Assignment 1

Deja Tyla Jackson & Ezekiel Lawal

[Djack133@students.kennesaw.edu](mailto:Djack133@students.kennesaw.edu); elawal@students.kennesaw.edu?

9/11/2018

Kennesaw State University, College of Computing & Software Engineering, Department of Computer Science

CS4308 Concepts of Programming Language W01

1. **Problem Statement**

This project involves creating a java interpreter that parses some minimal form of Julia statements. These Julia statements will only include the integer type, contain whitespace to separate tokens and will have only single letter identifiers. The java program should have the capability to error detect, build basic data structures to be executed and execute the Julia code.

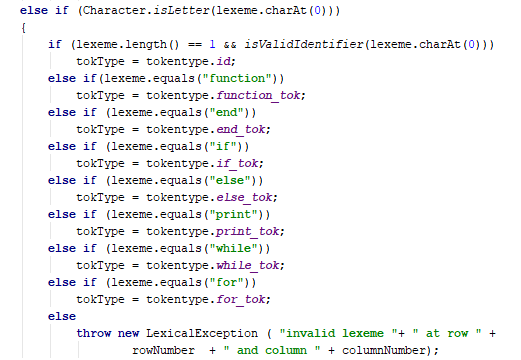
1. **Summary and Purpose**

The purpose of this report is to summarize the solution we have to the problem as mentioned. This report includes information on the solution to the problem, our results, limitations and improvements that could be made to help make this solution better. In addition, this report contains a conclusion section in which we have drawn logical conclusions from this project.

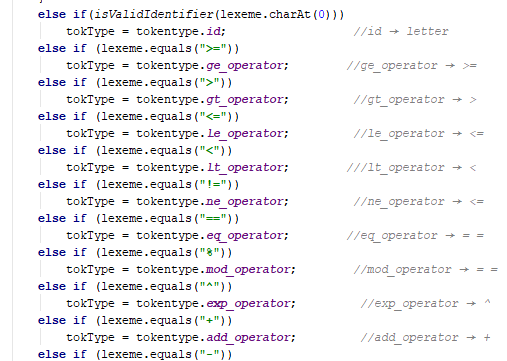
1. **Description**

The overall solution includes several classes that define the different functionalities within the Julia program. A huge emphasis is placed on being able to identify different arithmetic expressions and statements for a successful implementation of the language. Below are some two main classes that were integral to the development of the Julia interpreter.

