# Project 1 Documentation:

# Recursive Descent Parsers

Vincent Testagrossa

University of Maryland Global Campus

CMSC 330: Advanced Programming Languages

Dr. Yuhua Jiang

September 11th , 2022

# Contents

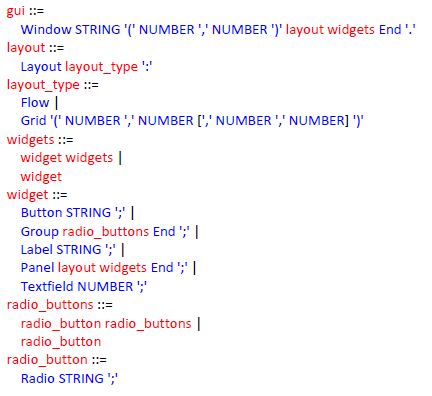
## [Testing](#_Testing_Provided_Calculator)

## [Problem Description](#_Problem_Description_1)

## [Reflection](#_Reflection_1)

# Problem Description

This project uses recursive descent parsing to read the input from a file formatted to the following grammar:

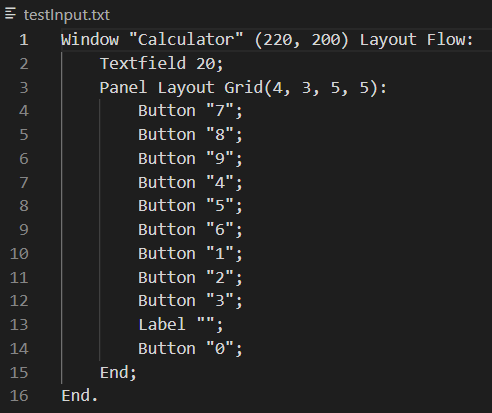


It produces a Java Swing application based on the input file if it’s successfully parsed. If there is an error detected, the parser is immediately halted and an exception is thrown. It properly handles nesting Panels within Panels.

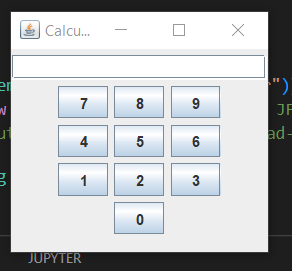
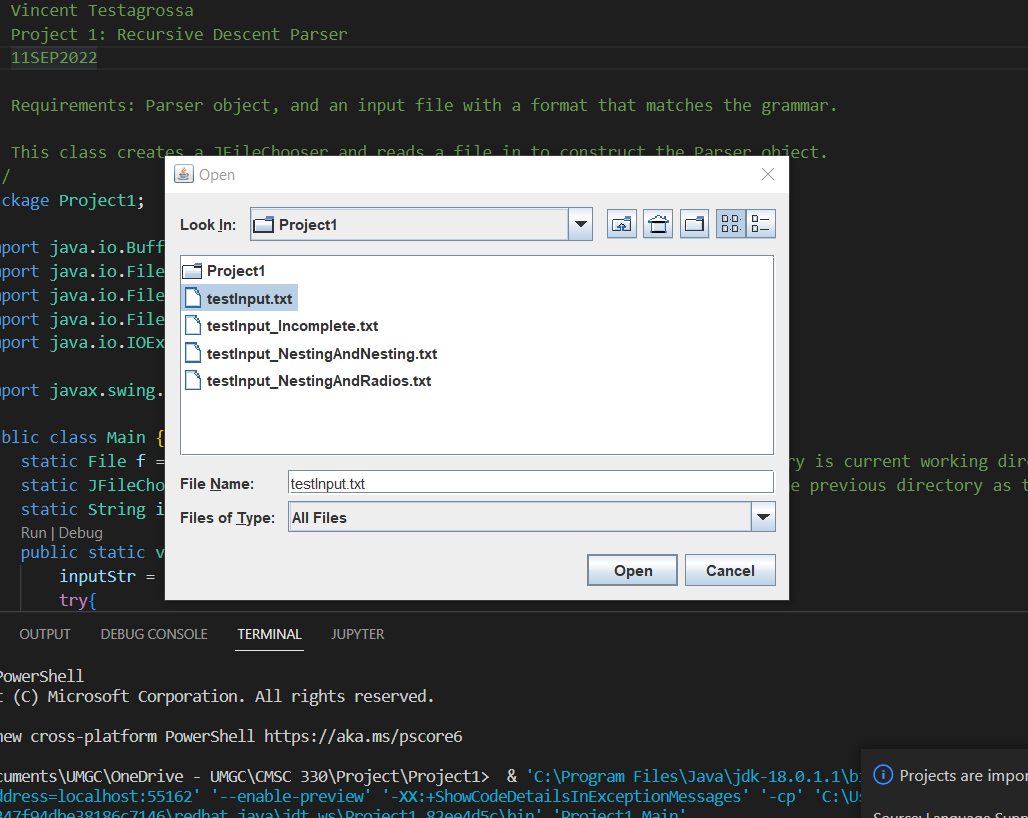
# Testing

