package Gramatica;

import Lectura.LeerArchivo 1; import Lectura.NoEsDeString;

public class AcomodoGramatica { private String[] gramaticaCompleta;

private String[J [] t od as L as P roducciones; private String[] ladosDerechos;

public String[] sirri.bolo s No T e rmin a l es;

public String[J s i m.bolo s T e rminales; private String[J te mp ;

p rivate final LeerArchivo 1 leerA;

private final NoEsDeString nes;

public AcomodoGarmatica() { leerA = new LeerArchivo\_l(); nes = new NoEsDeString(};

}

private void leerGramatica() leerA.leerArchivo();

public void ini(} leerGramatica(}; separarProducciones(); definirArreglos(}; todasLasProducciones(); ladosDerechos(}; simbolosNoTerminales(); si!llbolo s T er min a l es ( ) ;

// S y ste m . o u t. p rintln(nes.cantidadDeCadenas(}};

//System.out.println(nes.NoEsTrim(gramatica}};

}

private void definirArreglos()

todasLasProducciones = new String[nes.cantidadDeCadenas()]

[nes.cantidadDeCadenas(} J ;

l a do sDe rechos = new String[nes.cantidadDeCadenas() J ;

s imbolo s N o T e rmin a l es = new String[nes.cantidadDeCadenas()]; simbolosTerminales = new String[nes.ca.ntidadDeCadenas(}+30]; ternp = new String[3 J ;

}

p r iv ate void todasLasProducciones() {

for (int i = O; i < gramaticaCompleta.length-1; i++) { temp = nes.noEsEsplit(gramaticaCompleta[i], ">");

todasLasProducciones[i][O] = (temp[O]}.trim(}; // lado izquierdo todasLasProducciones[i][l] = (temp[l]}.trim(}; // lado derecho

}

}

private void ladosDerechos() {

for (int i = O; i < todasLasProducciones.length; i++) { ladosDerechos[i] = todasLasProducciones[iJ [ 1 ) ;

}

}

p rivate void sirobolo s N o T e rminales() { int i = O, j = O;

while (j < todasLasProducciones.length) {

if (! existeEnArreglo(simbolosNoTerminales,

todasLasProducciones[j][OJ}} {

s imbolo s N o T e rminales{i] = todasLasProducciones{ j ] [O] . trim();

i++;

}

j++;

}

}

private void si!llbolo s T e rminales() { int *i* = O, j = O;

// imprime(si!llbolo s No T e rmin a l es } ;

w hil e ( j < t od as L as P r od u ccion es. l e ng t h )

te mp = nes.noEsEsplit(todasLasProducciones[ j J [ 1 ] , " " ) ;

f o r ( S t ring tempDerecha : temp) { // 949 - vacio - ' e '

if (! existeEnArreglo(sirobolo s N o T e rminales, tempDerecha)&&

((int}tempDerecha.charAt(O} •= 949 }

&& !e x isteE nA rreglo(si!llbolo s T er min a l es, te mp De recha))

sirobolo s T e rmin a l es [ i ] =

tempDerecha;//todasLasProducciones[j][1];

i++;

}

}

j++;

}

}

private boolean existeEnArreglo(String[J arre glo , Str ing b u s c ar ) f o r (String buscarEn : arreglo) {

if (buscarEn ! = null)

if (buscarEn.equals(buscar}} return true;

}

return false;

}

public int indiceNoTerminal(String buscar) {

for (int i = O; i < simbolosNoTerminales.length; i++)

if (sirobolo s No T e rminales[i] ! = null)

// System.out.println("simbolo nt:"+simbolosNoTerminales[i]); if (simbolosNoTerminales[iJ .e q u a l s ( b u s c ar }} {

// S y ste m . o u t. p rintln("encontrado no terminal "+i); return i;

}

}

return O;

}

public int indiceTerminal(String buscar) {

fer (int *i* = O; *i* < simbolosTerminales.length; i++)

if (sirobolo s T e rmin a l es [ i ] ! = n u ll }

// S y ste m . o u t. p r in t ln ( "s i m.bolo t:" + s im.bolo s T e rrnina l es [ i ]) ; i f ( s imbolo s T e rminales[iJ .e q u a l s ( b u s c a r}}{

// System.out.println("encontrado terminal "+i);

return i;

}

return O;

}

public String[J p roduccionDerecha(int produccion) {

if (todasLasProducciones[produccion - 1][1] ! = null)

return nes.noEsEsplit(todasLasProducciones[produccion - 1][1], 11 " ) ;

}

return null;

}

private void imprirne G ra m at ic a Compl eta ( )

f o r (String[] prod : todasLasProducciones)

//System.out.println(prod[OJ + "\t->\t" + prod[l]}; System.out.printf("10s 3s 10s", prod[OJ ) ;

p u b lic *void* imprei(m)

