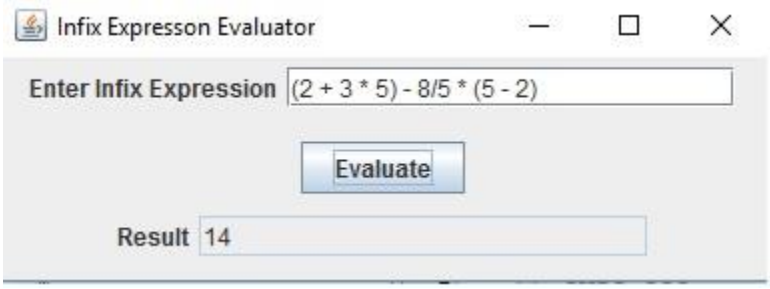
CMSC 350 Project 1

The first programming project involves writing a program that evaluates infix expressions of unsigned integers using two stacks. The program should consist of three classes. The main class should create a GUI that allows the user input an infix expression and displays the result. The GUI should look as follows:



The GUI must be generated by code that you write. You may not use a drag-and-drop GUI generator.

The second class should contain the code to perform the infix expression evaluation. The pseudocode for performing that evaluation is shown below:

```
tokenize the string containing the expression
while there are more tokens
     get the next token
     if it is an operand
          push it onto the operand stack
     else if it is a left parenthesis
          push it onto the operator stack
     else if it is a right parenthesis
          while top of the operator stack not a left parenthesis
               pop two operands and an operator
               perform the calculation
               push the result onto the operand stack
     else if it is an operator
          while the operator stack is not empty and
               the operator at the top of the stack has higher
               or the same precedence than the current operator
               pop two operands and perform the calculation
               push the result onto the operand stack
          push the current operator on the operators stack
while the operator stack is not empty
     pop two operands and an operator
     perform the calculation
     push the result onto the operand stack
the final result is a the top of the operand stack
```

Be sure to add any additional methods needed to eliminate any duplication of code.

Your program is only expected to perform correctly on syntactically correct infix expressions that contain integer operands and the four arithmetic operators `+ - * /`. It should not, however, require spaces between tokens. The usual precedence rules apply. The division performed should be integer division. A check should be made for division by zero. Should the expression contain division by zero, a checked exception `DivideByZero` should be thrown by the method that performs the evaluation and caught in the main class, where a `JOptionPane` window should be displayed containing an error message.

You are to submit two files.

1. The first is a `.zip` file that contains all the source code for the project, which includes any code that was provided. The `.zip` file should contain only source code and nothing else, which means only the `.java` files. If you elect to use a package the `.java` files should be in a folder whose name is the package name.
2. The second is a Word document (PDF or RTF is also acceptable) that contains the documentation for the project, which should include the following:
    a. A UML class diagram that includes all classes you wrote. Do not include predefined classes. You need only include the class name for each individual class, not the variables or methods
    b. A test plan that includes test cases that you have created indicating what aspects of the program each one is testing
    c. A short paragraph on lessons learned from the project

Grading Rubric:

Criteria	Meets	Does Not Meet
Design	**5 points**	**0 points**
	GUI is hand coded and matches required design (1)	GUI is generated by a GUI generator or does not match required design (0)
	Supplied algorithm is used (1)	Supplied algorithm is not used (0)
	Code duplication is eliminated (1)	Contains duplicated code (0)
	Contains separate class for expression evaluation (1)	Does not contain separate class for expression evaluation (0)
	Contains checked exception class (1)	Does not contain checked exception class (0)
Functionality	**10 points**	**0 points**
	Produces correct value for all operators(3)	Does not produce correct value for some operators (0)
	Correctly parses expressions without space delimiters (2)	Does not correctly parse expressions without space

		delimiters (0)
	Correctly implements precedence (2)	Does not correctly implement precedence (0)
	Correctly evaluates parenthesized expressions (2)	Does not correctly evaluate parenthesized expressions (0)
	Detects division by zero (1)	Does not detect division by zero (0)
Test Cases	**5 points**	**0 points**
	All operators included in test cases (1)	Some operators not included in test cases (0)
	Test cases include expressions without spaces (1)	Test cases don't include expressions without spaces (0)
	Test cases include cases to test precedence (1)	Test cases do not include cases to test precedence (0)
	Test cases include cases with parentheses (1)	Test cases do not include cases with parentheses (0)
	Test cases include a case to test division by zero (1)	Test cases do not include a case to test division by zero (0)
Documentation	**5 points**	**0 points**
	Correct UML diagram included (2)	Correct UML diagram not included (0)
	Lessons learned included (2)	Lessons learned not included (0)
	Comment blocks with class description included with each class (1)	Comment blocks with class description not included with each class (0)
Overall Score	**Meets**	**Does not meet**
	16 or more	**0-15**