CMSC 350 Project 1

**Class Diagram**

SyntaxError

ExpressionConverter

CMSC350Project1GUI

**Test Cases 1: All operators included in test cases**

Test Case 1a:

1. Type in input text field: 2 4 \* 1 - 2 3 / +
2. Press Postfix to Prefix Button.
3. The Result text field should display + - \* 2 4 1 / 2 3.

Test Case 1b:

1. Type in input text field: +6 -\* 4 / 100 2 50
2. Press Prefix to Postfix Button.
3. The Result text field should display 6 4 100 2 / \* 50 - +.

**Test Cases 2: Test Cases include expressions without spaces**

Test Case 2a:

1. Type in input text field: 42\*
2. Press Postfix to Prefix Button.
3. The Result text field should display \* 4 2.

Test Case 2b:

1. Type in input text field: +35
2. Press Prefix to Postfix Button.
3. The Result text field should display 3 5 +.

**Test Cases 3: Test Cases include cases that pop an empty stack**

Test Case 3a:

1. Type (nothing) in input text field:
2. Press Prefix to Postfix Button.
3. The Result text field should display error message stating “Expression is required”.

Test Case 3b:

1. Type (nothing) in input text field:
2. Press Postfix to Prefix Button.
3. The Result text field should display error message stating “Expression is required”.

**Test Cases 4: Test cases include cases which result in a non-empty stack**

Test Case 4a:

1. Type in input text field: 10 + 20
2. Press Postfix to Prefix Button.
3. The Result text field should display error message stating “Invalid expression”.

Test Case 4b:

1. Type in input text field: 5 10 \*
2. Press Prefix to Postfix Button.
3. The Result text field should display error message stating “Invalid expression”.

**Lessons Learned**

From this project, I learned more about data structures and how to usefully and successfully implement a Stack. I learned more about how to use the two operations of a Stack, which include pop and push. I also learned how to use more data structures such as LinkedLists and Queues. I know have a better idea of the general purposes each data structures serve in a program. Writing this code has enabled me to have a good grasp of the concept of how the stack is a LIFO-queue and how LinkedLists are memory efficient, dynamic to work with, and helpful in tracking information. I learned how to combine various data structures to create a program which converts expressions from postfix form to prefix and vice-versa. It was interesting and invaluable to learn how these various data structures interact, collaborate, and depend on each other. This project not only helped me learn how data structures such as stacks, LinkedLists, and Queues work, but also helped me better understand how to create an effective GUI. From this project, the advantages of using a LinkedList became clear in how using LinkedLists facilitated my approach to making effective use of Stacks and Queues.