Eric Olsen

Assignment:

The task for this assignment is to write a program that parses a GUI definition language using recursive descent. This parser must properly handle the fact that panels can be nested within other panels and all recursive properties of the language must be implemented using recursion. This program is also implemented in Java and provides alerts for instances of incorrect syntax. Below is a summary of the prescribed GUI definition language in Bacchus-Nor Form. Red font symbolizes non-terminals, blue tokens, and black are BNF meta-symbols.

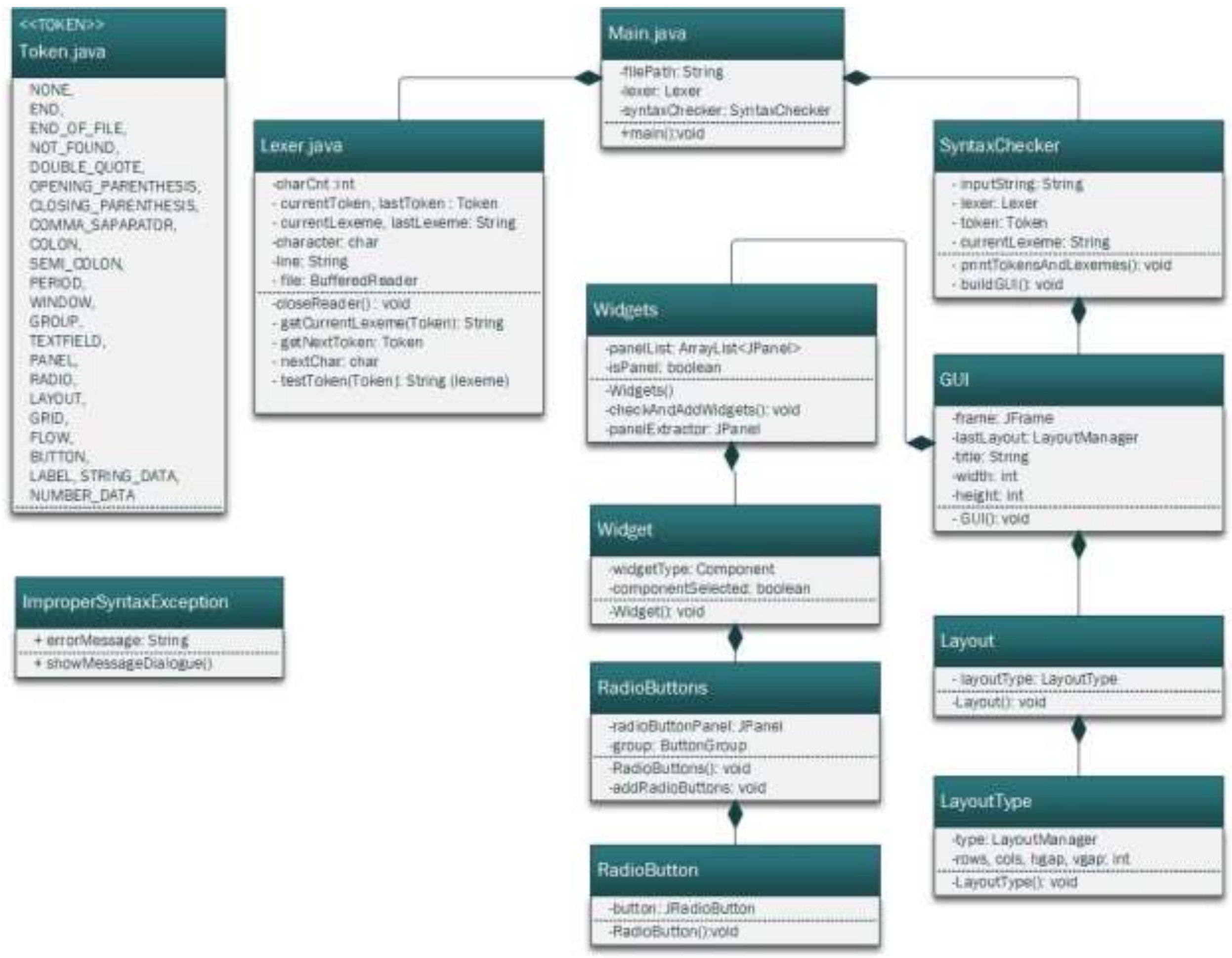
GUI → Window STRING '(' NUMBER ',' NUMBER ')' layout widgets End '.'
layout → Layout layout_type ':'
layout_type → Flow | Grid '(' NUMBER '.' NUMBER [',' NUMBER ',' NUMBER] ')'
widgets → widget widgets | widget
widget → Button STRING ';' | Group radio_Buttons End ; | Label STRING ';' | Panel layout widgets End ';'
| Textfield NUMBER ';'
radio_buttons → radio_button radio_buttons | radio_button
radio_button → Radio STRING ';'

