Project #1

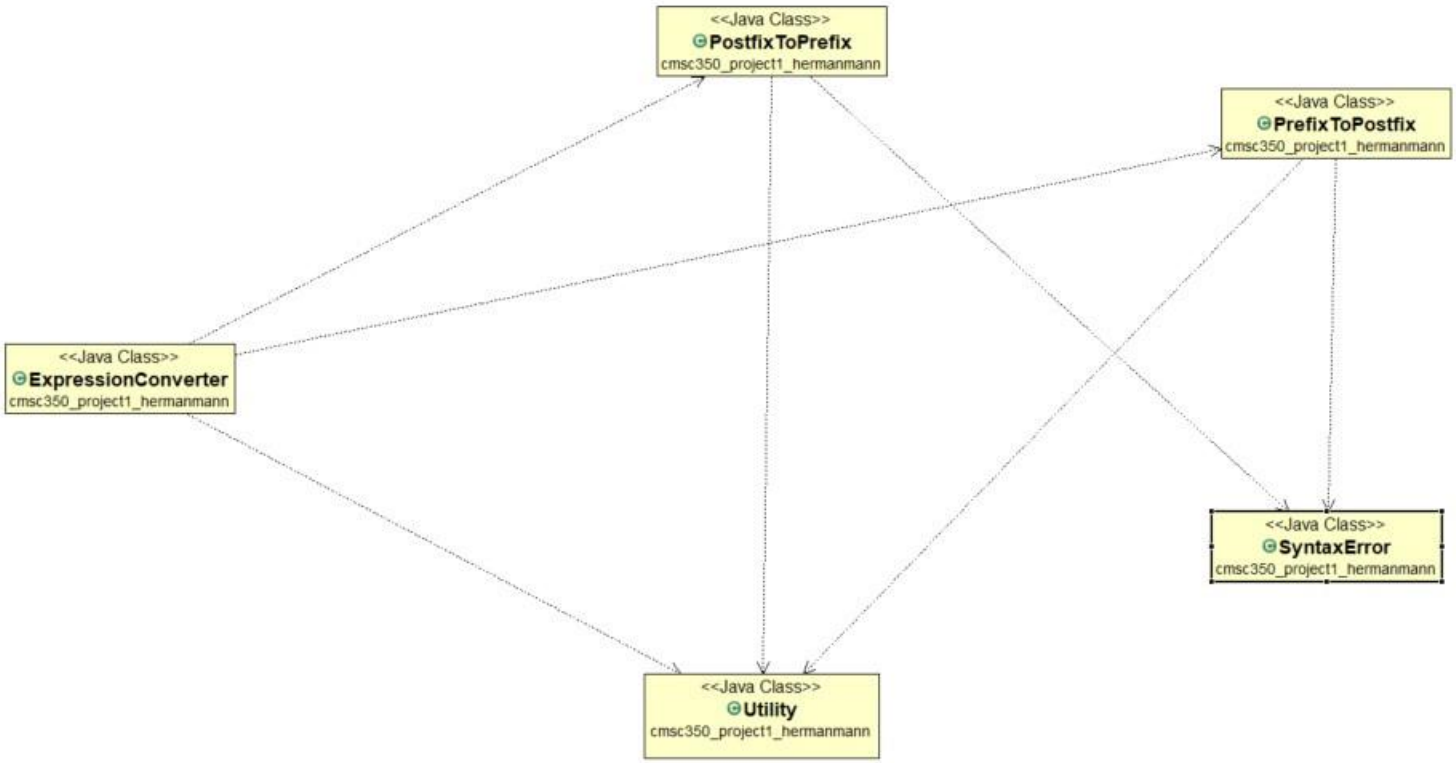
Name - Herman Mann

Course - CMSC 350

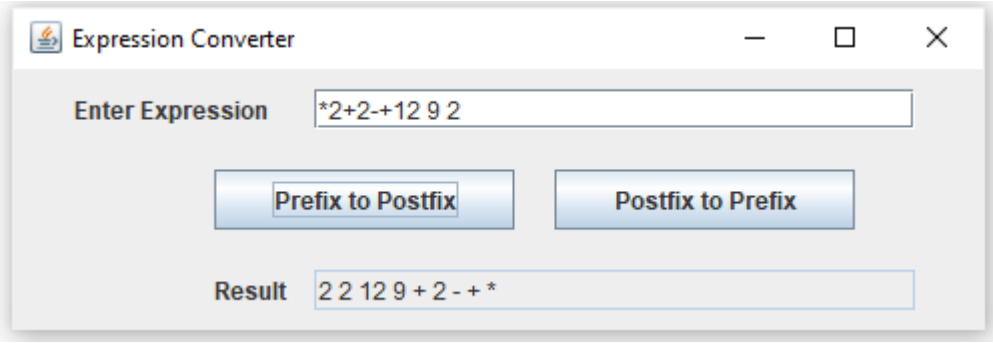
Date - 01/24/2022

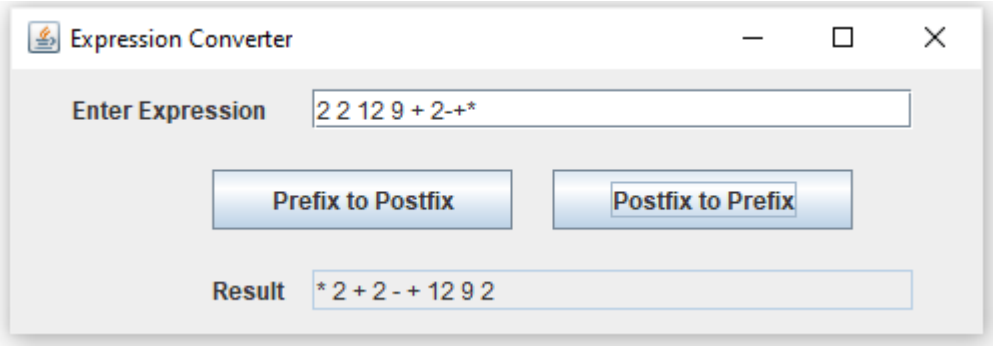
Table of Contents

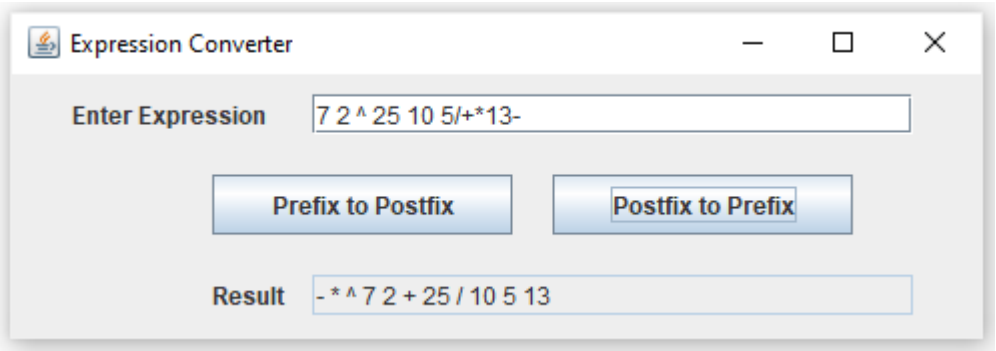
UML Diagram

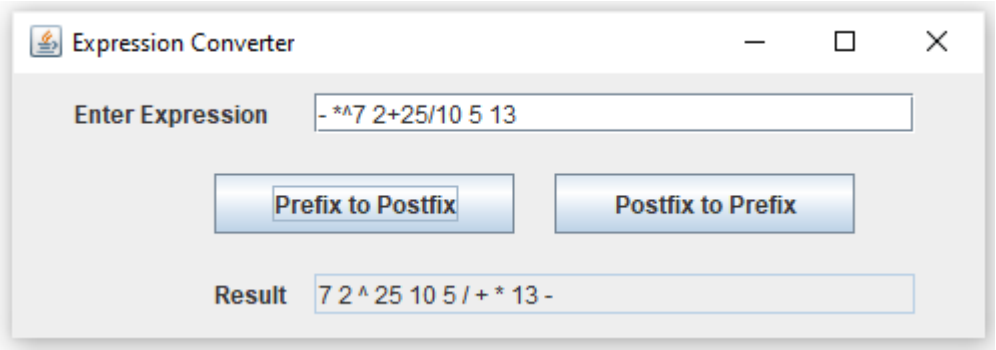


