# Documentado

## Clase analizador

package analizdor;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @Brayan Merlos

\*/

public class Analizador {

static Lexico m=new Lexico();

static Sintactico s=new Sintactico();

public static String linea;

public static String Ubicacion = "C:\\Compiladores", Nombre="\\tokens.txt";

public String codigo() throws FileNotFoundException, IOException{

FileReader fr = new FileReader(Ubicacion+Nombre);

BufferedReader br = new BufferedReader(fr);

String lineaI;

String codigo = "";

while((lineaI=br.readLine())!=null){

codigo +=lineaI;

}

return codigo;

}

public static void main(String[] args) throws InstantiationException, IllegalAccessException {

// TODO code application logic here

File archivo = null;

FileReader fr = null;

BufferedReader br = null;

try {

archivo = new File (Ubicacion,Nombre);

fr = new FileReader (archivo);

br = new BufferedReader(fr);

// Lectura del fichero

System.out.println(" TABLA DE SIMBOLOS ");

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

while((linea=br.readLine())!=null){

m.analizar(linea);

}

}

catch(Exception e){

e.printStackTrace();

}finally{

try{

if( null != fr ){

fr.close();

}

}catch (Exception e2){

e2.printStackTrace();

}

}

if(m.errorlex){

System.out.println();

System.out.println(" TABLA DE ERRORES LEXICOS ");

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

for (int i = 0; i < m.errorl.size(); i++) {

System.out.println(m.errorl.get(i)+" "+"Cadena no reconocida");

}

}else{

System.out.println();

System.out.println(" ANALISIS SINTACTICO ");

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

s.AnalizarSintac(m.tokens);

s.Equilibrio();

if(s.error){

System.out.println();

System.out.println(" TABLA DE ERRORES SINTACTICOS ");

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

for(int i=0;i<s.errors.size();i++){

System.out.println(s.errors.get(i));

}

s.Equilibrio();

}else{

System.out.println();

System.out.println(" ANALISIS SEMANTICO");

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

}

}

}

}

## Clase “clase principal”

package analizdor;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

\*

\* @author Brayan Merlos

\*/

public class ClasePrincipal {

public static void main(String[] args) {

Lexico lexi =new Lexico();

System.out.println("mostrando palabras reservadas del analizador lexico");

String patron = ("(while)|((a-zA-Z)+)|([>|<|=]+)|([0-9]+)|([(|)]+)|([(|)]+)|(;)");//definimos la expresion regular

String texto = "while (x=0) < (y=0)";//texto que queremos analizar con la expresion regular

Pattern p = Pattern.compile(patron);//compilar nuestra expresion regular

Matcher matcher = p.matcher(texto);

//nos permita encontrar las coincidencias "palabras que hacen parte de nuestra expresion regular"

while (matcher.find()) {

//variables para que coincidad

String tokenTipo1 = matcher.group(1);

if (tokenTipo1 != null) {

System.out.println("palabra reservada: " + tokenTipo1);

}

String tokenTipo2 = matcher.group(2);

if (tokenTipo2 != null) {

System.out.println("variables: " + tokenTipo2);

}

String tokenTipo3 = matcher.group(3);

if (tokenTipo3 != null) {

System.out.println("operador: " + tokenTipo3);

}

String tokenTipo4 = matcher.group(4);

if (tokenTipo4 != null) {

System.out.println("numero: " + tokenTipo4);

}

String tokenTipo5 = matcher.group(5);

if (tokenTipo5 != null) {

System.out.println("parentesis: " + tokenTipo5);

}

String tokenTipo6 = matcher.group(6);

if (tokenTipo6 != null) {

System.out.println("llaves: " + tokenTipo6);

}

String tokenTipo7 = matcher.group(7);

if (tokenTipo7 != null) {

System.out.println("numero: " + tokenTipo7);

}

}

}

}

## Clase “Lexico”

package analizdor;

import java.util.ArrayList;

import java.util.StringTokenizer;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

\*

\* @Brayan Merlos

\*/

public class Lexico {

public ArrayList<String> errorl=new ArrayList<String>();

public ArrayList<String> tokens=new ArrayList<String>();

public ArrayList<String> identis=new ArrayList<String>();

public boolean errorlex;

String a="Integer";

String b="";

String c="";

String d="";

// a="Integer";

// b="Boolean";

// c="Char";

// d="Double";

public void analizar(String linea){

StringTokenizer st=new StringTokenizer(linea, " =;()><[]{}+-/\*,",true);

String Vec[]=new String[st.countTokens()];

while(st.hasMoreTokens()){

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

Vec[i]=st.nextToken();

if(palabraReservada(Vec[i])){

System.out.println(Vec[i]+" "+"Palabra Reservada");

a=a+Vec[i];

}else if(operadorAritmetico(Vec[i])){

System.out.println(Vec[i]+" "+"Operador Aritmetico");

a=a+Vec[i];

}else if(isNumeric((Vec[i]))){

System.out.println(Vec[i]+" "+"Constante Entera Valida");

}else if(operadorRelacional(Vec[i])){

System.out.println(Vec[i]+" "+"Operador Relacional");

a=a+Vec[i];

}else if(Cadena(Vec[i])){

System.out.println(Vec[i]+" "+"Cadena de Caracteres");

}else if(Identificador(Vec[i])=="1"){

System.out.println(Vec[i]+" "+"Identificador");

}else if(Identificador(Vec[i])=="2"){

errorl.add(Vec[i]);

errorlex=true;

}

}

}

tokens.add(a);

