Ben Pallotti

Concepts of Programming Languages-4308-W02­ Summer 2023

**P1-Scanner**

7/7/23  
Professor Sharon Perry  
100% Complete

1. **Introduction**

This project is to develop a scanner for a minimal form of the Julia language. The scanner is responsible for tokenizing the input Julia file, extracting individual tokens, and providing them to the parser for further processing. The scanner is implemented in Java.

1. **Scanner Implementation**

The scanner implementation consists of the JuliaScanner class, which uses the java.util.Scanner class to read the Julia input file and extract tokens. The getNextToken() method retrieves the next token from the input file using the next() method of Scanner. If no more tokens are available, it returns null to indicate the end of the file.

1. **Execution and Output**

The program reads tokens from a file and retrieves a token, then the next token, get the corresponding lexeme symbol, and then its . The program’s main method demonstrates the scanner by printing the input file contents and the lexemes with their symbols. The lexeme-symbol mappings are defined in a hashmap. Here is the execution output:

A screenshot of a computer program

Description automatically generated

1. **Conclusion**

The getNextToken() method retrieves the next token from the input file using the next() method of Scanner. If no more tokens are available, it returns null to indicate the end of the file. The program reads tokens from a file and retrieves a token, then the next token, get the corresponding lexeme symbol, and then its line number. The program's main method demonstrates the scanner by printing the input file contents and the lexemes with their symbols. The lexeme-symbol mappings are defined in a hashmap.

SOURCE CODE:  
package projects;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.HashMap;

import java.util.Scanner;

public class JuliaScanner {

private Scanner fileScanner;

private int lineNumber;

private String currentToken;

private HashMap<String, String> lexemeSymbolTable;

public String getNextToken() {

// continuously loop while there are more tokens to read from the file

while (fileScanner.hasNext()) {

// Read the next token from the file

String token = fileScanner.next();

// check if the token is not blank (contains non-whitespace characters)

if (!token.isBlank()) {

// assign the current token to the field (no longer used in the code)

currentToken = token;

// return the token as the next token in the sequence

return token;

}

}

// return null if there are no more tokens in the file

return null;

}

public String getLexemeSymbol(String lexeme) {

return lexemeSymbolTable.getOrDefault(lexeme, "IDENTIFIER");

}

public int getLineNumber() {

// return the current line number being processed

return lineNumber;

}

public static void main(String[] args) {

try {

Scanner userInput = new Scanner(System.in);

System.out.println("Enter File Location:");

String filePath = userInput.nextLine(); // Read user input

JuliaScanner scanner = new JuliaScanner(filePath);

String token;

System.out.println("\nInput File Contents:");

// print the input file contents

while ((token = scanner.getNextToken()) != null) {

System.out.println(token);

}

scanner.close();

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.out.println("Lexeme Symbol");

System.out.println("-------------------------");

// reinitialize the scanner for lexeme and symbol printing

scanner = new JuliaScanner(filePath);

// print the lexemes and their symbols

while ((token = scanner.getNextToken()) != null) {

int lineNumber = scanner.getLineNumber();

String symbol = scanner.getLexemeSymbol(token);

System.out.printf("%-15s %s\n", token, symbol);

}

scanner.close();

} catch (FileNotFoundException e) {

System.err.println("File not available: " + e.getMessage());

}

}

// lexeme table as seen in the sample output

public JuliaScanner(String filename) throws FileNotFoundException {

fileScanner = new Scanner(new File(filename));

lineNumber = 1;

currentToken = null;

lexemeSymbolTable = new HashMap<>();

lexemeSymbolTable.put("function", "FUNCTION");

lexemeSymbolTable.put("(", "OPEN\_PARENTHESIS");

lexemeSymbolTable.put(")", "CLOSE\_PARENTHESIS");

lexemeSymbolTable.put("=", "ASSIGNMENT\_OPERATOR");

lexemeSymbolTable.put("!=", "NOT\_EQUAL\_OPERATOR");

lexemeSymbolTable.put("if", "IF");

lexemeSymbolTable.put("then", "THEN");

lexemeSymbolTable.put("print", "FUNCTION");

lexemeSymbolTable.put("else", "ELSE");

lexemeSymbolTable.put("end", "END");

}

public void close() {

fileScanner.close();

}

}