/\*\*

\* This program will implement a GUI application for reading in a file containing employee information, and sorting them based on name, age, pay, etc...

\*

\* Name: Nabeel Hussain

\* Class: CMSC350

\* Professor: Didier Vergamini

\* Project 1

\* Date: 11/4/2016

\*

\* **@author** Nabeel Hussain

\*/

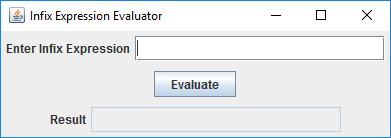
This program for project 1 involves implementing an infix expression evaluator, in which the user enters an expression to be evaluated, and the result is shown using a GUI. The expression evaluator uses two stacks, where one holds the operators [“+, -, \*, /, (, )”], and the second stack holds the operands (numbers and calculation results) during the evaluation process of the expression.

My program has three classes. The first class is the data element class, that implements the stack data structure. It uses an ArrayList to push and pop elements while evaluating the expression.

The second class is the data manager class, which holds the methods that will be used by my GUI class to perform the evaluation. This will include a method to evaluate the expression, which will follow the pseudo code given in the instructions to evaluate it. It will also contain a method that performs the four basic calculations, +, -, \*, / depending on the two operands given and the specified operator. It will also contain a third method, which will determine whether an operator has higher or equal precedence compared to another operator. The latter two methods will be used in the evaluateExpression() method to solve the expression.

The third class is used for the GUI, which will contain the buttons and a text field output to meet the functional requirements for this assignment.

My GUI:

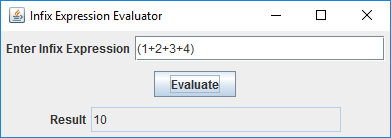


**Test Plan:**

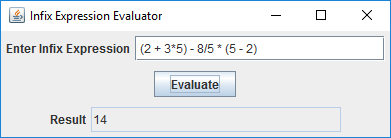
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Input** | **Expected Output** | **Did Test Pass?** |
| 1 | **Expression = (1+2+3+4)**  \*No blank spaces separating each token\* | 10 | Y |
| 2 | **Expression = (2 + 3\*5) - 8/5 \* (5 - 2)**  \*Leading blank space\*  \*Random spaces between tokens in the expression\*  \*All operators are used\* | 14 | Y |
| 3 | **Expression = 10-5\*10/5**  \*Blank space at the end of the expression\*  \*No right or left parenthesis used\*  \*Tests precedence. Multiplication is done first, then division, and then the subtraction.\* | 0 | Y |
| 4 | **Expression = ( ((100/2)/5 +2) - (5\* 2 ) )**  \*Multiple right and left parenthesis used  \*Random spaces between tokens in the expression\* | 2 | Y |
| 5 | **Expression = (2 + 3 \*5) - 8/0 \* (5 - 2)**  \*Random spaces between tokens in the expression\* \*Contains a division by zero\* | JOption Pane Show Message Dialog:  "Error. You cannot divide by 0." | Y |

**Screen shots of successful compilation and running for all test cases**

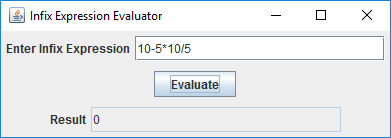
**Test Case 1:**



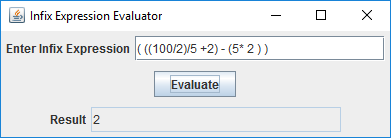
**Test Case 2:**



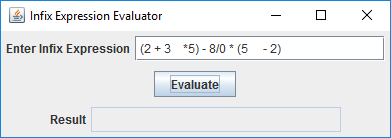
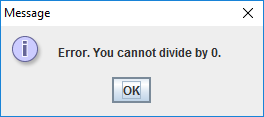
**Test Case 3:**



