Project 1

William Crutchfield

IntelliJ Ultimate 2017.1.3

Windows 10

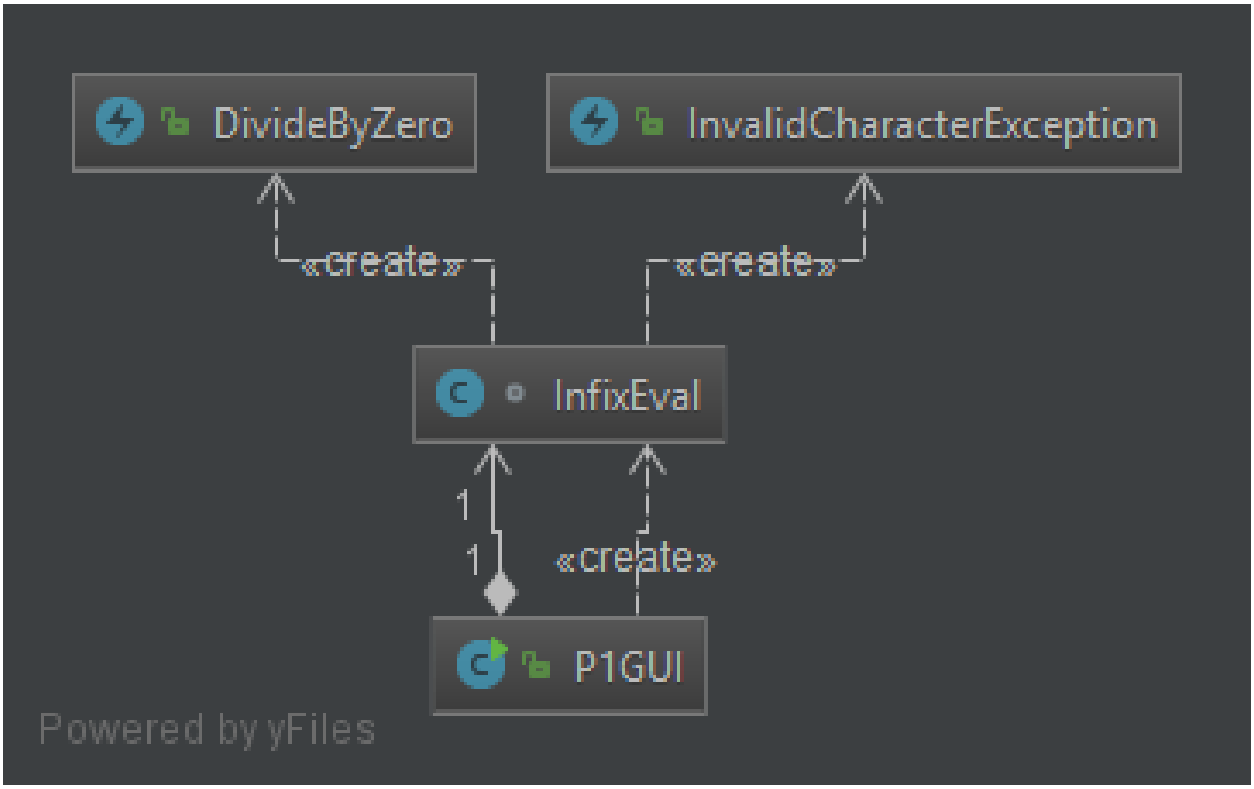
Contents

Assumptions

In this project, I have not assumed anything. In theory, I believe I have caught all errors that may occur when processing an input. The biggest one being EmptyStackException. When caught, this exception will simply tell the user that they have not entered a valid equation and should double check their input. An Invalid input could be something such as “2++2” or “(2+2()”.

However, I have implemented the logic to check for valid characters. This means that if someone entered “2&4” or “2a4”, a custom exception will be thrown. I have also ensured that whether the equation has spaces “2 * 2 + 3” or not “2*2+3” the output will remain the same. The output will also be correct even if you the user inconsistently inputs an equation with spaces “2* 2 +3”.