Systme.out.pritnln(11\tGramática compelta)"; imrpimeGramaticaCompelt(a);

Systme.out.println("\n\tLados Derec:los" ) ; imp rime(ladosDerechos);

System.out.println("\n\tSírobolo s N o T e rminales"); imprime(simbolosNoTerminales); System.out.println("\n\tSÍ!llbolo s T e rminales"); imprime(simbolosierminales);

}

private void separarProducciones() {

gramaticaCompleta = nes.noEsEsplit(leerA.datos(), 11\n");

}

private void imprime(String[] arreglo) { fer (String prod : arreglo} {

if (prod ! = null} System.out.println(prod);

}

}

public String simboloinicail(} { return si!llbolo s N o T er min a l es [ O J ;

}

}

cl ass test {

public static void main(String[] args) AcomodoGramatica ag = new AcomodoGramatica(); ag.ini(};

ag.imprime(};

}

}

**pac kage analizado r.lex ico.c;**

**irnpor t L e c t ura . L ee r Archivo; irnpor t Est ruc t ur as. Li stas R ;**

**public class ClasificaRebuilt Tipos tipo = new Tipos();**

**ConversionCaracter conv = new ConversionCaracter();**

**String archivo = "", token; LeerArchivo leer = new LeerArchivo(); ListasR listaTab;**

**PalabraReservada palR;**

**int actual = O;**

public ClasificaRebuilt() **leer.leerArchivo(); archivo = leer.datos();**

palR = new Palab ra Rese rvada ();

**c rea rLi stas T();**

)

**public void retroceder(){ actual=actual-6;**

**listaTab.agregarElerne n t oLSi rnbolo s R("b e gin ", t ipoP a lab ra ( " b e gin " ), li sta T a b . bu s c a R e pR( " b e gin" ) - 1, calValToken("begin"), -1, null);**

)

**private void crearListasT()** {

**listaTab = new ListasR();**

public String pedirToken() { q0(archivo);

**Systern. o u t. prin t ln( t o ke n) ;**

**ret u r n t o ke n ;**

)

public void q0(String archivo) { if (actual < archivo.length())

**conv.convertirCaracter(archivo.charAt(actual));**

if (tipo.esEspacio(conv.getAscii())) { if (actual + 1 < archivo.length()) {

**actual++;**

q0(archivo);

)

//NOTA: CREO FALTA AGREGAR ELSE

**else if ((tipo.esMinuscula(conv.getAscii())**

(tipo.esMayuscula(conv.getAscii()) == tr u e )) {

**a c t u a l++ ;**

**qlid e n t i ficador(archivo);**

true) 11

**else if (tipo.esNumero(conv.getAscii()) actua l++;**

**q2Num e roEnte ro( a rchivo );**

t rue) {

**else if (tipo.esParentesisl(conv.getAscii()) == true)** {

**actual++;**

**token = crearCadena(actual - 1, actual + 1, archivo);**

/•Token J Tipo To ke n J V a lo r To ke n J V a lo r Id e n tf J Tipo Id e n t i f l fV e c es R e pi te ¡ • /

**li st aTab . agr e gar E l e m e n t o L Si rnbolo s R ( t o ke n, " Simb . Es p." ,**

li sta Tab .b u sc a R e pR( t ok e n) + 1, (in t ) ( t o ke n .ch ar A t (O)) + 300, O, null);

// qlidentificador(archivo);

**else if (tipo.esParentesis2(conv.getAscii()) == true)** {

**actual++;**

**token = crearCadena(actual - 1, actual + 1, archivo); listaTab.agregarElerne n t oLSi rnbolo s R ( t o ke n, "Sirnb . Es p ." ,**

li sta T a b . b u sc a R e pR( t o ke n) + 1, (in t ) ( t o ke n .ch a rA t (O)) + 300, O, null);

// qlidentificador(archiv)o;

else if (tipo.esComa(conv.getAscii()) == t ru e ) {

**a c t u a l++ ;**

**t o ke n = c rea rC a d e n a ( a c t u a l - 1, actual + 1, archivo); listaTab.agregarElementoLSimbolosR(token, "Simb. Esp.**11 ,

listaTab.buscaRepR(token) + 1, (int) (token.charAt(0)) + 300, O, null);

// qlidentificador(archivo);

else if (tipo.esPyC(conv.getAscii()) == t ru e )

**ac t u al++ ;**

**t o ke n = c rea rC a d e n a ( a c t u a l - 1, actual + 1, archivo); listaTab.agregarElementoLSirnbolo s R ( t o ke n, "Sirnb . Es p ." ,**

li sta Tab . bu sc a R e pR( t ok e n ) + 1, (in t ) ( t o ke n .ch ar A t (0)) + 300, O, null);

// qlidentificador(archiv)o;

**else if (tipo.esDPuntos(conv.getAscii()) == true)** {

**actual++; q3Asignacion(archivo);**

else if (tipo.esMas(conv.getAscii()) == t r u e )

