Principles Of Programming Languages

Project 1 Part 2

Melik Özdemir 150120004

Ömer Deligöz 150120035

Ahmet Abdullah Gültekin 150121025

In this project, we are expected to implement a syntax analyzer for the given specific programming languages. In the second part of the project, we used the tokens produced by the scanner that we implemented in the first part.

**Code and Description:**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

We have a class that is called Parser. In this class, first of all, we have some ArrayLists that hold some special values that are useful for us. We have two ArrayList that are hold String data type. One is for tokens, and one is for output. The other ArrayList is holding the Integer data type for Indexes. Moreover, we have an Integer variable that is currentTokenIndex. It is used for following the index of current token.

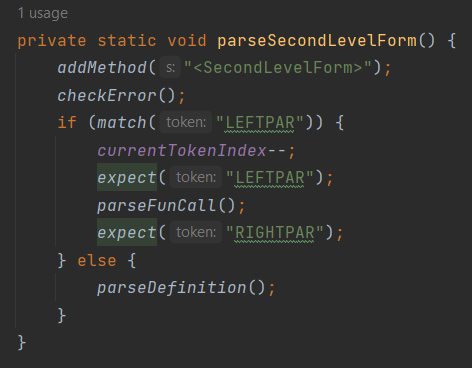
metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

We have parseProgram method. In this method, we add the String “<Program>” to the output. We can do this by addMethod that we used later. Moreover, we called the checkError method for checking the token. Actually, in checkerror method, we check the parentheses that are not normal parentheses. If we face LEFTSQUAREB or RIGHTSQUAREB or LEFTCURLYB or RIGHTCURLYB, the program gives the error message. The error message includes Syntax error and ındex of current token. Additionally, we have an if else condition. In this if condition, we check the LEFTPAR. If the index of current token equals to the LEFTPAR, we call the match method that checks the current token is leftpar. If the current index is LEFTPAR, the index of current token is increased by one. And then, the index of current of token is decreased by 1 for keeping index of current steady. Then, the parseTopLevelForm method is called. In parseTopLevelForm, we add the string “<TopLevelForm>” to the output. And last of all, we called the parseProgram merthod recursively.



In parseTopLevelForm, we add the string “<TopLevelForm>” to the output. Then, we check the parentheses that are not normal parentheses. If we face LEFTSQUAREB or RIGHTSQUAREB or LEFTCURLYB or RIGHTCURLYB, the program gives the error message. The error message includes Syntax error and index of current token. Additionally, we call the expect method for checking that the current token equals to the LEFTPAR or not. If it is LEFTPAR, the program will add the token to the output by addToken method. If it is not LEFTPAR, the program will give the error message (Syntax Error). Then, we call the parseSecondLevelForm method for add the string “<SecondLevelForm>”. Finally, we call expect method with RIGHTPAR string. And it checks the current token is RIGHTPAR or not.



In this method, the string “<SecondLevelForm>” is added to output. Then, we check the parentheses that are not normal parentheses. If we face LEFTSQUAREB or RIGHTSQUAREB or LEFTCURLYB or RIGHTCURLYB, the program gives the error message. Additionally, we have an if else condition. In this if condition, we check the LEFTPAR. If the index of current token equals to the LEFTPAR, we call the match method that checks the current token is leftpar. If the current index is LEFTPAR, the index of current token is increased by one. And then, the index of current of token is decreased by 1 for keeping index of current steady. Additionally, we call the expect method for checking that the current token equals to the LEFTPAR or not. If it is LEFTPAR, the program will add the token to the output by addToken method. If it is not LEFTPAR, the program will give the error message (Syntax Error). Then, we call the parseFunCall method for adding the string “<FunCall>” to the output. Then, the method calls the expect method with RIGHTPAR string. And it checks the current token is RIGHTPAR or not.

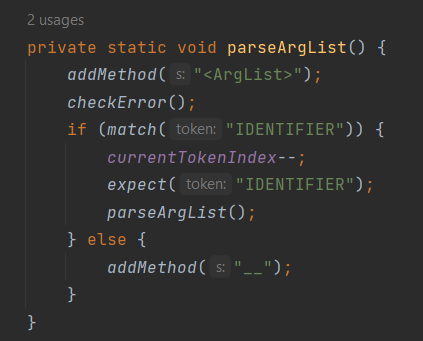
metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