UML Diagram

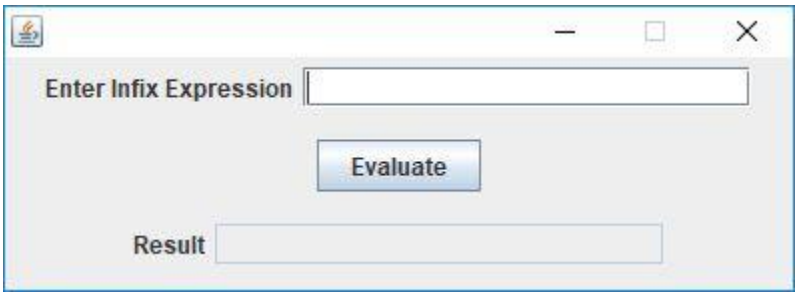

DivideByZero
InvalidCharacterException
«create»
«create»
InfixEval
1
1
«create»
P1GUI
Powered by yFiles

Test Cases

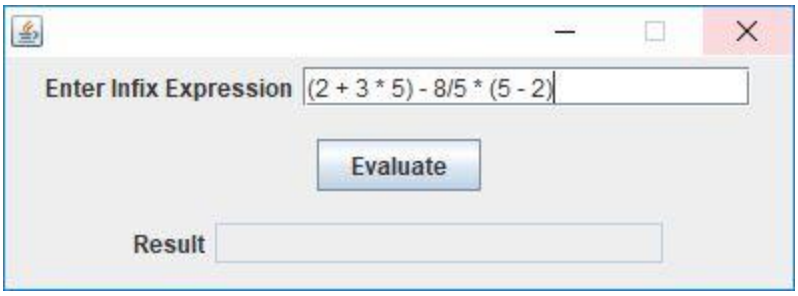
Aspect Tested	Input	Expected Output	Actual Output	Test Outcome
“+” operator, equation w/o spaces	2+2	4	4	Passed
“*” operator, equation with spaces	2*4	8	8	Passed
“-” operator, parentheses, equation with inconsistent spaces	4 -2 - (9-3)	-4	-4	Passed
“/” operator, parentheses	(2 + 3 * 5) - 8/5 * (5-2)	14	14	Passed
Empty input		Error: Expression Required	Error: Expression Required	Passed
Invalid Character	2 + 3 * 5 & 2	Error: Invalid Character	Error: Invalid Character	Passed
Division by 0	(9 + 17) * (8/0) + 37	Error: Division by 0	Error: Division by 0	Passed
Invalid Expression	(2+2()	Error: Invalid Expression	Error: Invalid Expression	Passed

Final Product

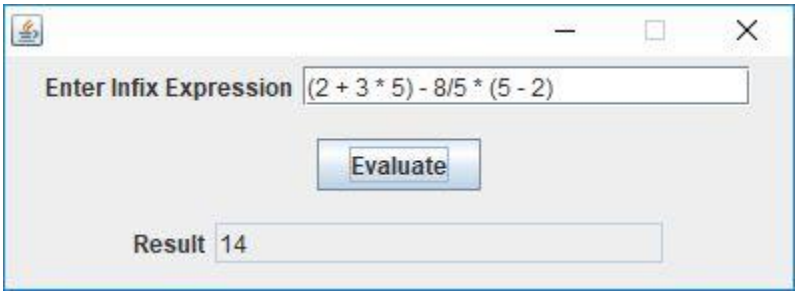
Upon opening the program, a window will open that looks like this.