**a c t u a l++ ;**

**t o ke n = c rea rC a d e n a ( a c t u a l - 1, actual + 1, archivo); listaTab.agregarElementoLSimbolosR(token, "Simb. Esp.**11 ,

listaTab.buscaRepR(token) + 1, (int) (token.charAt(O)) + 300, O, null);

// qlidentificador(archivo;)

**else if (tipo.esMenos(conv.getAscii()) == true) actual++;**

**token = crearCadena(actual - 1, actual + 1, archivo);**

**listaTab.agregarElerne n t oL S irnbolo s R ( t o ke n, 11 S irnb. Es p .** 11 ,

li sta Tab . b u sc a R e pR( t o ke n) + 1, (in t ) ( t o ke n .ch a rA t (O)) + 300, O, null);

// qlidentificador(archivo); else {

**actual++; qErrorLexico(archivo, O);**

**public void q1Identificador(St ring archivo)** {

**int movs = 1;**

**fer (int i = actual; i < archivo.length(); i++) conv.convertirCaracter(archivo.charAt(i));** if (tipo.esEspacio(conv.getAscii())) {

**token = crearCadena(actual - 1, actual + movs, archivo);**

**listaTab.agregarElerne n t oL S irnbolo s R ( t o ke n, t ipo Pa l a b ra ( t o ke n), li sta Tab . b u s c a R e pR( t o ke n) + 1, calValToken(token), -1, null);**

if (!palR.ExistePalabraReservada(token)) {

**token = "ID";**

**actual = actual + movs; break;**

else if ((tipo.esMinuscula(conv.getAscii()) == **t rue)**

11 (t ipo .es May uscula(conv .get Ascii ()) == **t rue)**

11 (t ipo .es Num er o(conv .get Ascii ())

**movs ++;**

else {

t ru e)) {

**to ken = c rearCad ena(ac tual - 1, actual + movs, archivo); listaTab.agregarElementoLSirnbolo s R( t o ke n, t ipoP a lab ra ( t o ke n),**

**li sta T a b . bu s c a R e pR( t o ke n) + 1, calValToken(token), -1, null);**

if (!palR.ExistePalabraReservada(token)) {

**to ke n = "I D";**

)

**act ual = act ua l + mov s - 1;**

// **q Er ro rL ex ico(archivo, mov s ); br ea k;**

)

**public void q2NumeroEntero(St ring archivo)** {

**int movs = 1;**

**fer (int i = actual; i < archivo.length(); i++) conv.convertirCaracter(archivo.charAt(i));** if (tipo.esEspacio(conv.getAscii())) {

**token = crearCadena(actual - 1, actual + movs, archivo);**

**listaTab.agregarElementoLSirnbolo s R( t o ke n, " Núm e ro ", li st aTab . b u s c a R e pR( t o ke n)** +

**1, 500, token, null);**

**token = "int";**

**actual = actual + movs; break;**

**else if (tipo.esNumero(conv.getAscii())** true) {

**movs++;**

else {

**token = crearCadena(actual - 1, actual + movs, archivo); token = "int";**

**actual = actual + movs - 1;**

// **qErrorLexico(archivo, movs); break;**

**public void q3Asignacion(St ring archivo)** {

**int movs = 1;**

**fer (int i = actual; i < archivo.length(); i++) conv.convertirCaracter(archivo.charAt(i));** if (tipo.esEspacio(conv.getAscii())) {

**token = crearCadena(actual - 1, actual + movs, archivo); listaTab.agregarElementoLSimbolosR(token, "Simb. Esp.**11 ,

listaTab.buscaRepR(token) + 1, (int) (token.charAt(O)) + 300, O, null);

**actual = actual + movs; break;**

else if (tipo.esigual(conv.getAscii()) true) {

**movs++;**

//NOTA: SOLO DEBE HABER UNO

else {

**token = crearCadena(actual - 1, actual + movs, archivo); actual = actual + movs - 1;**

// **qErrorLexico(archivo, movs); break;**

**public void qErrorLexico(St ring archivo, int movs)** {

**fer (int i = actual + movs; i < archivo.length(); i++) conv.convertirCaracter(archivo.charAt(i));**

if (tipo.esEspacio(conv.getAscii()) == t r u e )

**mov s ++ ;**

**t o ke n = c rea rCad e n a (ac t u a l - 1, actual + movs, archivo); actual = actual + movs;**

**break;**

else {

**movs ++;**

)

**public String crearCadena(int i, int f, String archivo)** {

**String cad** 11" ;

fer (int j i ; j < f - 1; j ++) {

**cad = cad + archivo.charAt(j);**

**return cad;**

public void imprimeTablas() {

**listaTab.mostrarListaSirnbolo s R( );**

)

public String tipoPalabra(String token) { if (palR.ExistePalabraReservada(token))