Program Design:

When I approached the design of this program, I decided to use inner classes to define each of the above grammar rules. This would enable a recursive series of references to be built at run time given a variety of different input text files. I also decided to build a Lexical Analyzer very similar to the C-Formatter that we observed in week 2 of this course. This Java class would essentially read the input text file one character at a time and return a valid Token if discovered. This sounds simple but ended up spanning over 200 lines of code.

I then decided to build a Java class "SyntaxChecker.java" to produce the desired GUI. All the inner classes are housed within this class. This essentially gave the inner classes access to the Lexical Analyzer's internal methods like "getNextToken()" so it could build its components and conduct checks for valid syntax. Finally, I also built a custom Exception class which provides data to a user when encountering improper syntax. My intent was to direct the user to the location of the input text where the error occurred. I did not get so far as to specify an actual line number like the Java compiler does, but it does identify sections and lexemes which caused the error.

Below is a UML Diagram of the overall program:



UML Observations:

I treated the instances of the GUI inner-classes as composition. The Token Enumeration is also what I produced to assist the SyntaxChecker class. Due to some nuance in the Java compiler, I re-named STRING and NUMBER to STRING_DATA and NUMBER_DATA respectively.

Code Observations:

I developed a testing method printTokensandLexemes() within the Syntax Checker as a means of validating proper Token output. This method is commented out and not used for the project’s GUI generation, but was extremely helpful in tracing the logic flow of the program’s token output.

There is a somewhat intricate dance between iterative and recursive programming in this project. The GUI inner class, for example, is the root for our grammar structure. It should only be called once, and it adds all components into a JFrame which also should occur only once. The GUI class consequently takes an iterative approach when adding all internal components. After this class instantiates the first widgets class, however, it fires off a long series of recursive object instantiations. When the program flow returns from these instantiations, all the components are created and ready to be added to the frame. The GUI class utilizes an iterative for-each loop to add each nested JPanel. This is the only non-recursive instance of the program.

There is nothing particularly special about the Layout and LayoutType classes other than the LayoutManager they generate. The Layout class has an instance of a LayoutType class and this class specifies either a FlowLayout or GridLayout Manager after checking for proper syntax.

The Widgets class requires some special attention since a possible widget is a JPanel which requires its own layout manager and internal widgets. To house a potentially large number of recursively defined JPanels and other widgets, I decided to include an ArrayList of JPanels and a special Boolean field, isPanel, to assist in this class's recursive calls. I also created an internal recursive helper method to Widgets called "checkAndAddWidgets()." Essentially, if the evaluated token is anything besides a JPanel, it creates a new Widget class and the program flow continues to the Widget constructor. If a JPanel is identified, the isPanel is flipped to true and a Layout and Widgets object are created. Inner class constructors are what continuously guide the program control structure. Once the isPanel field is false and the evaluated Token is "End," this recursive helper method ceases to call itself and program flow returns to the class that preceded it. It now has an ArrayList of JPanels which are ready to be pulled onto the frame.

