/\*\*

\* This is main parser class, which will follow the grammar to create a GUI. It will read in the tokens from a gui definition input file the user selects.

\*

\* Name: Nabeel Hussain

\* Class: CMSC 330

\* Professor: Amitava Karmaker

\* Project 1

\* Date: 1/31/2017

\*

\* **@author** Nabeel Hussain

\*/

This project 1 involves writing a program that parses, using recursive descent, a GUI definition language defined in an input file and generates the GUI that it defines.

My program has **four classes:**

My first class is a SyntaxException that will be used if there are any syntax errors in the input file that is read in.

The second class is an enum class that will be used to hold the set of terms used in the grammar for the gui definition language.

The third class is my lexer / scanner class, which will break down the input file into the appropriate tokens to generate the gui.

The fourth class is my recursive descent parser, which will follow the grammar rules, in order to parse the input file, so that the gui can be generated.

**Test Plan:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Input** | **Expected Output** | **Did Test Pass?** |
| 1 | **Input file name:** input1.txt  Window "Calculator" (200, 200) Layout Flow:  Textfield 20;  Panel Layout Grid (4, 3, 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.  \*Will test if the gui gets generated properly, given a **syntactically correct input file**. | A gui frame that is 200 x 200, with a frame title “Calculator”  It contains a Textfield that is 20 wide  It also contains a Grid that has 4 rows and 3 columns, with horizontal and vertical gaps of 5 in between each component. Each grid component will be a button with numbered labels 0 – 9.  **Image of the gui generated will be shown beneath this table.** | Y |
| 2 | **Input file name:** input2.txt  Window "Nested Panels" (600, 300) Layout Flow:  Button " Nested Panels ";  Textfield 20;  Group  Radio "A";  Radio "B";  Radio "C";  Radio "D";  Radio "E";  End;  Panel Layout Grid (4, 3, 5, 5):  Button "7";  Button "8";  Button "9";  Button "4";  Button "5";  Button "6";  Button "1";  Button "2";  Button "3";  Label "";  Button "0";  End;  Panel Layout Flow:  Group  Radio "1";  Radio "2";  Radio "3";  End;  Panel Layout Grid (3, 3):  Button "A";  Button "B";  Button "C";  Button "D";  Button "E";  Button "F";  Button "G";  Button "H";  Button "I";  End;  Panel Layout Grid (3, 3, 10, 10):  Button "One";  Button "Two";  Button "Three";  Button "Four";  Button "Five";  Button "Six";  Button "Seven";  Button "Eight";  Button "Nine";  End;  End;  End.  \*Will test if the parser properly handles **nested panels**, given a syntactically correct input file. | **Image of the gui generated will be shown beneath this table.** | Y |
| 3 | **Input file name:** input3.txt  Window "Radio Button Test" (300, 200) Layout Grid (5, 1):  Group  Radio "Button 1";  Radio "Button 2";  Radio "Button 3";  Radio "Button 4";  Radio "Button 5";  End;  End.  \*Will test if the gui properly generates the **Group** widget type and **radio button** options | **Image of the gui generated will be shown beneath this table.** | Y |
| 4 | **Input file name:** input4.txt  Window "Calculator" (200, 200) Layout Flow:  Textfield 20;  Panel Layout Grid (4, 3, 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.  \*Will test if the program throws a **syntax error** if the input file contains a token in the incorrect place, which does not follow the grammar rules. | **SyntaxException:** “Syntax error in file. File does not syntactically follow the grammar rules” | Y |
| 5 | **Input file name:** input5.txt  Window "Calculator" (200, 200) Layout Flow:  Textfield 20;  Panel Layout Grid (4, 3, 5, 5):  Invalid "7";  Buttonnnn "8";  Button "9";  Button "4";  Button "5";  Button "6";  Button "1";  Button "2";  Button "3";  Label "";  Button "0";  End;  End.  \*Will test if the program throws a **syntax error** if the input file contains a token that’s not a part of the grammar rules. | **SyntaxException:** “Invalid token Butonnnn” | Y |
| 6 | **Input file name:** input6.txt  Window "Textfields" (300, 200) Layout Flow:  Textfield 20;  Textfield 15;  Textfield 10;  Textfield 15;  Textfield 20;  End.  \*Will test if the gui properly generates the **Flow Layout and Textfield** widget type. | **Image of the gui generated will be shown beneath this table.** | Y |

**Screen shots of successful compilation and running for all test cases**

**Test Case 1:** Generating Calculator

Input1.txt:

Window "Calculator" (200, 200) Layout Flow:

Textfield 20;

Panel Layout Grid (4, 3, 5, 5):

Button "7";

Button "8";

Button "9";

Button "4";

Button "5";

Button "6";

Button "1";

Button "2";

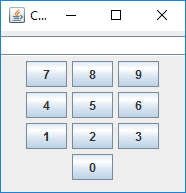
Button "3";

Label "";

Button "0";

End;

End.



**Test Case 2:** Nested Panels Test

Input2.txt

Window "Nested Panels" (600, 300) Layout Flow:

Button " Nested Panels ";

Textfield 20;

Group

Radio "A";

Radio "B";

Radio "C";

Radio "D";

Radio "E";

End;

Panel Layout Grid (4, 3, 5, 5):

Button "7";

Button "8";

Button "9";

Button "4";

Button "5";

Button "6";

Button "1";

Button "2";

Button "3";

Label "";

Button "0";

End;

Panel Layout Flow:

Group

Radio "1";

Radio "2";

Radio "3";

End;

Panel Layout Grid (3, 3):

Button "A";

Button "B";

Button "C";

Button "D";

Button "E";

Button "F";

Button "G";

Button "H";

Button "I";

End;

Panel Layout Grid (3, 3, 10, 10):

Button "One";

Button "Two";

Button "Three";

Button "Four";

Button "Five";

Button "Six";

Button "Seven";

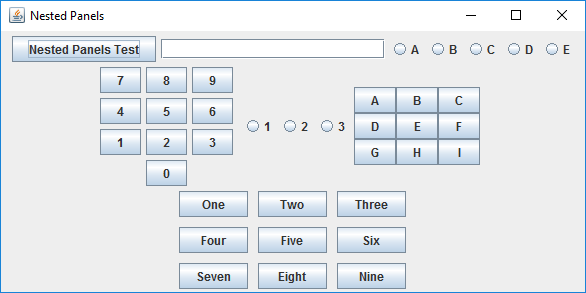
Button "Eight";

Button "Nine";

End;

End;

End.



