**Experiment No. 8**

**Aim:** Implementation of Lexical Analyzer

**Requirement:** Compatible version of java

**Theory:**

The first phase of a compiler is called lexical analysis or scanning. The lexical analyzer reads the stream of characters making up the source program and groups the characters into meaningful sequences called lexemes. the lexical analyzer produces as output a token of the form: <token-name; attribute-value> that it passes on to the subsequent phase, syntax analysis.

Token-name is an abstract symbol that is used during syntax analysis.

Attribute-value points to an entry in the symbol table for this token.

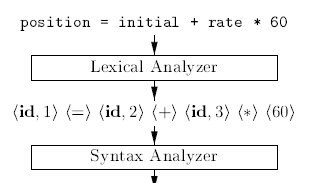
Information from the symbol-table entry is needed for semantic analysis and code generation. For example, suppose a source program contains the assignment statement

position = initial + rate \* 60

The representation of the above assignment statement after lexical analysis as the sequence of tokens

<id, 1> <=> <id, 2> <+> <id, 3> <\*> <60>

In this representation, the token names =, +, and \* are abstract symbols for the assignment, addition, and multiplication operators, respectively.



**Code:**

import java.util.\*;

import java.io.\*;

class Lex {

    public static void main(String[] args) {

        List<String> keywords = Arrays.asList("break","else","auto","case","char","const","continue","default","do","double","else","enum","extern","float","for","goto","if","int","long","register","return","short","signed","sizeof","static","struct","switch","typedef","union","unsigned","void","volatile","while");

        List<String> operators = Arrays.asList("+","-","\*","/","%","==","!=","||","=","|","&&","&","{","}","(",")");

        List<String> delimiter = Arrays.asList(";");

        List<String> functions = Arrays.asList("printf","main","scanf");

        Vector<String> identifiers = new Vector<>();

        try{

            BufferedReader reader = new BufferedReader(new FileReader("code.txt"));

            String line = reader.readLine();

            int i;

            while(line!=null) {

                if(line.length()==1) {

                    if(operators.contains(line)) System.out.print("Operator"+operators.indexOf(line)+" ");

                }

                else {

                    String temp="";int igb=0;

                    for(i=0;i<line.length();i++) {

                        if(line.charAt(i) == ' ') {

                            if(keywords.contains(temp)) System.out.print("Keyword"+keywords.indexOf(temp)+" ");

                            else if(operators.contains(temp)) System.out.print("Operator"+operators.indexOf(temp)+" ");

                            //System.out.print(temp+"//");

                            temp="";

                        }

                        else if(line.charAt(i) == '(') {

                            if(functions.contains(temp)) {

                                System.out.print("Function"+functions.indexOf(temp)+" ");

                                igb=1;

                                temp="";

                            }

                            else {

                                System.out.print("Operator"+operators.indexOf("(")+" ");

                            }

                            //System.out.print(temp+"//");

                        }

                        else if(line.charAt(i) == ')') {

                            if(igb==1) igb=0;

                            else {

                                if(temp!="") {

                                    if(keywords.contains(temp)) System.out.print("Keyword"+keywords.indexOf(temp)+" ");

                                    else if(functions.contains(temp)) System.out.print("Function"+functions.indexOf(temp)+" ");

                                    else {

                                        System.out.print("Identifier");

                                        if(!identifiers.contains(temp)) identifiers.add(temp);

                                        System.out.print(identifiers.indexOf(temp)+" ");

                                    }

                                }

                                System.out.print("Operator"+operators.indexOf(")")+" ");

                            }

                            //System.out.print(temp+"//");

                            temp="";

                        }

                        else if(igb==1) continue;

                        else if(line.charAt(i) == ',') {

                            if(keywords.contains(temp)) System.out.print("Keyword"+keywords.indexOf(temp)+" SpecialCharacter");

                            else if(functions.contains(temp)) System.out.print("Function"+functions.indexOf(temp)+" SpecialCharacter");

                            else {

                                System.out.print("Identifier");

                                if(!identifiers.contains(temp)) identifiers.add(temp);

                                System.out.print(identifiers.indexOf(temp)+" SpecialCharacter ");

                            }

                            //System.out.print(temp+"//");

                            temp="";

                        }

                        else if(operators.contains(Character.toString(line.charAt(i)))) {

                            if(temp!="") {

                                if(keywords.contains(temp)) System.out.print("Keyword"+keywords.indexOf(temp)+" ");

                                else if(functions.contains(temp)) System.out.print("Function"+functions.indexOf(temp)+" ");

                                else {

                                    System.out.print("Identifier");

                                    if(!identifiers.contains(temp)) identifiers.add(temp);

                                    System.out.print(identifiers.indexOf(temp)+" ");

                                }

                            }

                            System.out.print("Operator"+operators.indexOf(Character.toString(line.charAt(i)))+" ");

                            //System.out.print(temp+"//");

                            temp="";

                        }

                        else if(delimiter.contains(Character.toString(line.charAt(i)))) {

                            if(temp!="") {

                                if(keywords.contains(temp)) System.out.print("Keyword"+keywords.indexOf(temp)+" ");

                                else if(functions.contains(temp)) System.out.print("Function"+functions.indexOf(temp)+" ");

                                else {

                                    System.out.print("Identifier");

                                    if(!identifiers.contains(temp)) identifiers.add(temp);

                                    System.out.print(identifiers.indexOf(temp)+" ");

                                }

                            }

                            System.out.print("Delimiter ");

                            //System.out.print(temp+"//?");

                            temp="";

                        }

                        else temp+=Character.toString(line.charAt(i));

                    }

                }

                System.out.println();

                line = reader.readLine();

            }

            System.out.println("Identifiers: "+identifiers);

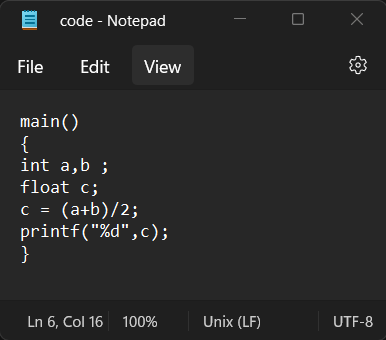
        }catch(IOException e){}

    }

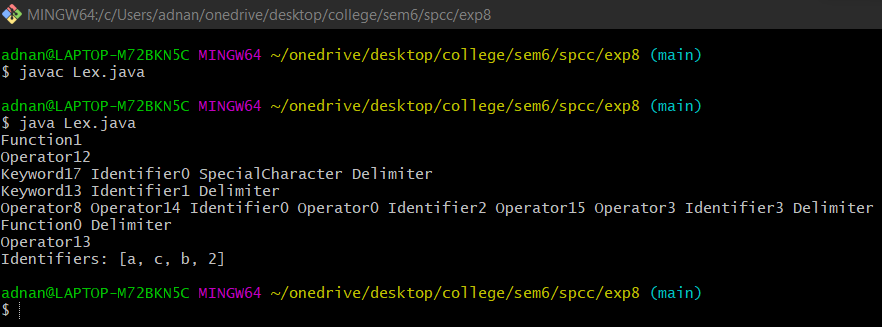
}

**Output:**

**Input File:**



**Execution:**



**Conclusion:** Thus we have successfully implemented Lexical Analyzer in Java.