The RadioButtons and RadioButton operate in a similar fashion to the Widgets class. In this case a ButtonGroup object needs to be at the head of the class so the radio buttons behave as a user would expect. The RadioButtons class has two fields: a JPanel which holds all the RadioButtons and a ButtonGroup which logically links them. The class also uses a recursive helper method which adds new buttons to the JPanel and ButtonGroup until encountering the END token.

As some final touches, the Improper Syntax Exception is thrown whenever the input text does not meet the syntax requirements of the GUI definition language. It has three possible constructors and directs attention to the Token that generated the syntax error, the section of the GUI, and the lexeme which triggered the error. It could be improved, but probably functions well enough to show the user where the error occurred.

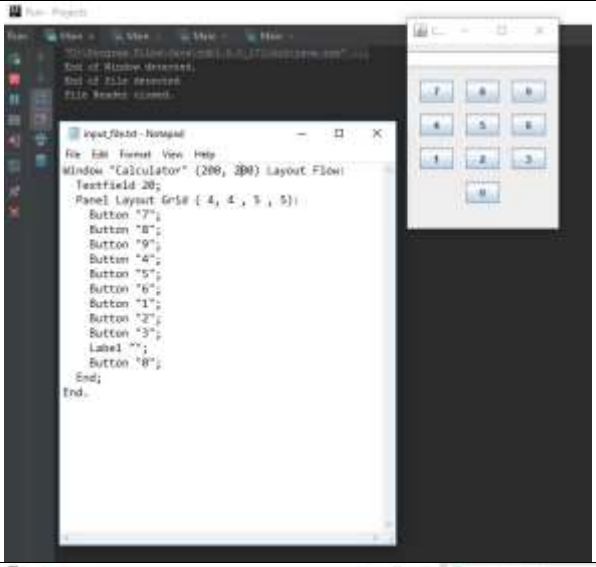
Unresolved Issues:

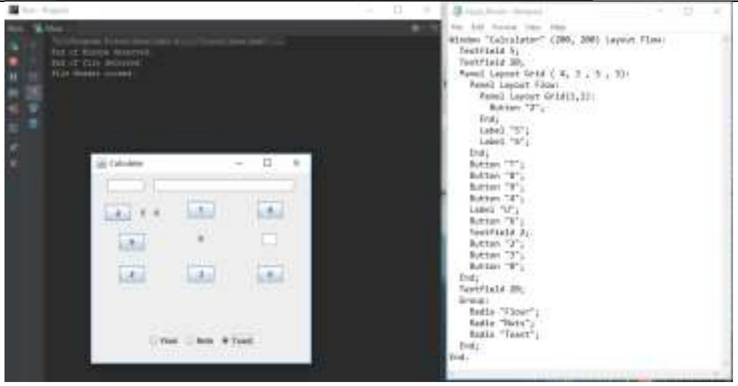
In closing this project, there are some remaining bugs which I ran out of time to correct. The first is that a period '.' will be mistaken as the period Token when observed in the middle of a String. I'm content to just not allow it for now, but future work should probably make some exceptions for allowing a period in String data.

In the main method, I am also unable to run both the buildGUI() method and the printTokenAndLexemes() methods at the same time. This generates a nullPointerException. I elected to just disallow running both, but a better design would be to figure out how to remove this possible issue.

Project Testing

Test Case #	Input	Description	Expected Output	Actual Output	Pass / Fail	Screen shot

1	Window "Calculator" (200, 200)
	Layout Flow:
	 Textfield 20;
	 Panel Layout Grid (4, 4 , 5 , 5):
	  Button "7";
	  Button "8";
	  Button "9";
	  Button "4";
	  Button "5";
	  Button "6";
	  Button "1";
	  Button "2";
	  Button "3";
	  Label "";
	  Button "0";
	 End;
	End.
		Test data given in the project description	Java GUI which matches assignment description	See Image 1	Pass	