**return "Palabra Re.";**

else {

**return "Identificador";**

)

public int calValToken(String Token) {

if (palR.ExistePalabraReservada(token))

**return palR.getValorPalabraReservada(Token);**

else {

if (listaTab.ExistePalabraT(token)) listaTab.buscaRepR(token);

) else {

if (listaTab.ExistePalabraT(token))

**return listaTab.buscaident(token);**

else {

**return listaTab.ultimoEnFila() + 1;**

//return -1;

**return O;**

public static void main(String[ J a rg s ) {

**Cl a si f ic a R e buil t obj = n e w Cl a si f ic a R e b u il t () ;**

// **L ee r Archivo leer = new LeerArchivo();**

// leer.leerArchivo();

**//String archivo = "Este .9 Este O 00 .9 es 002 OOt un Ar rA rA3 rA3 .s 4.3e4 a rchivo**

archivo2 archi. O. de prueba 12 12.12 O O 00 edwsd 12 12.1 .

**//String archivo2** = ";";

//Stringarchivo3= leer. datos();

// obj .q0( a rchi vo2);

// obj .im(archivo );

**whil e (!obj .pedi rTo ken() .equa ls("end"))**

obj .pedirTo ken(); obj .imp rimeTab las();

edsd li li U ";

# package analizador .lexico.c;

public class Convers ionCaracter {

private int ascii = O; private char carac;

public void convertirCaracter(char caracter){ ascii

//convi erte caracter a

ascii = (int) caracte r;

}

publíc ínt getAscíí(}{ ascii

// devuelve el valor en

return ascii;

}

public int getAscii(char caracter){ valor en ascii

return (int) caracter;

// devuelve el

}

public void convertirAscii (int nuro ro) { caracter

carac = (char)numero;

//convi erte ascii a

}

public char getCaracter(){ return carac;

//devuelve el caracter

}

1

package Lectura;

import java.awt.HeadlessException; import java.io.BufferedReader; import java.io.File;

import java.io.FileReader; import java.io.IOException; import javax.swing.JFileChooser;

import javax.swing.filechooser.FileNarneExte n s ionFil ter; public class LeerArchivo extends javax.swing.JFrame {

private JFileChooser ventana = null; private File archivo = null; private String cadena = 1111 ;

public void leerArchivo(){ String bfRead;

venta.na = new JFileChoose r(}; ventana.setCurrentDirectory(new File("src/Archivo s/ 11 ) ) ;

FileNam.eExte n s ionFil te r filtroTxt = new FileNam.eExte n s ionFil te r("IXT",

11txt11 ) ;

### ventana.setFileFilter(filtroixt); ventana.setDialogTitle("Abrir archivo");

if (venta.na.showOpenDialog(venta.na) == JFileGhooser. APPROVE\_OPTION) {

archivo = ventana.getSelectedFile(); try{

BufferedReader bf = new BufferedReader(new FileReader(archivo)); whíle ((bfRead = bf.readLíne()) ! = null)

cadena += bfRead + ' ';

} catch (HeadlessException I IO Ex c e pt ion ex ) { Sy ste m .o u t . p ríntln(ex.getHessage());

)

}

}

public String rutaArc ivo()

return ventana.getSelectedFile().toString();

}

publíc Stríng datos() { return cadena + ' ' ;

}

}

package Estructuras;

public class ListasR<dato> {

private NodoTSimR inicioSirn, finSirn;

p r iv ate NodoT Err o res inicio Err , f in Err; p r iv ate NodoTTok e n inicioTok, finTok; private NodoTReservadas inicioR, finR;

###### /•

Supuesta ta bl a d e s ímbolos

Token I Tipo Token

//V e c es R e pi te 1

w id e n t i f ic a do r

3

1 V a lor Tok e n V a lor Id e n tf Tipo Identif

115 o int

###### •/

class NodoTSimR<dato> {

public String token, tipoToken, tipoidentificador; dato valorToken, valoridentificador, vecesRepite; public NodoTSimR siguiente;

public NodoTSirnR(S tr ing t o ke n, S tr ing t ipoTo ke n, d at o v a lo rTok e n, d at o v a lorid e n t i f ic a dor, S t ring t ipoid e n t i f ic a dor, d at o v e c es R e pi te ) {

t hi s .t ok e n = t ok e n ;

t hi s .t ipoTo ke n = t ipoTo ke n ;

t hi s . v a lo rTok e n = v a lo rTok e n ;

t hi s. v a lorid e n t i f ic a dor = v a lorid e n t i f ic a dor ; t hi s.t ipoid e n t i f ic a dor = t ipoid e n t i f ic a dor ;

t is.vecesRepite = vecesRepite;

}

)

class NodoTErrores { public String error;

public NodoTErrores siguiente;

public NodoTErrores (String error) { this.error = error;