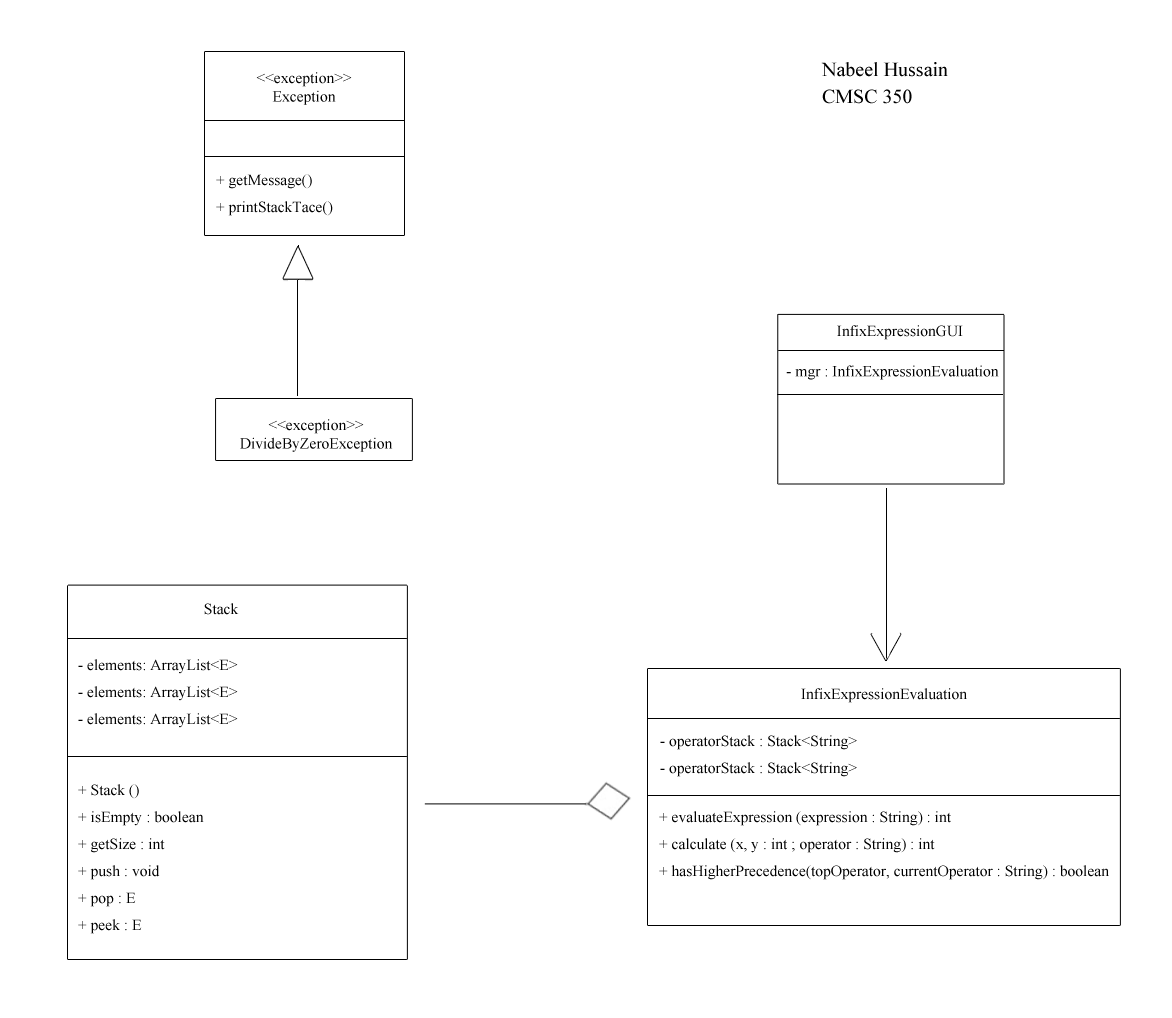
**Test Case 4:**



**Test Case 5:**



**UML Diagram:**



**Lessons Learned:**

While working on Project 1, the majority of the concepts applied in this program were a review to me more than anything. The implementation and use of stacks and ArrayLists are all concepts I have learned before, and this project helped me to reinforce and refresh my memory. The pseudo code given to evaluate the expression was very helpful and descriptive, allowing me to complete the assignment fairly easily, without too many complications. It did take me a bit of time to figure out how to tackle the issue of reading in an expression and separating each token properly, but after reading up more about the StringTokenizer class, I was able to use the operators as the delimiters to separate each element in the expression to individual tokens. Overall, this assignment helped me to refresh a lot of the concepts I have learned in my previous programming classes, such as using stacks, handling exceptions, creating GUI’s, among other things.