Project 2A - Team1 Documentation

Overview

Team 2 (ie The Wookie Workgroup) created a command line tool that computes the result of infix expressions (Project2A). The group consists of Daniel Mitchel, Joshua Neustrom, and Chen Wang.

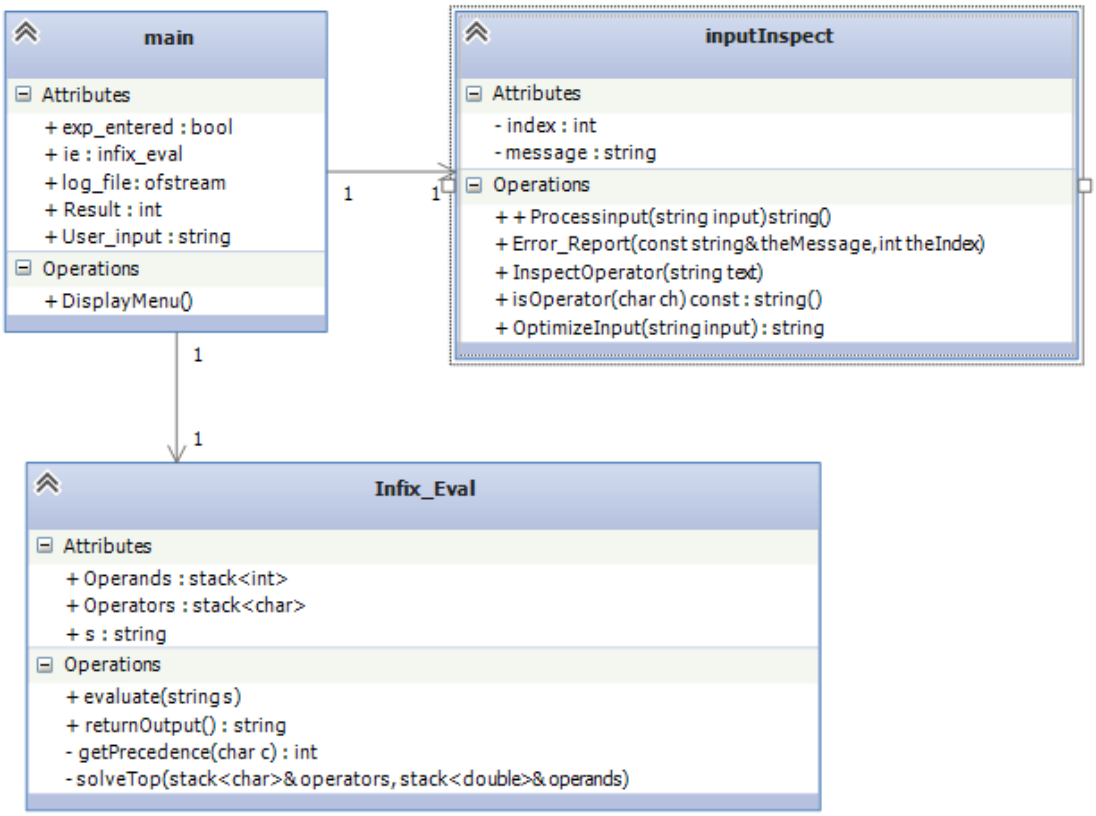
The report gives an overview of our solution including the following

1. Assumptions
2. UML Class Diagram
3. Github Project Link
4. Algorithm
5. Error Handling
6. Known Issues
7. Efficiency Analysis of Algorithms
8. References

Assumptions

1. No variables used (can accept ints)
2. -1 must be written as (-1)
3. No history stored past the last expression entered
4. Spaces are cleaned from the user expression
5. Errors caught in the pre-check reference the char number of that caused the error after spaces have been removed. (chars numbered starting at zero)
6. Division involves ints only
7. Base ten number system
8. Only accepts () and no {} [] for setting precedence
9. Calculation or Syntax Error results in answer to expression being cleared
10. User can enter another expression if the last one entered results in an error
11. Expression Error Clears all previous user input (to purge any bad data)
12. Prefix increment and decrement only (no 1++, or 1--)
13. Wookies rule

UML Class Diagram



Github Site

https://github.com/WookieWorkgroup/Project2/

Algorithm

0. User interface – continuous loop that displays a command line menu
1. Option 1 - Ask for Expression
    a. Store line
    b. Clean the string
        i. Optimize
            1. Eliminate Spaces
            2. Check for two operands in a row
            3. Check for Improper Paren position
        ii. Inspect operators
            1. Check for paren error

2. Check binary operator position
3. Check for incomplete operator expression
4. Check for two operators in a row

iii. Error messages include char position
iv. Return a cleaned string

c. Compute the expression
   i. Make expression postfix by putting elements in the operand and operator stacks
      1. Read in digit and add to operand stack
      2. Read in operand
         a. If expression start in - at beginning then has digit, make the digit a negative
         b. If expression has – after operator, then make the digit after the - symbol a negative
         c. Add operands to stack
            i. Check precedence
            ii. Higher precedence on bottom
   ii. Compute the expression using postfix
      1. Pop operand
      2. Unary – Pop one operand and push result
      3. Binary – Pop two operands and push the result
      4. Once operator stack is empty, pop and return the result from the operands stack

2. Option 2 – Show the previous expression entered by user (if any)
3. Option 3 – Show the previous result (if any)
4. Option 4 – Clear – user input, results, ect
5. Option 5 – Exit (close console)

Error Handling

1. Bad Input – Error message retuned and user sent back to the main menu
2. Logging – Log.txt contains record of actions completed to help troubleshoot
3. Error before calc – if caught by our pre-check, an error message and char number is returned to the user (char numbering starts at zero and ignores spaces)
4. Error during calculations – Error message returned along with error number (each exception in the calculations has a specific error number)

Known Issues

1. Division returns result an int. Possible to get divide by zero error due to ints that would not occur with doubles. System throws exception and prompts user for new input

Efficiency of Algorithms

1. Evaluate – O(n) or aprox T(2n) – one loop through string for convert to postfix, one loop through stacks to compute result

2. Inspect Input – O(n) or aprox T(2n) – One loop to optimize and one loop to look for operators (loops are not nested)

References

1. Evaluate algorithm - https://en.wikipedia.org/wiki/Shunting-yard_algorithm
2. Method for eliminating user input that causes overflow - http://stackoverflow.com/questions/3826281/how-do-i-make-a-c-program-that-filter-out-non-integers
3. Post Fix, Syntax Error, and Infix to Postfix from class Lectures used as a starting reference
4. The Force