Brian Quilty – 18373856 **Assignment 1: A Lexical and Syntax** **Analyzer**

Name(s): Programme: CASE4   
Module Code:

Assignment Title: CA4003

Submission Date: 27/10/22

Module Coordinator: Dr. David Sinclair

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I/We have read and understood the Assignment Regulations. I/We have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged, and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

I/We have read and understood the referencing guidelines found at

<http://www.dcu.ie/info/regulations/plagiarism.shtml> , <https://www4.dcu.ie/students/az/plagiarism>

and/or recommended in the assignment guidelines.

Name(s): \_\_\_BrianQuilty\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_27/10/22\_\_\_\_\_\_\_\_\_

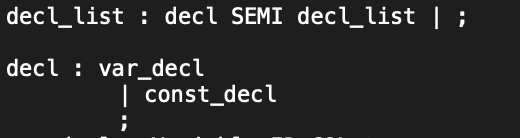
**Cal.G4**

Lines 1 through to 77 define the grammar rules of the grammar file.

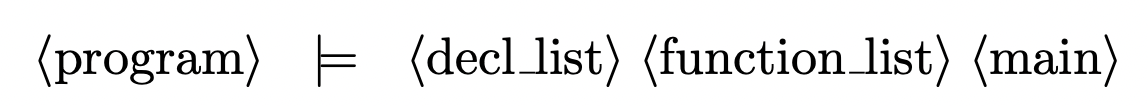
The first line defines what the grammar file is to be called. In this case it is cal.

The following 76 lines define the main body of the grammar rules that were given in the document. The grammar rule name is followed by ‘:’. The syntax for the rules proceeds the naming of the rules and once the rules have been defined, they are all followed by a ‘;’ which signifies the creation of a grammar rule.

Once a rule is created, inside that rule is a new rule that has not yet been defined. The following rule therefore should be the rule that has not yet been defined giving a natural order to the grammar.



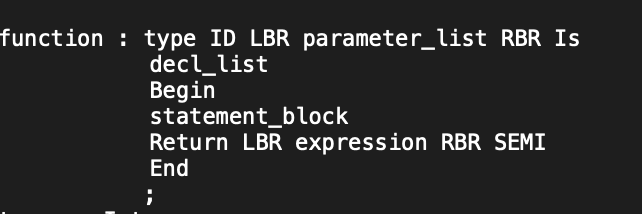
For example, decl\_list is defined here. Inside this rule decl is referenced which has not yet been defined. The following rule therefore will define the grammar rule decl.





This is an example of taking the grammar that is defined inside the document and simply applying it to the grammar file. The |= is replaced with a “:” and for the rules ‘decl\_list’ ‘function\_list’ ‘main’ the brackets are removed simply defining the rules by their names.

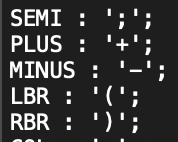
Inside the grammar rules there are reserved words and operators which will be defined later inside the grammar file.



For example, the function grammar rule, line 5 ‘Return LBR expression RBR SEMI’ uses both reserved words and operators to help define a rule. Return is defined as a reserved word as follows;



This is made up of the fragments r, e, t, u, r, n where any of these fragments can either be uppercase or lowercase. ￼



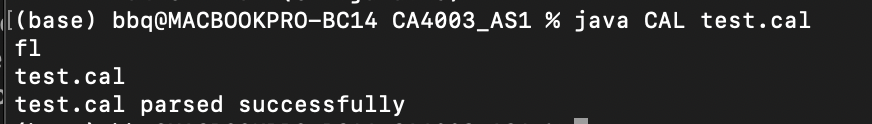
Here we can see that the semi, LBR and RBR operators have been defined. The name for the operator is followed by “:” then the operator that was defined in the document for example “)” is quoted and then followed by a semicolon signifying that it has been defined.



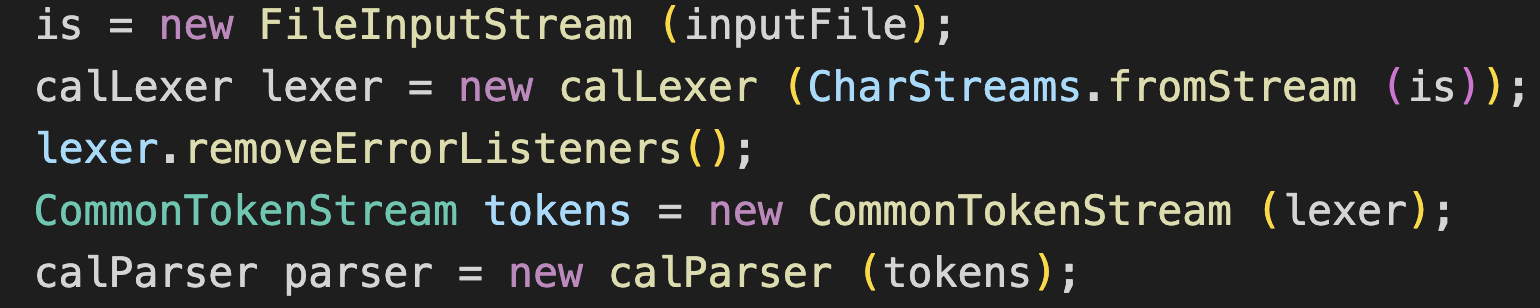
We must allow for whitespace characters. Thus, if there is a space, tab new-line or return one or more times we call skip.

**Cal.java**

The main function of cal.java takes in an input file from the argument passed in through the terminal/ command line.



Next a Callexer is created from the charstreams of our inputfile which is now a fileinputstream. A commontokenstream is created from the callexer that was just defined.



We remove the errorlisteners from the lexer as we will define a new errorlistener for our parser, we will also apply the same methodology to our parser, removing the errorlisteners aswell.

An errorlistener will throw a parsecancellationexception if it meets an error during the parsing process. A try and catch exception will run the parser from the program node. If it catches an exception from the error listener that was just created, it will print a new line from the commandline/terminal stating that the input file was not parsed properly. Otherwise, it will print a new line from the commandline/terminal stating the input file was parsed successfully.