)

class NodoTToken {

public dato palabra; String tipoTok; int tokenTok; public NodoTToken siguiente;

public NodoTToken (dato palabra, String tipoTok, int tokenTok) this.palabra = palabra;

this.tipoTok = tipoTok; this.tokenTok = tokenTok;

}

)

class NodoTReservadas {

public String palabraR; int tokenR; public NodoTReservadas siguiente;

public NodoTReservadas (String palabraR, int tokenR) { this .palabraR = palabraR;

this.tokenR = tokenR;

)

)

/\* Inserta nuevos elero ntos Símbolos

Tipo Identificador Valor

###### •/

public void agregarElementoLSimbolosR(String token, String tipoToken, dato vecesRepite, dato valorToken, dato valoridentificador, String tipoidentificador) {

if (!ExistePalabraT(token))

NodoTSimR agregarElemento = new NodoTSimR(token, tipoToken, valorToken, valoridentificador, tipoidentificador, vecesRepite);

if (inicioSim != null) { // Existe el inicio

finSim.siguiente = agregarElemento; //Agregar al final de la finSim = agregarElemento;

) else {

inicioSirn = finSirn = a g re g arE l e m e n t o ; // C rea l a li sta con su

primer Nodo

}

)

} else

siExiste(token, vecesRepite);

public void siExiste(String token, dato vecesRepite) NodoTSirnR reco rrer = inicioSi rn;

whil e ( re co rrer ! = null ) {

i f (r e corr er.t o ke n .e qu a l s ( t o ke n )) {

// Sy ste m . ou t. p r in t ln ( " vr : 11 + v e c es R e pi te ) ; re cor re r . v e c es R e pi te = v e c es R e pi te;

b rea k ;

)

re co rrer = re co rre r . s igui e n te;

)

}

###### /•

Er ro res

P a l a b r a E rror

###### /

public void agregarElementoLErroresR(String palabra) { NodoTErrores agregarElemento = new NodoTErrores(palabra); if (inicioErr != null) {

finErr.siguiente = agregarElemento; finErr = agregarElemento;

} else {

inicioErr = finErr = agregarElemento;

)

)

###### /•

Tokens

Palabra Tipo Token

###### •/

public void agregarElementoLTokensR(dato palabra, String tipo, int token)

NodoTToken agregarElemento = new NodoTToken(palabra, tipo, token); if (inicioTok != null) {

finTok.siguiente = agregarElemento; finTok = agregarElemento;

) else {

inicioTok = finTok = agregarElemento;

)

)

###### /•

Reservadas Palabra Token

###### •/

t o ke n ) ;

public void agregarElementoLReser-va d as R(S tr ing p a l a b ra, in t t ok e n ) NodoTR ese rv a d as a gr e g a r E l e m e n t o = n e w NodoTR ese rv a d as (p a l a b r a ,

i f (inicioR != null) { finR.siguiente = agregarElemento; finR = agregarElemento;

else {

inicioR = finR = agregarElemento;

}

)

public void mostrarListaErroresR() NodoTErrores recorrer = inicioErr; while (recorrer != null) {

System.out.println(recorrer.error); recorrer = recorrer.siguiente;

}

)

public void mostrarListaSimbolosR() NodoTSirnR reco rrer = inicioSim ;

Sy ste ro. ou t. prin t ln ( " To ke n I Tipo Token Valor Token Valor Identf I Tipo Identif 1 #V e c es R e pi te" ) ;

whil e ( re co rre r != null ) {

Sy ste m .ou t. p rin tf ( "% - 10 s % 13s % 10 s % 17 s % 18 s % 15s % n ", re corr e r. t ok e n,

re co rrer. t ipoTo ke n ,

re co rrer. va lo rTok e n,

re co rrer. va lo r id e n t i f ic a do r, re corr e r .t ipoid e n t i f ic a dor, re co rrer. ve c es R e pi te ) ;

re co rre r = re co rre r .s ig u ie n te;

}

)

public void mostrarListaTokensR() NodoTToken recorrer = inicioTok; while (recorrer != null) {

System.out.println(recorrer.palabra + 11\t" +

recorrer.tipoTok + 11\t" +

recorrer.tokenTok); recorrer = recorrer.siguiente;

)

)

public void mostrarListaReservadasR() NodoTReservadas recorrer = inicioR; while (recorrer != null) {

System.out.println(recorrer.palabraR + 11\t" +

recorrer.tokenR + 11\t11

} ;

re co r rer = re co rrer. s igui e n te;

)

}

public boolean ExistePalabraT(String token) { NodoTSimR recorrer = inicioSim;

while (recorrer != null) {

if (recorrer.token.equals(token)} return true;

else

)

recorrer = recorrer.siguiente;

return false;