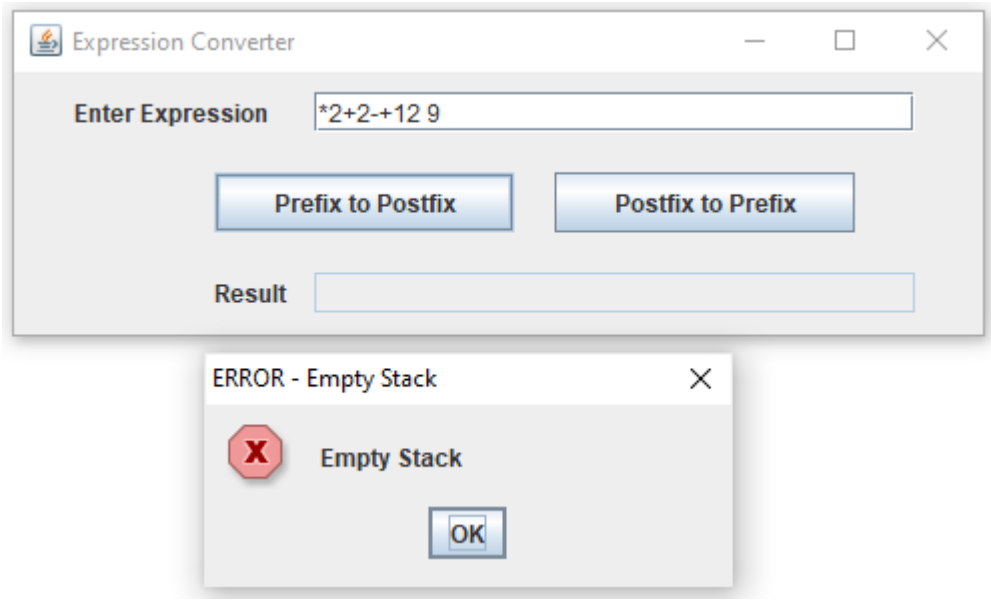
Test Plan

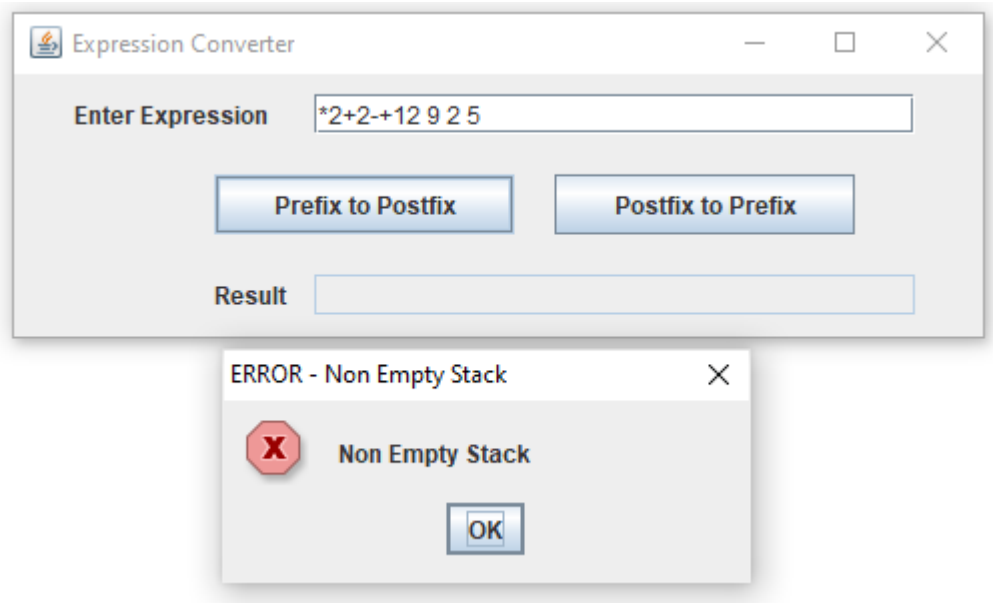
Test Case # 1	
Description	Prefix expression to Postfix Conversion Without spaces unless they are consecutive operands
Input	Prefix expression as an input Without spaces unless they are consecutive operands *2+2-+12 9 2
Expected Output	Postfix expression is created (Required The output expressions should always have a space between all symbols) 2 2 12 9 + 2 - + *
Actual Output	Postfix expression is created with a space between all symbols 2 2 12 9 + 2 - + *
Pass	**Pass**
Screenshots	

