Steven Thaw

sjthaw@me.com

Abstract

A Java program which converts from Prefix Notation to Postfix Notation

CMSC 350 Data structures and analysis

Project 1

Lessons Learned

This project had a lot of lessons learned for me. First and probably most importantly, write pseudo code before you begin to write code. Originally, I had read the project instructions, opened up IntelliJ Idea, and started hacking away at the problem. This resulted in basically a loss of an entire day. I tried numerous different things and just couldn’t get it write. I stopped and took a break, coming back the next day with a fresh head. I wrote what I wanted to do in pseudo code first, and within a few hours I had a working version of the Evaluator class.

The next important lesson I learned was to completely read the directions including the pseudo code included and grading rubric. Somehow I had missed the instructions to not use space as a delimitator, and I was using StringTokenizer with space as a delimitator, which resulted in having to refactor my Evaluator class to not use delimitators.

The final lesson I learned is instead of using class variables, use method variables wherever possible. I spent countless hours trying to figure out why my GUI class was not clearing the results textbox when using the code resultText.setText(“”). Using this line of code, I expected a fresh results box every time a button was pressed, but instead I kept getting combined expressions. I finally realized that the problem wasn’t with my GUI coding (admittedly one of my weak areas), but that I was using a class variable which was not reset to null when the button was pressed, so it was displaying old results along with new results.

All in all, this project, while supremely frustrating at first, left me feeling great at the end and forced me to learn some new ways of doing things. It also further enforced the fact that even though I have been a student for a long time, I need to make sure I read the instructions closely.

UML Class Plan

A screenshot of a cell phone

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Input | Expected Output | Pass/Fail |
| 1 | 5 25 \* (postfix to prefix)  Test all operators | \* 25 5 | Pass |
| 2 | /358 42 (prefix to postfix)  Test all operators/Spaces | 358 42 / | Pass |
| 3 | 54 27 + 15 \* (postfix to prefix)  Test all operators | \* 15 + 27 54 | Pass |
| 4 | +1 2 -1 (prefix to postfix)  Pop empty stack | Invalid Syntax Error | Pass |
| 5 | Blank Input (prefix to postfix)  Pop empty stack | Invalid Syntax Error | Pass |
| 6 | + 5 1 2 5 4  Result in non-empty stack | + 5 1 | Pass |

A screenshot of a cell phone

Description automatically generated

Figure 1: Test Case 1

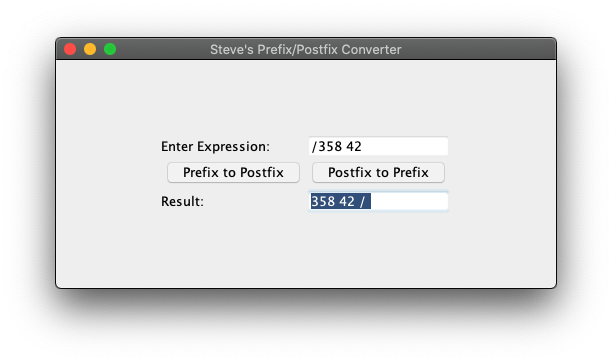


Figure 2: Test Case 2

A screenshot of a cell phone

Description automatically generated

Figure 3: Test Case 3

A screenshot of a cell phone

Description automatically generated

Figure 4: Test Case 4

A screenshot of a cell phone

Description automatically generated

Figure 5: Test Case 5

A screenshot of a cell phone

Description automatically generated

Figure 6: Test Case 6