**Test Case 3:** Radio Button Test

Input3.txt

Window "Radio Button Test" (300, 200) Layout Grid (5, 1):

Group

Radio "Button 1";

Radio "Button 2";

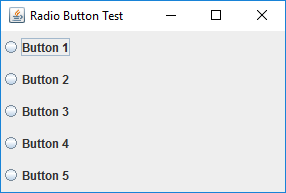
Radio "Button 3";

Radio "Button 4";

Radio "Button 5";

End;

End.



**Test Case 4:** Syntax Error Test

Input4.txt

Window "Calculator" (200, 200) Layout Flow:

Textfield 20;

Panel Layout Grid (4, 3, 5, 5):

Button "7";

Button "8";

Button "9";

Button "4":

Button "5";

Button "6";

Button "1";

Button "2";

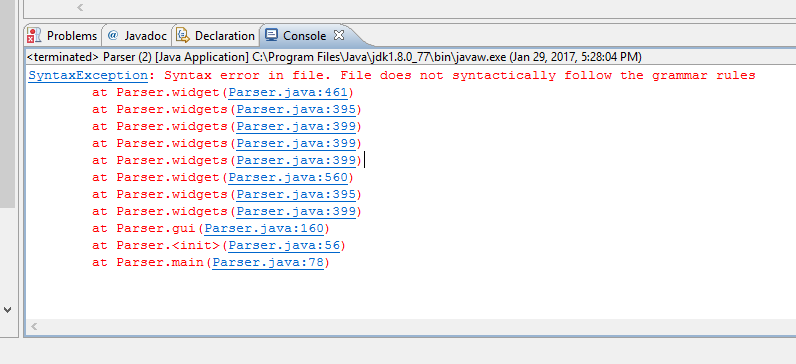
Button "3";

Label "";

Button "0";

End;

End.



**Test Case 5:** Syntax Error Test

Input5.txt

Window "Calculator" (200, 200) Layout Flow:

Textfield 20;

Panel Layout Grid (4, 3, 5, 5):

Invalid "7";

Buttonnnn "8";

Button "9";

Button "4";

Button "5";

Button "6";

Button "1";

Button "2";

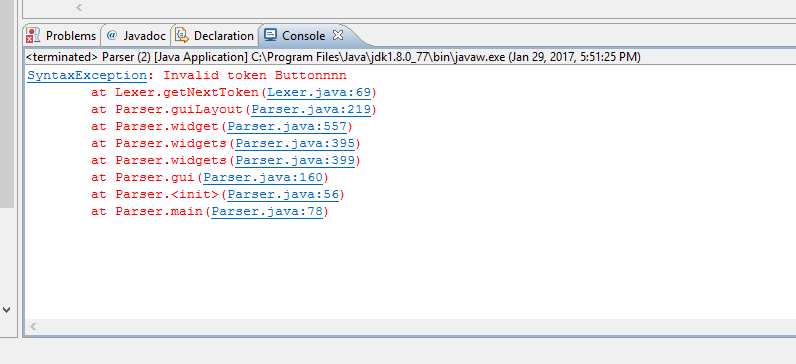
Button "3";

Label "";

Button "0";

End;

End.



**Test Case 6:** Textfields Test

Input6.txt

Window "Textfields" (300, 200) Layout Flow:

Textfield 20;

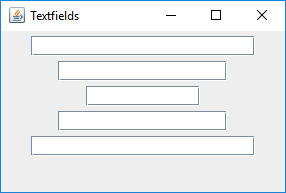
Textfield 15;

Textfield 10;

Textfield 15;

Textfield 20;

End.



**Lessons Learned:**

While working on Project 1, there were a lot of new concepts and techniques that I learned about programming recursive descent parsers and how to implement them using a set of specified grammar. Most of the material from these first couple of modules were brand new information for me, so it took a while before I was able to understand the entire scope of this project. However, once I was able to fully grasp what needed to be done, I created a lexer class which would read in an input file and break it down into tokens. I then used these tokens to follow the gui grammar, in order to generate the actual gui. To break the file down into tokens, I used a StreamTokenizer, which simplified the process for me in terms of separating the file into string, number, and punctuation tokens. It also helped me to distinguish string tokens that were inside quotation marks using the quoteChar() method.

I also used an enumerator class to store the key words from the GUI grammar definition language. This way, when you reach a number, word, or string token, then it should return enumerated type NUMBER, WORD, or STRING back to the lexer, and in turn to your main class. Then the main class will decide what to do with that specific token type by following the grammar rules to create the gui.

In the recursive descent parser class, I used the examples from the week 2 readings to guide me in creating the parsing methods for the productions. Each method returns a Boolean value: true if the production statement parses, and false otherwise. I found myself having to use a lot of nested if-else statements and switch statements as well to properly incorporate all the different types of widget and layout options that can be chosen to build the gui.

Overall, this assignment was challenging, but it definitely helped me understand and reinforce all the material I learned from the weekly modules much better.