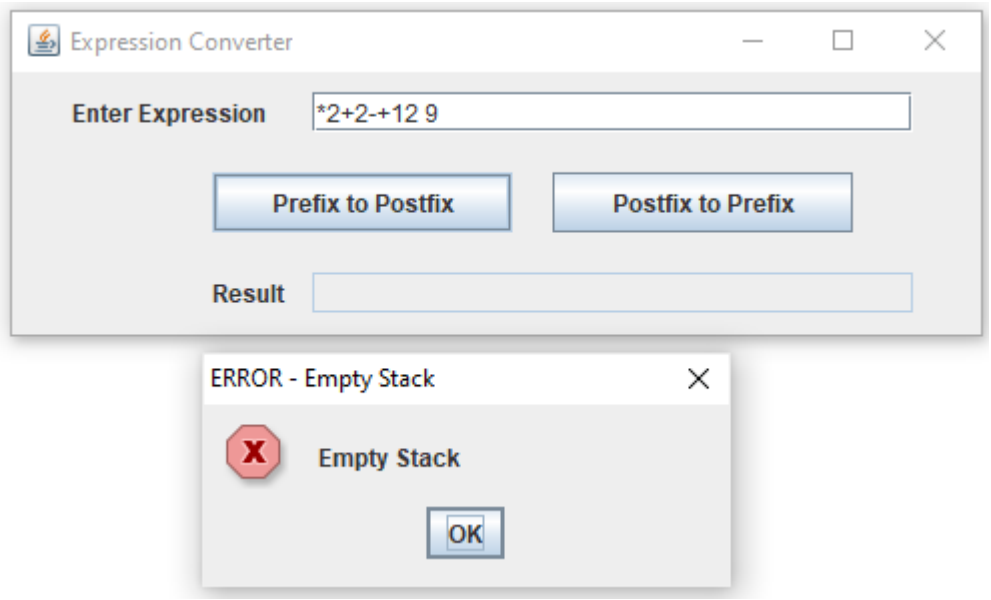
Test Case # 2	
Description	Postfix expression to Prefix Conversion Without spaces unless they are consecutive operands
Input	Postfix expression as an input Without spaces unless they are consecutive operands 2 2 12 9 + 2-+*
Expected Output	Prefix expression is created (Required The output expressions should always have a space between all symbols) * 2 + 2 - + 12 9 2
Actual Output	Prefix expression is created with a space between all symbols * 2 + 2 - + 12 9 2
Pass	**Pass**
Screenshots	

Test Case # 3	
Description	Include All the operators in Postfix expression to Prefix Conversion Without spaces unless they are consecutive operands
Input	Include All the operators in the Postfix expression as an input Without spaces unless they are consecutive operands 7 2 ^ 25 10 5/+*13-
Expected Output	Prefix expression is created (Required The output expressions should always have a space between all symbols) - * ^ 7 2 + 25 / 10 5 13
Actual Output	Prefix expression is created with a space between all symbols - * ^ 7 2 + 25 / 10 5 13
Pass	**Pass**
Screenshots	

Test Case # 4	
Description	Include All the operators in Prefix expression to Postfix Conversion Without spaces unless they are consecutive operands
Input	Include All the operators in the Prefix expression as an input Without spaces unless they are consecutive operands - *^7 2+25/10 5 13
Expected Output	Postfix expression is created (Required The output expressions should always have a space between all symbols) 7 2 ^ 25 10 5 / + * 13 –
Actual Output	Postfix expression is created with a space between all symbols 7 2 ^ 25 10 5 / + * 13 –
Pass	**Pass**
Screenshots	

Test Case # 5	
Description	Popup an empty stack exception
Input	Prefix expression as an input Without spaces unless they are consecutive operands *2+2-+12 9 (It has one less operand)
Expected Output	Message popup dialog with empty stack exception message
Actual Output	Message popup dialog with empty stack exception message
Pass	**Pass**
Screenshots	

Test Case # 6	
Description	Popup a non-empty stack exception
Input	Prefix expression as an input Without spaces unless they are consecutive operands *2+2-+12 9 2 5 (It has an extra operand 5 at the end)
Expected Output	Message popup dialog with non-empty stack exception message
Actual Output	Message popup dialog with non-empty stack exception message
Pass	**Pass**
Screenshots	

Test Case # 7	
Description	Popup an empty stack exception
Input	Blank input
Expected Output	Message popup dialog with “Please enter a valid expression” message
Actual Output	Message popup dialog with “Please enter a valid expression” message
Pass	**Pass**
Screenshots	

Lessons Learned

Upon completing project 1 for this course, I have got to learn and experience a lot about the different parts of the Java Programming Language in more depth and detail. For example, I really liked how I gained more exposure and experience into learning about the Stack data structure with its corresponding methods that were used within this project overall and how it specifically works as noted that Stacks operate in (LIFO) Last-In_First-Out manner. Also, I learned about the different conversions such as converting Prefix to postfix expressions and vice versa and how to really go about doing this through various pieces of Java Programming code itself. Moreover, I learned about how to really properly design a Java GUI (Graphical User Interface) by hand as this was my first-time experience in doing it. I gained skills in using the Stack which is heavily used in Java and the overall (OOP) object-oriented programming world, and finally learned about creating a new checked exception class such as the one I created for this project which is the SyntaxError checked exception. Also, I learned about the EmptyStackException since I had to throw this exception within my program for the project whenever the stack was empty when using the exception when I was converting prefix to postfix and postfix to prefix. In fact, I got to experience with using different javax files that needed to be imported specifically in the creation of the Expression Converter GUI needed for this project (examples include, java.awt.event.ActionListener and java.awt.event.ActionEvent). This project has made me more confident in programming in Java, but there is still a lot more for me to learn about and delve more into in order to be a champion in this language and overall mainly in the field of Computer Science.