## Testing Provided Calculator Sample:

Input:

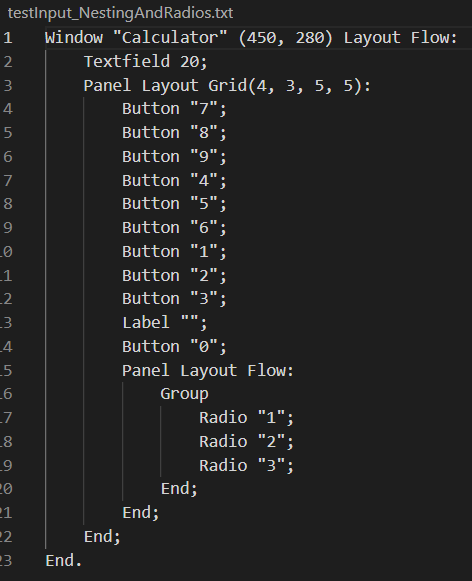


Output:

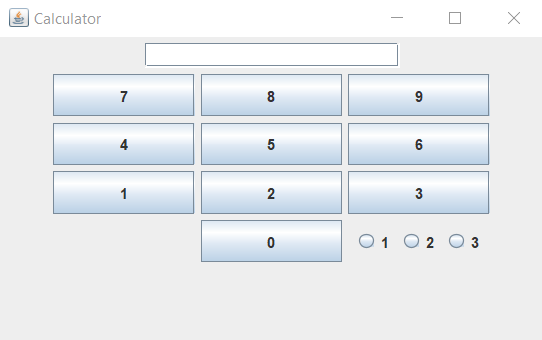


## Testing with Radio Buttons

Input:

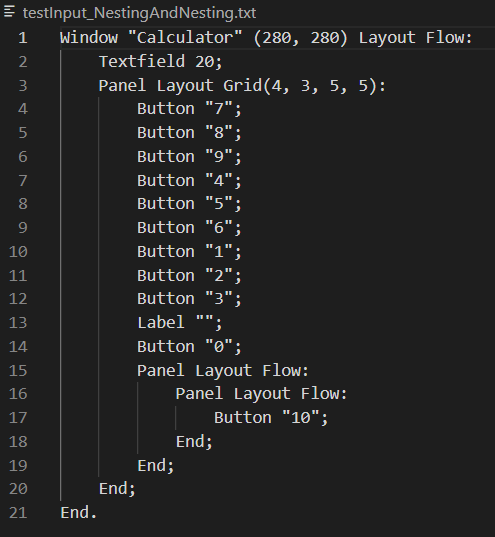


Output:

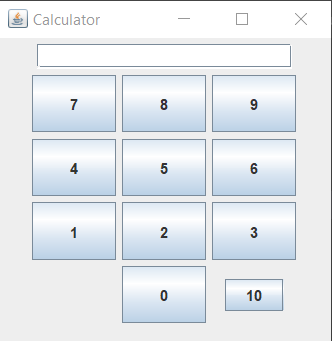


## Testing with Nested Panels

Input:

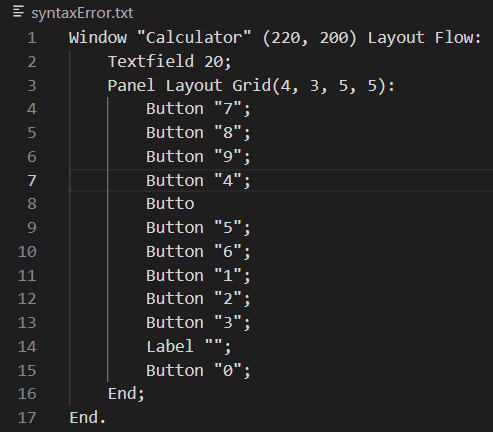
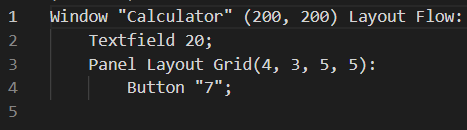


Output:



## Testing Syntax Errors

Input:



Output:



# Reflection

Initially, I wanted to reduce the amount of nested if statements the project had, and I was working hard to put everything into loops containing switch statements. I wound up programming myself into a corner, and the result was not nearly as neat as I had wanted it to be. Recursion was harder to accomplish within loops and there were a lot of “hacky” fixes going on, which is why I decided to rewrite the Parser. Ensuring that the panels properly nested within each other took some tweaking, but I was fortunate that I decided to put a getNext() method in the parser, because it allowed me finer control over when the tokens were consumed. The lexer was relatively straightforward to put together. I had started with an ArrayList as the internal representation so I could use an Iterator to pass the token within getNextToken(), but wanted to have the peek() method that comes with LinkedLists, and wound up changing it to that and then didn’t even use it in the end.

Overall, this was one of the tougher projects I’ve had, but mostly because I was my own worst enemy trying to make my code more maintainable. In the future, I’ll be starting with what seems simpler and refactoring later, rather than trying to do it all in a single pass.