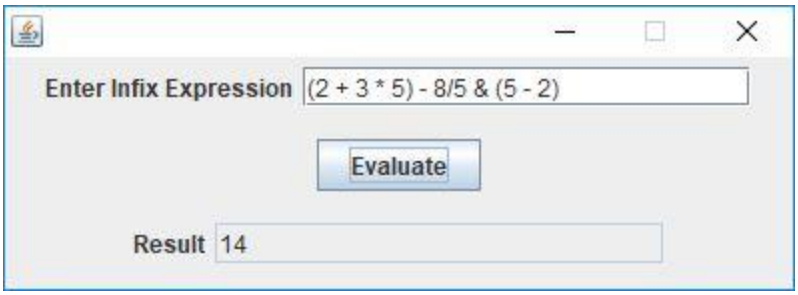
We can then type an Infix Expression into the Text Field. I have decided to use the equation from the instructions.



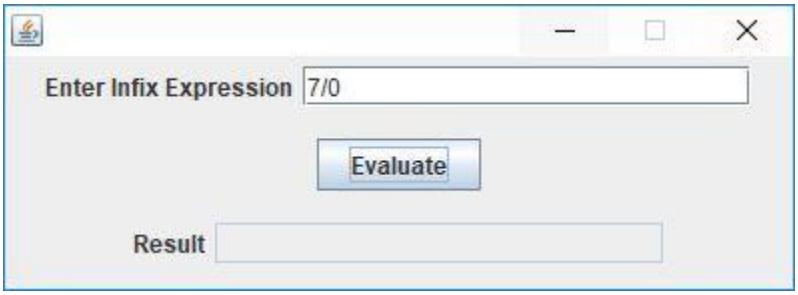
After entering our expression, we simply click the “Evaluate” Button to receive the result.



If we were to have an invalid character in the equation, we will get an error message.

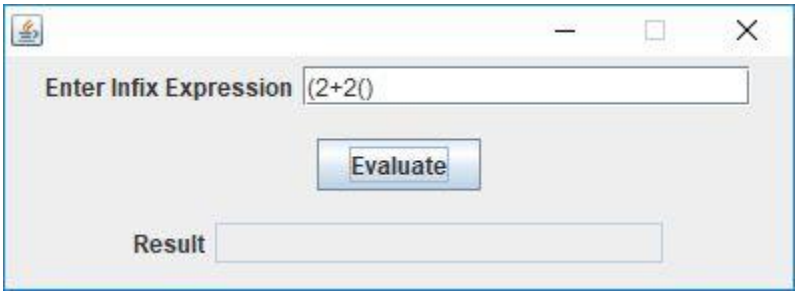


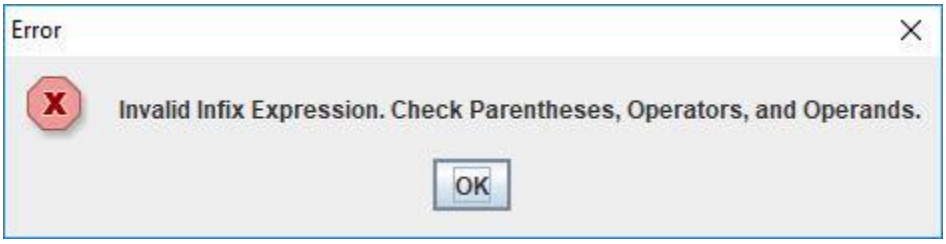
If we were to try and divide by zero, we would get this error message.



If we tried to evaluate an empty input we would get this error message.

Lastly, if we were to enter a logically invalid expression, we would receive this error message.





Lessons Learned

Overall, I learned a lot about how stacks work, store data, and make use of that data! I thoroughly enjoyed this project as it challenged me. When it came to tokenizing the string, and putting the tokens into a stack, I was al little lost at first. For me, this was the most challenging aspect of the project as I didn't have a ton of experience with lists, or stacks, in general. I also learned how to implement patterns when checking the tokens for which type of character it was. This allowed the program to determine a pattern one time, instead of every time it was evaluating which type of character a token is. Please let me know if there is anything I could improve upon!