)

public int buscaident(String token) { NodoTSimR recorrer = inicioSim; while (recorrer != null) {

if (recorrer.token.equals(token)) return (int) recorrer.valorToken;

recorrer = recorrer.siguiente;

)

return O;

)

public int buscaRepR(String token) NodoTSirnR reco rrer = inicioSim ; int s ur.oEn = O;

whil e ( re co rrer != null ) {

i f ( re cor re r .t o ke n .e qu a ls ( t o ke n )) { sumEn = (int) recorrer.vecesRepite;

)

recorrer = recorrer.siguiente;

)

return sumEn;

)

public int ultimoEnFila() { NodoTSirri.R re co rre r = inicioSim ; in t idV a lu e = O ;

whil e ( re co rre r ! = null )

i f ( re co rrer.t ipoTok e n .e qu a l s ( " Id e n t i f ic a do r" )) idV a lu e = (in t ) re co rrer. v a lo r To ke n ;

r e corr e r = re corr e r .s igui e n te;

}

ret urn idV a l u e;

}

)

p a c ka g e S in ta c t ico ;

impor t Est r u c t u r as. Pil a;

impo rt G ra m at ic a. AcomodoG ra m at ic a;

impo rt analizador.lexico.c.ClasificaRebuilt;

public class MatrizPredictiva {

Pila pila; AcomodoGramatica gramatica; ClasificaRebuilt lexico; boolean error = false;

MatrizPredictiva()

pila = new Pila();

gramatica = new AcomodoGramatica();

gramatica.ini(); / / pedimos la gramatica y se trabaja

lexico = new ClasificaRebuilt(); // pedimos el programa a analizar

)

public int matriz(int x, int y)

/• l b e g l e nd l id l : = 1 ; l rea l ( 1 l w ril l in t l

+ 1 - 1 $ 1 •/

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m at rizPredictiva = { { | 1, | o, | o, | o, | o, | o, | o, | o, | o, | o, | o, |
|  | o, | o, | 2, | o, | o, | 2, | o, | o, | 2, | o, | o, |
|  | o, | 4, | 3, | o, | o, | 3, | o, | o, | 3, | o, | o, |
|  | o, | o, | 5, | o, | o, | 6, | o, | o, | 7 , | o, | o, |
| { | o, | o, | e, | o, | o, | o, | o, | o, | o, | o, | o, |
|  | o, | o, | o, | o, | o, | o, | o, | 10, | o, | 9, | o, |
|  | o, | o, | 1 1 , | o, | o, | o, | 1 1 , | o, | o, | o, | 1 1 , |
|  | o, | o, | o, | o, | 12, | o, | o, | 13, | o, | o, | o, |
| { | o, | o, | 14, | o, | o, | o, | 14, | o, | o, | o, | 14, |
|  | o, | o, | o, | o, | 1 6 , | o, | o, | 1 6 , | o, | 1 6 , | o, |
|  | o, | o, | 18 , | o, | o, | o, | 1 7 , | o, | o, | o, | 1 9 , |
|  | o, | o, | o, | o, | o, | o, | o, | o, | o, | o, | o, |

in t [ J [ J

|  |  |  |
| --- | --- | --- |
| o,  o, | o,  o, | o}'  O}' |
| o, | o, | O}' |
| o, | o, | o}' |
| o, | o, | o}' |
| o, | o, | o}' |
| o, | o, | O}' |
| o, | o, | o}' |
| o, | o, | o}' |

15, 15, O}'

o, o, O}'

20 , 2 1, o}'

o, o, O}};

{22 ,

o, o, o, o, o, o, o, o, o, o,

return matrizPredictiva[x][y];

}

public int obtenProduccionMatrizP(String x, String a) int posX = gramatica.indiceNoTerminal(x); // O int posY = gramatica.indiceTerminal(a); // O return matriz(posX, posY);

}

public boolean noEsTerminal(String buscar) {

fer (String buscarEn : gramatica.sim.bolo s N o T e rminales) if (buscarEn != null)

if (buscarEn.equals(buscar))

return true;

}

return false;

}

public void LlDiver() { //a sera el lexema enviado del analizador lexico pila.pus (grarnat ic a.s i boloinicial());

String x = pila.peak(); // tope de la pila

// System.out.println("x: "+x);

String a = lexico.pedirToken(); // pedir la primer palabra

// System.out.println(11 pedir token 1");

// System.out.println("a: "+a);

// System.out.println("inicia pila:");

// pila.imprime();

// Systern. o u t. p rintln("termina pila");

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

while(pila.isEmpty()) { if (noEsTerminal(x)) {

if(obtenProduccionMatrizP(x, a) != O) {

//

a )) ;

izquierda

//

//

//

//

//

//

System.out.println("no-matriz: "+obtenProduccionMatrizP(x,

pila.pop(); //y un ciclo push();

cicloPush(obtenProduccionl1at rizP(x, a)); // derecha a

System.out.println("inicia pila:");

pila.imprime();

