**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет   
імені Петра Могили**

**Факультет комп’ютерних наук**

**Кафедра інженерії програмного забезпечення**

**ЛАБОРАТОРНА РОБОТА №2**

з дисципліни «Теорія автоматів та формальних мов»

на тему: **«**Лексичний аналіз. Регулярні вирази**»**

121 – ЛР.ПЗ.02 – 308.1710809

Виконав: студент 3 курсу, групи 308

спеціальності 121 «Інженерія програмного забезпечення», **Зурілов І.М.**

Керівник: професор, **Фісун М.Т.**

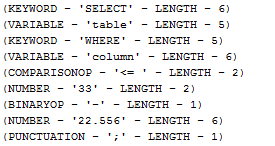
**м. Миколаїв – 2019 рік**

Код лексичного аналізатора наведено у додатку.

**ЗАПИТ :**

**SELECT table WHERE column <= 33 - 22.556;**

**РЕЗУЛЬТАТ :**

****

**Додаток**

**package lexer;**

**import java.util.ArrayList;**

**import java.util.regex.Pattern;**

**import java.util.regex.Matcher;**

**public class Lexer {**

**public static enum TokenType {**

**NUMBER("[-+]?[0-9]\*\\.?[0-9]+"), VARIABLE("[a-z$\_][a-zA-Z0-9$\_]\*"), BINARYOP("[\*|/|+|-]"),**

**KEYWORD("\\b(?:INSERT|UPDATE|DELETE|CREATE TABLE|FROM|UNION|GROUP BY|SELECT|WHERE)\\b"),**

**COMPARISONOP("(=|>|<|>=|<|<=|<>)\\s"), WHITESPACE("[\t\f\r\n]+"), PUNCTUATION("[.|;|!]");**

**public final String pattern;**

**private TokenType(String pattern) {**

**this.pattern = pattern;**

**}**

**}**

**public static class Token {**

**public TokenType type;**

**public String data;**

**public String inputScript;**

**public Token(TokenType type, String data) {**

**this.type = type;**

**this.data = data;**

**}**

**@Override**

**public String toString() {**

**return String.format("(%s - \'%s\' - LENGTH - %d)", type.name(), data, data.replace(" ", "").length());} }**

**public static ArrayList<Token> lex(String input) {**

**// The tokens to return**

**ArrayList<Token> tokens = new ArrayList<Token>();**

**// Lexer logic begins here**

**StringBuffer tokenPatternsBuffer = new StringBuffer();**

**for (TokenType tokenType : TokenType.values())**

**tokenPatternsBuffer.append(String.format("|(?<%s>%s)", tokenType.name(), tokenType.pattern));**

**Pattern tokenPatterns = Pattern.compile(new String(tokenPatternsBuffer.substring(1)));**

**// Begin matching tokens**

**Matcher matcher = tokenPatterns.matcher(input);**

**while (matcher.find()) {**

**if (matcher.group(TokenType.NUMBER.name()) != null) {**

**tokens.add(new Token(TokenType.NUMBER, matcher.group(TokenType.NUMBER.name())));**

**continue;**

**}**

**else if (matcher.group(TokenType.BINARYOP.name()) != null) {**

**tokens.add(new Token(TokenType.BINARYOP, matcher.group(TokenType.BINARYOP.name())));**

**continue;**

**}**

**else if (matcher.group(TokenType.WHITESPACE.name()) != null) {**

**tokens.add(new Token(TokenType.WHITESPACE, matcher.group(TokenType.WHITESPACE.name())));**

**continue;**

**}**

**else if (matcher.group(TokenType.KEYWORD.name()) != null) {**

**tokens.add(new Token(TokenType.KEYWORD, matcher.group(TokenType.KEYWORD.name())));**

**continue;**

**}**

**else if (matcher.group(TokenType.VARIABLE.name()) != null) {**

**tokens.add(new Token(TokenType.VARIABLE, matcher.group(TokenType.VARIABLE.name())));**

**continue;**

**}**

**else if (matcher.group(TokenType.COMPARISONOP.name()) != null) {**

**tokens.add(new Token(TokenType.COMPARISONOP, matcher.group(TokenType.COMPARISONOP.name())));**

**continue;**

**}**

**else if (matcher.group(TokenType.PUNCTUATION.name()) != null) {**

**tokens.add(new Token(TokenType.PUNCTUATION, matcher.group(TokenType.PUNCTUATION.name())));**

**continue;**

**}**

**}**

**return tokens;**

**}**

**public static void main(String[] args) {**

**String input = "SELECT table WHERE column <= 33 - 22.556;";**

**// Create tokens and print them**

**ArrayList<Token> tokens = lex(input);**

**for (Token token : tokens)**

**System.out.println(token);**

**}**

**}**