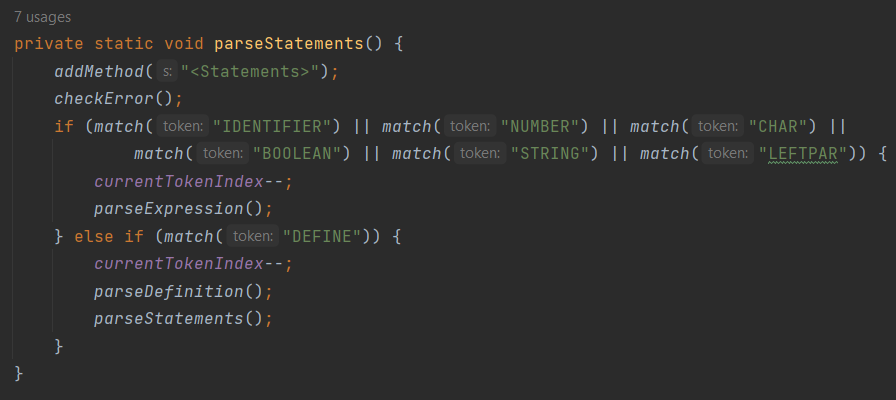
In this method, first of all, we call the addMethod for adding the string “<Definition>” to the output. Then, we call the check error method. Unlike the above, we call the expect method with “Define”. Last of all, we call the parseDefinitionRight method for adding the string “<DefinitionRight>” to the output.



After adding the string to the output, we again call the checkError method. Moreover, we have if-else condition. In this if condition, we check the LEFTPAR. If the index of current token equals to the LEFTPAR, we call the match method that checks the current token is leftpar. If the current index is LEFTPAR, the index of current token is increased by one. And then, the index of current of token is decreased by 1 for keeping index of current steady. Then, we call the expect method with string LEFTPAR and IDENTIFIER. Additionally, we call the parseArgList for adding the string “<ArgList>” to the output. Then, we call the expect method with RIGHTPAR. Last of all, we call the parseStatements method for adding the string “<Statements>” to the output.



After adding the string to the output, we again call the checkError method. Moreover, we have if-else condition. In this if condition, we check the LEFTPAR. If the index of current token equals to the LEFTPAR, we call the match method that checks the current token is leftpar. If the current index is LEFTPAR, the index of current token is increased by one. And then, the index of current of token is decreased by 1 for keeping index of current steady. Additionally, we have an if else condition. In this if condition, we check the IDENTIFIER. If the index of current token equals to the IDENTIFIER, we call the match method that checks the current token is leftpar. If the current index is IDENTIFIER, the index of current token is increased by one. And then, the index of current of token is decreased by 1 for keeping index of current steady. Then, we call the expect method with IDENTIFIER. Finally, the method is called recursively.



After adding the string to the output, we again call the check error method. Then in if condition we call the match method with some parameters such as Identifier, number, char, Boolean, string, leftpar. In match method, we return true or false, according to the answer, if condition or else condition is executed. In if condition, parseExpression method is called. In else if condition, parseDefinition and parseStatements methods are called.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

After adding the string to the output, we again call the check error method. Then in if condition we call the match method with some parameters such as Identifier, number, char, Boolean, string, leftpar. In match method, we return true or false, according to the answer, if condition or else condition is executed. In if condition, parseExpression and parseExpressions methods are called.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

After adding the string to the output, we again call the check error method. Then in if condition we call the match method with some parameters such as Identifier, number, char, Boolean, string, leftpar. In match method, we return true or false, according to the answer, if condition or else condition is executed. In else if condition, expect method is called with LEFTPAR and RIGHTPAR. Additionally, parseExpr method is called.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

In this method, the string “<expr>” is added to the output. After adding string, the checkError method is called. Unlike the above, in this if condition, first of all, we check that is current token is LET or not. If it is LET the parseLetExpression method is called. If the current token is COND, the parseCondExpression method is called. If the current token is IF, the parseIfExpression is called. If the current token is BEGIN, the parseBeginExpression method is called. If the current token doesn’t equal to the above, the parseFunCall method is called.

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

In this method, the string is added to the output and checkError method is called. Then, expect method is called with IDENTIFIER. Finally, parseExpressions method is called.



In this method, the string is added to the output and checkError method is called. Then, expect method is called with LET. Finally, parseLetExpr method is called.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

In this method, the string “<LetExpr>” is added to the output. Then, we call the check error method. Moreover, we have if else condition. If the current token equals the LEFTPAR, the expect method will be called with LEFTPAR and RIGHTPAR. IF the current token equals the IDENTIFIER, the expect method will be called with IDENTIFIER, LEFTPAR, RIGHTPAR. In both cases, the parseVarDefs and parseStatements methods will be called.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

In first method, the string “<VarDefs>” added to the output. Then the checkError method is called. Expect method is called with LEFTPAR, IDENTIFIER and RIGHTPAR. And some methods are called.

In second method, the string “<VarDef>” added to the output. And checkError method is called. Then, in the if condition, if the current token, equals to the LEFTPAR, parseVarDefs method is called.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

In this method, the string “<CondExpression>” is added to the output. Then checkError method is called. Moreover, the expect method is called with COND. Last of all, the parse CondBranches method is called.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

In the first method, the string “<CondBranches>” added to the output. As always, the checkError method is called. Expect method is called with LEFTPAR and RIGHTPAR. And some methods are called like above.