System.out.println("termina pila");

System.out.println("despues");

x = pila.peak();

System.out.println("x: "+x);

l e l se {

Sy ste m .o u t. p rintln("error 1"); errorSintactico(a); lexico.irnp rirne T a bl as () ;

b reak;

)

) else {

if(x.equals(a))

pila.pop();

x = pila.peak();

// System.out.println("x: "+x); a = lexico.pedirToken();

// System.out.println(11 pedir token 2");

// System.out.println("a: "+a); System.out.print(a+" ");

)

else if(x.equals("e"))

pila.pop();

// pila.pop();

x = pila.peak();

System.out.println(11vacio x: 11+x);

)

else {

// System.out.println("error 2"); errorSintactico(a); lexico.irnp rirne T a bl as () ;

b reak;

}

}

i f ( e rror == fa lse ) {

i f (! a.e q u a l s ( "$ " ))

// Sy ste m .o u t. p rintln("inicia pila:");

// pila.imprime();

// Systern. o u t. p rintln("termina pila"); System.out.println(11 " ) ; LlDiver();

) else {

System.out.println("A'.l'lá li s i s te rminado"); lexico.irnp rirne T a bl as () ;

}

)

p rivate void cicloPush(int produccion)

String[] deriva = gramatica.produccionDerecha(produccion); produccion a ingresar

int i = deriva.lengt ; while (i != O) {

pila.push(deriva[i-1]);

i=i-1;

// la

)

}

private void errorSintactico(String x) if(!x.equals ( " $ " ) ) {

System.out.println("\nError sintactico en: 11 + x); error = true;

}

}

class test {

public static void main(String[) args) MatrizPredictiva m = new MatrizPredictiva(); m.LlDiver();

//m.obtenProduccionHatrizP(O, 7);

)

}

package Lectura;

public class NoEsDeString { private int numero;

public int cantidadDeCadenas() return nuroe.ro ;

// H ét odo S pl i t

p u blic S t ring [ J no EsEs pli t ( S t ring C a d, S t ring comodin ) {

// c u a n t o s se p a r a do res ex i ste n e n l a c a d e n a num e r o = O;

for ( int i = O; i < Cad .length(); i+ +) {

if (Cad .ch a rAt( i) == comodin .ch a rAt(0)) { num ero+ +;

}

}

St ring [] vectem p = new St ring [num e ro + 1]; int i = O;

whil e (i < nume ro) {

int posicion = Cad.indexOf(comodin );

vectem p[i] = Cad .subst ring(0, pos icion); Cad = Cad .subst ring(pos icion + 1);

*i++;*

}

vectem p[i] = Cad; return (vecternp) ;

//V ic t o r V i e ra B a l a n ta;

}

public String NoEsTrim(String palab ra) {

int *inicio =* O, fin = O;

##### // recorre la palabra hasta encontrar un carcater diferente de espacio for(int i = O, c = palab ra.length(); i < c; i++) {

if(palab ra.c arAt( i) == ' ')

##### inic io+ +;

else

}

break;

for( int i = palabra.length() - 1; i >= O; i--) { if(palabra.charAt(i) == ' ')

fin++ ;

else

}

break;

ret u rn palab ra .subst ring ( inic io, palab ra .length() - fin);

}

}

package analizador.lexico.c;

 Enumerado Palabras Reservadas

### enum PalabrasReservadas { // lista de palabras reservadas a usar READ (200),

ID (201),

WRITE (202),

INT (203),

END (204),

BEGIN (205);

#### \*\*\*\*\*\*\*\*\*\*//

private final int valor; // valor numérico de cada palabra reservada

private PalabrasReservadas (int valor){ // constructor de la clase enum PalabrasReservadas

this.valor = valor;

}

public int getValor(){ // método para conseguir el valor numérico de cada palabrareservada

return valor;

)

public class PalabraReser-va d a {

// m ét odo q u e r e cib e u n st ring ( p a l a br a r ese rv a d a ) y v e ri f ic a s i ex i ste e n e l e n um e r a do

public boolean ExistePalabraReservada(String pReservada)

// ciclo que recorre el enumerado hasta encontrar la palabra recibida fer (PalabrasReservadas pr : PalabrasReservadas.values()) {

if (pr.name().toLowerCase().equals(pReservada.toLowerCase())) return true; // si se encuentra regresa un true

}

return false; // en caso de que no existe se regresa un false

)

// método que devuelve el valor num.e rico de una palabra reservada public int getValorPalabraReservada(String pReservada)

// ciclo que recorre el enUI(l rado hasta encontrar la palabra recibida fer (PalabrasReservadas pr : PalabrasReservadas.values()) {

if (pr.narne ( ) .t o L o w er C ase ( ) .e q u a l s ( pR ese rvada.toLowerCase())) return pr.getValor(); // *si* la encuentra devuelve su valor

numérico

)

return -1; // en caso contrario masna un -1 que seria el caso de error (no encontrado)