2	Window "Calculator" (200, 200) Layout Flow:
	 Textfield 5;
	 Textfield 20;
	 Panel Layout Grid (4, 3 , 5 , 5):
	  Panel Layout Flow:
	   Panel Layout Grid(1,1):
	    Button "2";
	   End;
	   Label "5";
	   Label "6";
	  End;
	  Button "T";
	  Button "8";
	  Button "9";
	  Button "4";
	  Label "U";
	  Button "6";
	  Textfield 2;
	  Button "2";
	  Button "3";
	  Button "0";
	 End;
		Test for 3 nested panels and all widgets.	Java GUI, see image 2	See image 2	Pass	

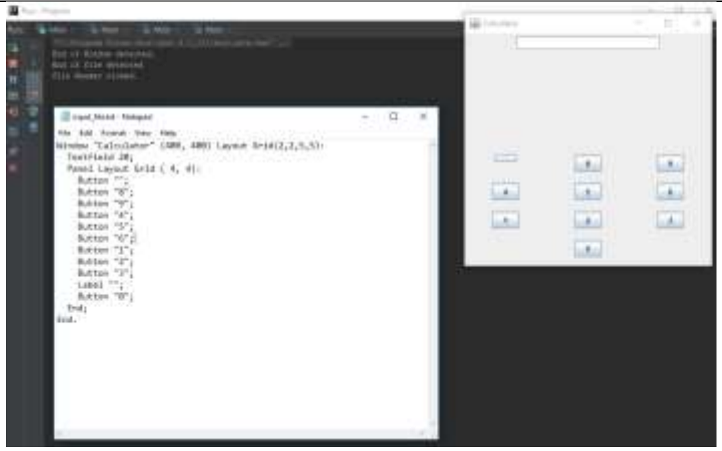
```
[illegible]
End of Window detected.
End of File detected.
File Reader closed.
```

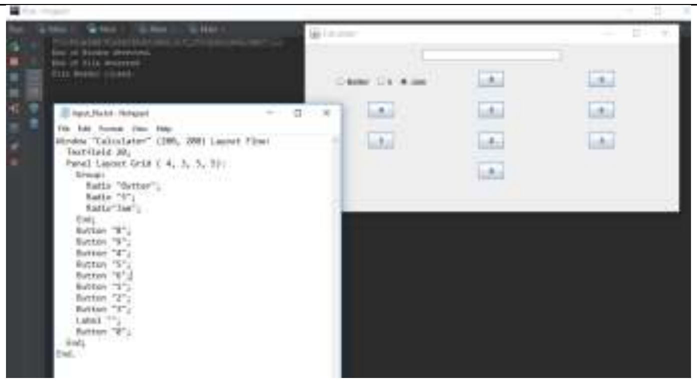
```
Window "Calculator" (200, 200) Layout Flow:
  Textfield 20;
  Panel Layout Grid ( 4, 4 , 5 , 5):
    Button "7";
    Button "8";
    Button "9";
    Button "4";
    Button "5";
    Button "6";
    Button "1";
    Button "2";
    Button "3";
    Label "";
    Button "0";
  End;
End.
```

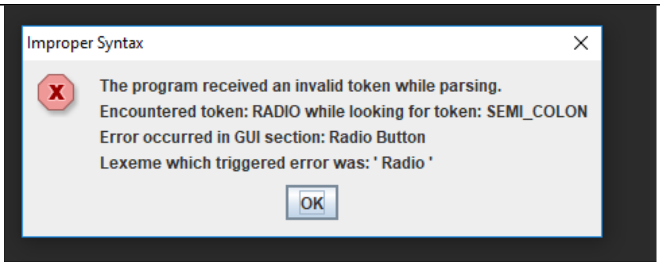
```
[illegible]
End of Window detected.
End of File detected.
File Reader closed.
```

