the previous project that were easily reused for this project werer created and, as needed, slightly modified by group members to fit this particular project. Shane worked on the method to evaluate the tree and assigning references to needed methods to make it function as well as implementing the output.

Improvements:

One possible improvement would be a simple menu that could not only return the output of the equation, but return the equation in postfix, infix, or prefix notations. The menu could also give the option to output the height of the tree or even the number of operators.