}

public boolean palabraReservada(String cad){

Pattern pat=Pattern.compile("what|how|i|when|whai|that|is|not|are"

+ "we|you|i|she|he|they|it|its|are"

);

Matcher mat=pat.matcher(cad.toUpperCase());

if(mat.matches()){

return true;

}else{

return false;

}

}

public boolean operadorAritmetico(String cad){

Pattern pat=Pattern.compile("(-|/)");

Matcher mat=pat.matcher(cad.toLowerCase());

if(mat.matches() || cad.equalsIgnoreCase("\*") || cad.equalsIgnoreCase("+")){

return true;

}

else{

return false;

}

}

public boolean operadorRelacional(String cad){

Pattern pat=Pattern.compile("(>=|<=|>|<|=|}|]|;|,)");

Matcher mat=pat.matcher(cad.toLowerCase());

if(mat.matches() || cad.equalsIgnoreCase("{") || cad.equalsIgnoreCase("[") || cad.equalsIgnoreCase("(") || cad.equalsIgnoreCase(")"))

return true;

else{

return false;

}

}

public boolean Cadena(String cad){

if(cad.startsWith("'") && cad.endsWith("'")){return true;}

else{return false;}

}

private boolean isNumeric(String cadena){

try {

Integer.parseInt(cadena);

return true;

} catch (NumberFormatException nfe){

return false;

}

}

public String Identificador(String cad) {

try{

if(cad.endsWith(" ")){return "3";}

if(isNumeric(String.valueOf(cad.charAt(0)))){return "2";}

if(operadorAritmetico(String.valueOf(cad.charAt(0))) && operadorAritmetico(String.valueOf(cad.charAt(1)))==false){return "2";}

if(operadorAritmetico(String.valueOf(cad.charAt(0))) && operadorAritmetico(String.valueOf(cad.charAt(1)))){return "2";}

}catch(IndexOutOfBoundsException nfe){

return "3";

}

identis.add(cad);

return "1";

}

}

## Clase “sintactico”

package analizdor;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Stack;

import java.util.StringTokenizer;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

\*

\* @Brayan Merlos

\*/

public class Sintactico {

int parentesis=0;

Lexico l=new Lexico();

public ArrayList<String> errors=new ArrayList<String>();

Stack pc = new Stack();

Stack pa = new Stack();

boolean error;

public void AnalizarSintac(ArrayList<String> t1){

for (int i = 0; i < t1.size(); i++) {

if(CrearTabla(t1.get(i))){

System.out.println("Linea: "+i+" Creacion de Tabla");

System.out.println("Linea: "+i+" Modificacion de Tabla");

// }else if(Select(t1.get(i))){

System.out.println("Linea: "+i+" Seleccion");

}else if(Insert(t1.get(i))){

System.out.println("Linea: "+i+" Insercion");

pa.add("(");

System.out.println("Linea: "+i+" Operador Relacional");

parentesis++;

}else if(t1.get(i).contains(")")){

pc.add(")");

System.out.println("Linea: "+i+" Operador Relacional");

parentesis--;

}else if(t1.get(i).equals("")){

}else{

errors.add("Linea: "+i+" Error Sintactico");

error=true;

}

}

}

public void Equilibrio(){

if(pa.size()!=pc.size())

{

errors.add("Error Sintactico hacen falta parentesis");

error=true;

}

else{

System.out.println("Parentesis equilibrados");

}

}

public boolean Insert(String linea){

String linea2=linea.replace(",", "");

int result1=linea2.compareToIgnoreCase("INSERTINTO");

int result2=linea2.compareToIgnoreCase("VALUES()");

int result3=linea2.compareToIgnoreCase("INSERTINTOVALUES()");

if(result1==0||result2==0||result3==0){

return true;

}else{

return false;

}

}

public boolean CrearTabla(String linea){

int result1=linea.compareToIgnoreCase("CREATETABLE");

if(result1==0){

parentesis++;

return true;

}else{

return false;

}

}

public static void main(String[] args) {

String patron = ("(while)|((a-zA-Z)+)|([>|<|=]+)|([0-9]+)|([(|)]+)|([(|)]+)|(;)");//definimos la expresion regular

String texto = "while (x=0) < (y=0)";//texto que queremos analizar con la expresion regular

Pattern p = Pattern.compile(patron);//compilar nuestra expresion regular

Matcher matcher = p.matcher(texto);

//nos permita encontrar las coincidencias "palabras que hacen parte de nuestra expresion regular"

while (matcher.find()) {

//variables para que coincidad

String tokenTipo1 = matcher.group(1);

if (tokenTipo1 != null) {

System.out.println("palabra reservada: " + tokenTipo1);

}

String tokenTipo2 = matcher.group(2);

if (tokenTipo2 != null) {

System.out.println("variables: " + tokenTipo1);

}

String tokenTipo3 = matcher.group(3);

if (tokenTipo3 != null) {

System.out.println("palabra reservada: " + tokenTipo1);

}

String tokenTipo4 = matcher.group(4);

if (tokenTipo4 != null) {

System.out.println("digitos: " + tokenTipo1);

}

String tokenTipo5 = matcher.group(5);

if (tokenTipo5 != null) {

System.out.println("palabra reservada: " + tokenTipo1);

}

String tokenTipo6 = matcher.group(6);

if (tokenTipo6 != null) {

System.out.println("palabra reservada: " + tokenTipo1);

}

String tokenTipo7 = matcher.group(7);

if (tokenTipo7 != null) {

System.out.println("palabra reservada: " + tokenTipo1);

}

}

}

}