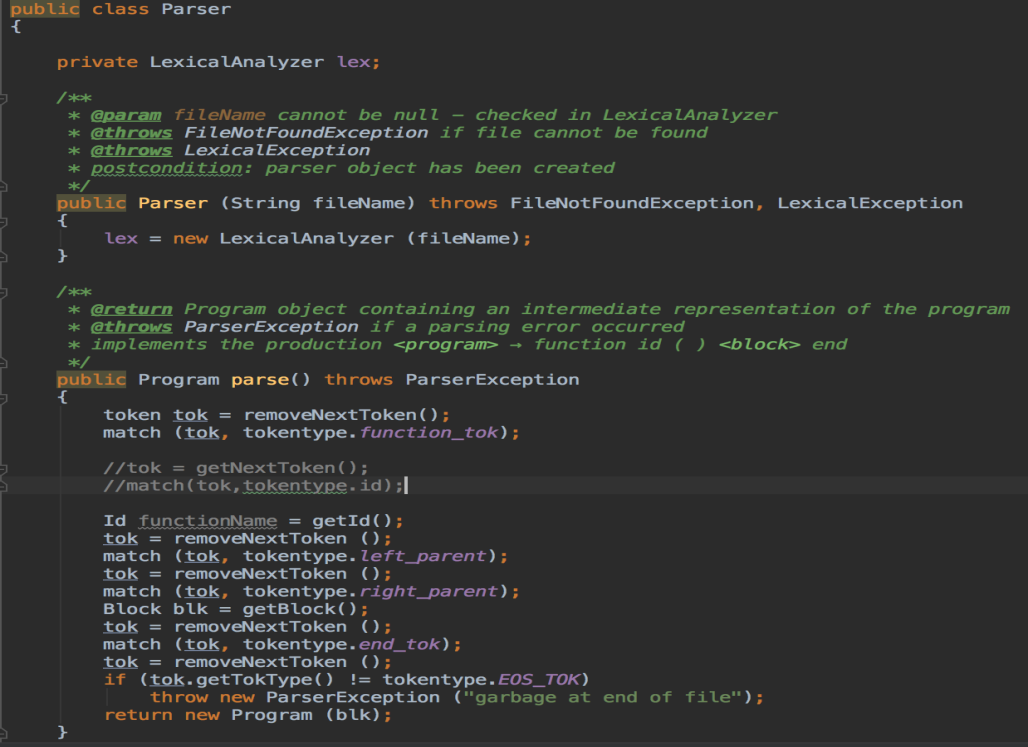
**Lexical Analyzer**

****

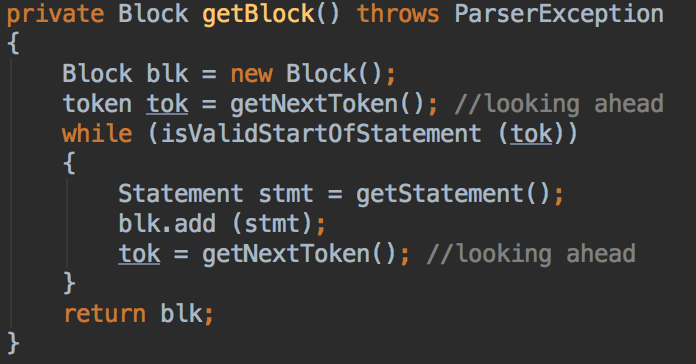
This portion of the lexical analyzer checks for reserved words.

****

This portion of the lexical analyzer checks for various operators within our grammar.

****

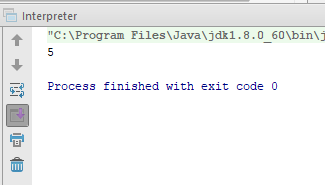
Parse() initializes the identifiers within a program.



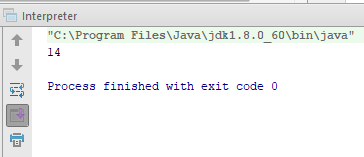
getBlock() will parse a block for valid statements and return the <Block>. There are several other methods that will do similar things in terms of getting and parsing statements, print statements, while statements, arithmetic expressions, boolean expressions, etc.

1. **Results**

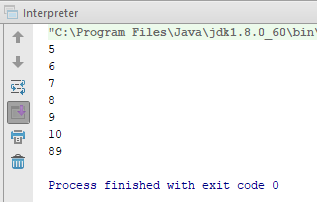
**The following are the results from Test1:**



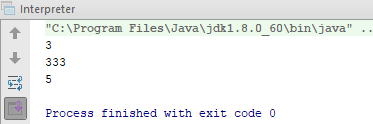
**The following are the results from Test2:**



**The following are the results from Test3:**



**The following are the results from Test4:**



1. **Improvements**

Some improvements could have been made in the architecture of the overall design. There are some pieces of code (methods) that aren’t necessary to run the tests successfully. We could definitely clean the classes up a bit by adopting a uniform Capitalization and nomenclature structure. It can often get confusing which is a class and which is a method.

1. **Limitations**

The limitation of the interpreter is that it only accepted one data, which is an integer. Resultantly, data types to include decimals or floating numbers will not be understood by the interpreter. In addition, the interpreter only accepts identifiers that are single letters such as “a”. As a result of this limitation, identifiers such as “sum” or “total” would not be understood and may result in errors from the program.

1. **Conclusion**

With a great divide and conquer effort, we were able to successfully complete the project without feeling completely overwhelmed as an individual. Using GitHub, we were able to still operate collaboratively sharing changes in real-time and meeting consistently virtually.

Our code is efficiently able to parse the 4 test files from the minimal Julia language. When errors are encountered it reports an error message and gives the indicated column and row number to assist with troubleshooting.

During the implementation phase, we encountered some difficulties in programming the for statement class and using the iterator class with it. In future implementations of this project, we will spend more time in the design phases prior to beginning the code to ensure our ideas and solutions will work as intended.