In the second method, the string “<CondBranch>” is added to the output. Then checkError method is called. In if condition, if the current token equals to the LEFTPAR, the expect method is called with LEFTPAR and RIGHTPAR. Additionally, the parseExpression and parseStatements methods are called.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

In this method the string “<IfExpression>” is added to the output. Then expect method is called with the IF. Last of all some methods are called like above.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

In this method, the string “<EndExpression>” is added to the output. Then in if condition we call the match method with some parameters such as Identifier, number, char, Boolean, string, leftpar. In match method, we return true or false, according to the answer, if condition or else condition is executed.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

In this method, the string “<BeginExpression>” is added to the output. Then checkError method is called and expect method is called with BEGIN.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

In this method, as we mentioned above, we check the current token, if it equals to the parameter token, the index of token increased by one and method returns true. Else, the method returns false.

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

If token doesn’t match with the current token, the program will give the error message with Syntax Error.

metin, ekran görüntüsü, multimedya yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

In this method, as we mentioned above, If the token equals to the LEFTSQUAREB or RIGHTSQUAREB or LEFTCURLYB, or RIGHTCURLYB, the program will give the error message with the SYNTAX ERROR.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

In this method, If the input lexically incorrect, the parser outputs the scanner’s error message as in first part of the project.

metin, ekran görüntüsü, yazı tipi içeren bir resim

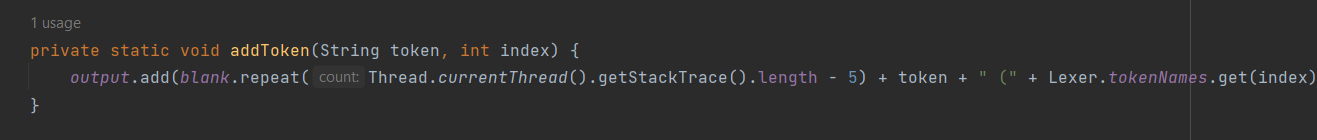
Açıklama otomatik olarak oluşturuldu

In this method, we create a new output.txt file for adding the outputs to the output file.

A screen shot of a computer

Description automatically generated with low confidence

In this method, we add parameter ”s” to the output file and concatenate them.



In this method, we add parameter token with index of it.

metin, elektronik donanım, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

In this method, we have an empty string that is called str. Then we have a try catch expression. The contents of the file are read with the Files.readAllBytes method. Then it is assigned to the string str. If the file cannot be read, the message of the exception object is written to the console. The variable str is divided to the strArr with “\r\n”. If the last element of strArr is started with the LEXICAL ERROR, the program gives the error and ends. The method printLexicalError is called and error message is written. The program ends with the System.exit(-1). An ArrayList that is called strLists is created. The Array list has parts of the lines. It creates another loop for each item of the strList list. The first part of each element (s[0]), token accessories is added to an ArrayList.

The second part (s[1]) is divided by the ":" character and assigned to an array with parts. This array contains row and column indexes. The values in the parts array are converted to integers with the Integer.parseInt method and assigned to the row and col variables. An ArrayList named indexes is created and an Integer array containing row and col values is added to this list.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

In main method, an input file is wanted from the user. And the program processes that file. In try catch expression, we used the lexerOutputFile and we called the parse method for starting the process. After the parse method is finished, the results are printed to the console.

**Conclusion:**

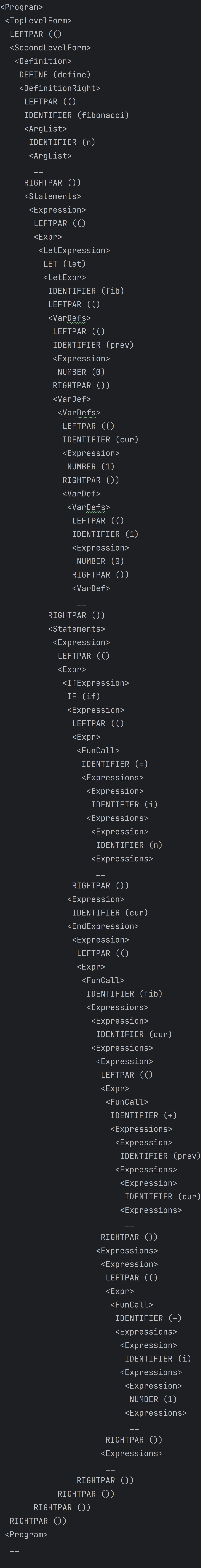
We have completed all the parts of the assignment, and it works as desired. The program produces the desired output, and we are confident that it meets the requirements of the assignment. I have also included a detailed explanation of how the program works.

**Input-Outputs**

**Correct input:**

A picture containing text, font, screenshot, black

Description automatically generated**Output:**

**A picture containing text, screenshot

Description automatically generated**

**Incorrect input:**

**A picture containing text, font, screenshot, black

Description automatically generated**

**Output:**

**A picture containing screenshot, text, black

Description automatically generatedA picture containing screenshot, text, black

Description automatically generated**