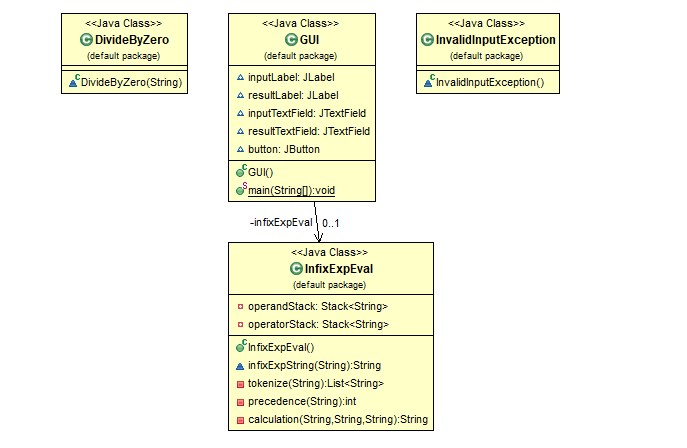
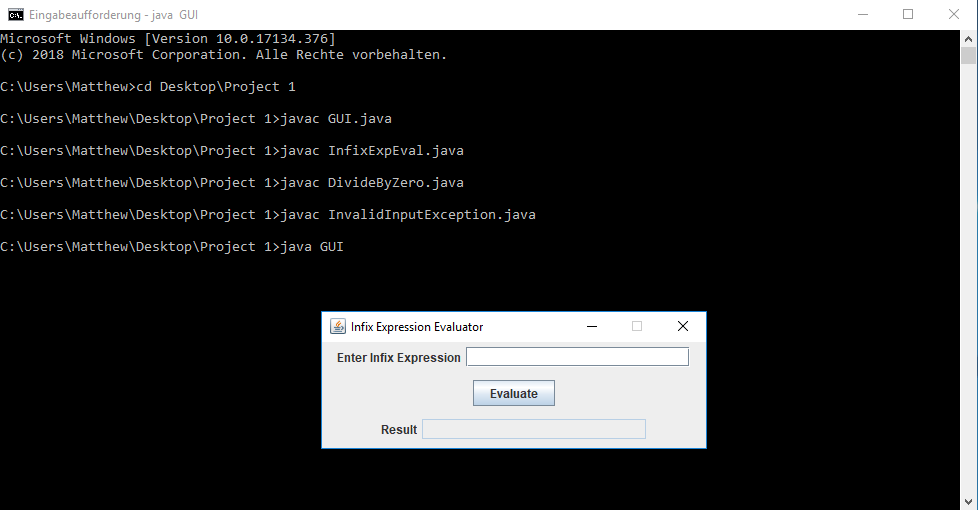
|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Expected Output** | **Actual Output** | **Pass?** |
| **Test Case from Project Prompt:**  **(2 + 3 \* 5) – 8/5 \* (5 – 2)** | **14** | **14** | **Yes** |
| **Test Case for expression w/o spaces:**  **(2+3\*5)-8/5\*(5-2)** | **14** | **14** | **Yes** |
| **Test Case for precedence:**  **4+5\*8-3/2** | **Should be calculated as: 4+(5\*8)-(3/2)**  **4 + (5 \* 8) - (3 / 2)**  **4 + (40) – (1) NOTE: Integer Division**  **4 + 40 – 1**  **43** | **43** | **Yes** |
| **Test Case for parentheses:**  **(((6+3) / 3)+ 8) – (2\*4)** | **(((6+3) / 3)+ 8) – (2\*4)**  **((9 / 3) + 8) – 8)**  **(3 + 8) – 8**  **11 – 8**  **3** | **3** | **Yes** |
| **Test Case for invalid input:**  **8 / h + 3** | **Throws Exception:**  **“Invalid input.**  **The Infix Expression can only contain integer operands (0-9) and the following arithmetic operators:**  **+ - \* / ().”** | **Throws Exception:**  **“Invalid input.**  **The Infix Expression can only contain integer operands (0-9) and the following arithmetic operators:**  **+ - \* / ().”** | **Yes** |
| **Test Case for division by zero:**  **(3 + 4) / 0** | **Throws Exception:**  **“Division by zero.”** | **Throws Exception:**  **“Division by zero.”** | **Yes** |

**UML Class Diagram:**

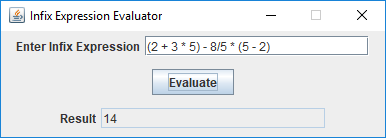


**Test Case Table:**

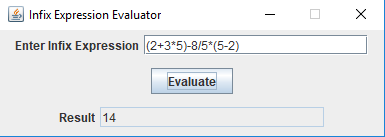
Screen Capture of me successfully compiling and executing my Java program:



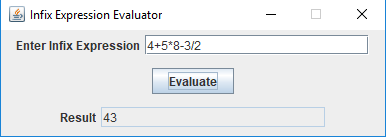
**Screen Captures of Test Case #1:**



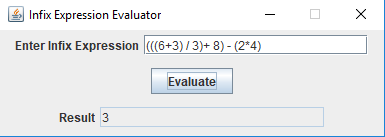
**Screen Captures of Test Case #2:**



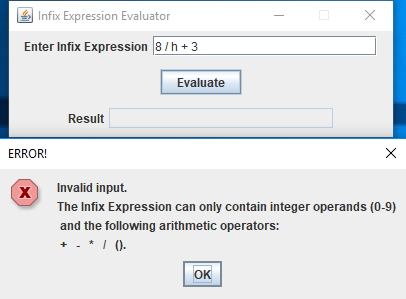
**Screen Captures of Test Case #3:**



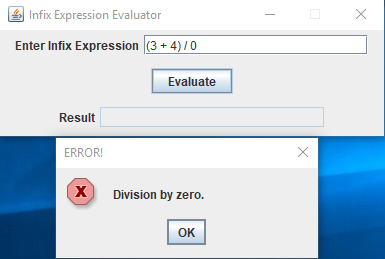
**Screen Captures of Test Case #4:**



**Screen Captures of Test Case #5:**



**Screen Captures of Test Case #6:**



**Lessons Learned:**

This was an admittedly challenging project for me. Fortunately, I was able to learn some new things to help me as a program designer. First, I learned how to use patterns in Java to define a set of characters and then compare a string to it to see if it matches the pattern. This was used to determine if a token was either an operand or an operator. Secondly, this was the first time I had used a “Stack” and all the functions that go with it like: push, pop, peek. I can see a stack being useful for last in, first out data structures. Overall it was a challenging yet fun assignment.