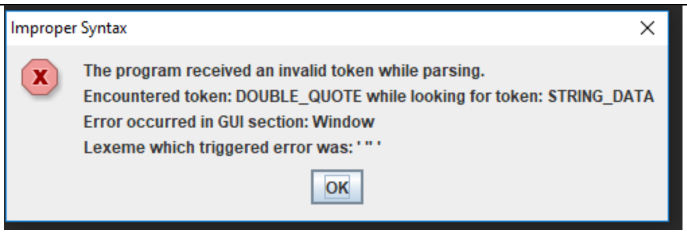
```
Window "Calculator" (200, 200) Layout Flow:
  Textfield 5;
  Textfield 20;
  Panel Layout Grid ( 4, 3 , 5 , 5):
    Panel Layout Flow:
      Panel Layout Grid(1,1):
        Button "2";
      End;
      Label "5";
      Label "6";
    End;
    Button "T";
    Button "8";
    Button "9";
    Button "4";
    Label "U";
    Button "6";
    Textfield 2;
    Button "2";
    Button "3";
    Button "0";
  End;
  Textfield 20;
  Group:
    Radio "Flour";
    Radio "Nuts";
    Radio "Toast";
  End;
End.
```

	Textfield 20;
	Group:
	Radio "Flour";
	Radio "Nuts";
	Radio "Toast";
	End;
	End.					

3	Window "Calculator" (400, 400) Layout Grid(2,2,5,5):
	Textfield 20;
	Panel Layout Grid (4, 4):
	Button "";
	Button "8";
	Button "9";
	Button "4";
	Button "5";
	Button "6";
	Button "1";
	Button "2";
	Button "3";
	Label "";
	Button "0";
	End;
	End.
		Test both grid constructors and ensure that they can nest properly	Java GUI	See image 3	Pass	

4	Window "Calculator" (200, 200) Layout Flow:
	Textfield 20;
	Panel Layout Grid (4, 3, 5, 5):
	Group:
	Radio "Butter";
	Radio "5";
	Radio"Jam";
	End;
	Button "8";
	Button "9";
	Button "4";
	Button "5";
	Button "6";
	Button "1";
	Button "2";
	Button "3";
	Label "";
	Button "0";
	End;
	End.
		Test Radio Button inputs with various spacing	Functional Radio buttons	See Image 4	Pass	

5	Window "Calculator" (200, 200) Layout Flow:
	 Textfield 20;
	 Panel Layout Grid (4, 3, 5, 5):
	  Group:
	   Radio "Butter"
	   Radio "5";
	   Radio"Jam";
	  End;
	  Button "8";
	  Button "9";
	  Button "4";
	  Button "5";
	  Button "6";
	  Button "1";
	  Button "2";
	  Button "3";
	  Label "";
	  Button "0";
	 End;
	End.
		Test Syntax error: missing ; on first widget	Error message, exit code 1	See image 5	Pass	 Improper Syntax
						The program received an invalid token while parsing.
						Encountered token: RADIO while looking for token: SEMI_COLON
						Error occurred in GUI section: Radio Button
						Lexeme which triggered error was: ' Radio '
						OK

6	Window "" (200, 200) Layout Flow:
	 Textfield 20;
	 Panel Layout Grid (4, 3, 5, 5):
	  Group:
	   Radio "Butter";
	   Radio "5";
	   Radio"Jam";
	  End;
	  Button "8";
	  Button "9";
	  Label "";
	  Button "0";
	 End;
	End.
		Empty String on Window name	Error message, exit code 1			 Improper Syntax
						The program received an invalid token while parsing.
						Encountered token: DOUBLE_QUOTE while looking for token: STRING_DATA
						Error occurred in GUI section: Window
						Lexeme which triggered error was: ' " '
						OK

Lessons Learned:

This was an extremely challenging assignment for having taken the last year off on any significant programming exercises. Just to site a few lessons learned, I discovered how useful Boolean fields can be as control measures for avoiding duplicate object instantiation. I also learned that recursively defined language features always need some means of translating back into an iteratively generated compile process. Finally, I don't think I fully appreciated how enumerated data types can be used to validate program input.

Looking at the project now, I think I bloated the code larger than it could be. I think I could have experimented with the ternary operator or incorporated a few more switch statements. The Syntax checker ended up being over 600 lines after implementing all the valid input checks and thrown exceptions. However, I vastly preferred to keep these checks verbose, so I could troubleshoot some of the resulting logic errors.