)

public void reservadas() {

fer (PalabrasReser-va d as p r : P a l ab rasReser-va d as. v a l u e s ( )) Sy ste m . o u t. prin t ln ( pr . n a m e ( } + 11\t" + pr.getValor(}};

)

)

public String tipoPalabra(String token) { if (ExistePalabraReservada(token))

return II Palabra Re.11 ;

### else

return "Identificador";

)

)

class test {

public stat ic void main(String[ J ar g s ) {

P a l a br a R ese rvada pr = new PalabraReservada();

System.out.println(pr.tipoPalabra("bololean")); System.out.println("Existe: " + pr.ExistePalabraReservada("boolean"}}; System.out.println("Palabra: 11 + pr.getValorPalabraReservada(11boolean11 ) ) ;

}

)

package Estructuras;

public class Pila { Nodo inicio, fin;

public Pila() { inicio = null;

public class Nodo Nodo sig; String dato;

public Nodo(String dato) this.dato = dato;

}

@OVerride

public String toString() {//conversion de posicion de memoria a string return dato.toString();

}

}

public void push(String dato) { Nodo nuevo = new Nodo(dato); push(nuevo);

}

private void push(Nodo dato) if (inicio != null) {

dato.sig = inicio; inicio = dato;

} else

*inicio =* fin = dato;

}

public void pop() { Nodo temp = inicio;

if (inicio != null) { inicio = temp.sig; temp.sig = null;

l else

System.out.println(11 Pila vacia");

}

public String peak() { if (inicio != null) {

return inicio.dato;

} else

return null;

}

public boolean isEmpty() { return inicio != null;

}

public void imprime() { Nodo ternp = *inicio ;*

w hil e ( te mp ! = n u ll ) {

Sy ste rn. o u t. p r in t ln ( te rnp) ; te rnp = te rnp .s ig ;

}

}

cl ass

//

//

//

//

//

//

//

//

//

//

}

}

test {

public static void main(String[) args) { Pila p = new Pila();

p.pus:1( " 1 " ) ;

p . p u s h ( 11 2 11 ) ; p . p u s h ( 11 3 11 ) ; p . p u s h ( 11 4 11 ) ;

#### S y ste m .o u t . prin t ln ( 11 : 11 + p . p eak ()) ;

p . pop () ;

p . imprim e () ;

}

p a c ka g e S in ta c t ico ;

p u blic c l ass tesst {

public static void main(String[] args) { MatrizPredictiva m = new Matr izPredictiva(); m. Ll Dí ve r () ;

)

package Estructuras; public class TestLista {

ListasR al;

Test L is t a () {

al = new ListasR();

}

public void test(String token, String tipo) { al.agregarElementoLSimbolosR(token, tipo, al.buscaRepR(token) + 1,

( i nt ) (t o ke n . c h a r At (O)) , O, n u l l ) ;

}

public void imp ( ) { System.out.println(al.ExistePalabraT("0")); al.mostrarListaSimbolosR();

### System.out.println(al.buscaident("!"));

Sy st em. o ut . pr i nt l n (a l. u lt i moEn Fi l a () ) ;

}

public static void roa in ( S t ring [] a rg s ) { T est L i sta t = n e w T est L i sta () ;

t. test(":=', "Identificador");

t.test(" ", "Siro. Es p ." ) ;

t.test ( " ! " , " Id e n t i f ic a dor " ) ;

t. test ( ": = ", " Sim . Es p ." ) ; t.test ( " # " , " Siro . Es p ." ) ; t.test ( " # " , "Id e n t i f ic a do r"); t.test("!", "Siro. Es p ." ) ;

t . í mp () ;

// c ha r e [ ] =

•

|  |  |  |
| --- | --- | --- |
| //  //  //  // |  | 1 '( 1 *I*'  ' ) ' ,  .' +-.' , |
| //  //  // |  | . ' . ' '  , ' . '' |
| //  //  // |  | ' =' };  fer (char ci : c)  System.out.println( Integer.valueOf(ci) ); |
|  | } |  |
| } |  |  |

package analizador.lexico.c; public class Tipos

public boolean esEspacio(int ascii){

íf(ascíí == 32 11 ascíí == 3 11 ascíí == 9) elsereturrn true;

returrn false;

**)**

public boolearn esCero(int ascii){ íf(ascíí == 48)

elsereturn true;

returrn false;

**)**

public boolean esMayuscula(int ascii){ es mayuscula

íf(ascíí >= 65 && ascíí <= 90)

elsereturn true;

## returrn false;

}

public boolean esMinuscula(int ascii){ minuscula

es

íf(ascíí >= 97 && ascíí <= 122)

## elsereturn true;

returrn false;

}

public boolean esNumero(int ascii){ numero

es

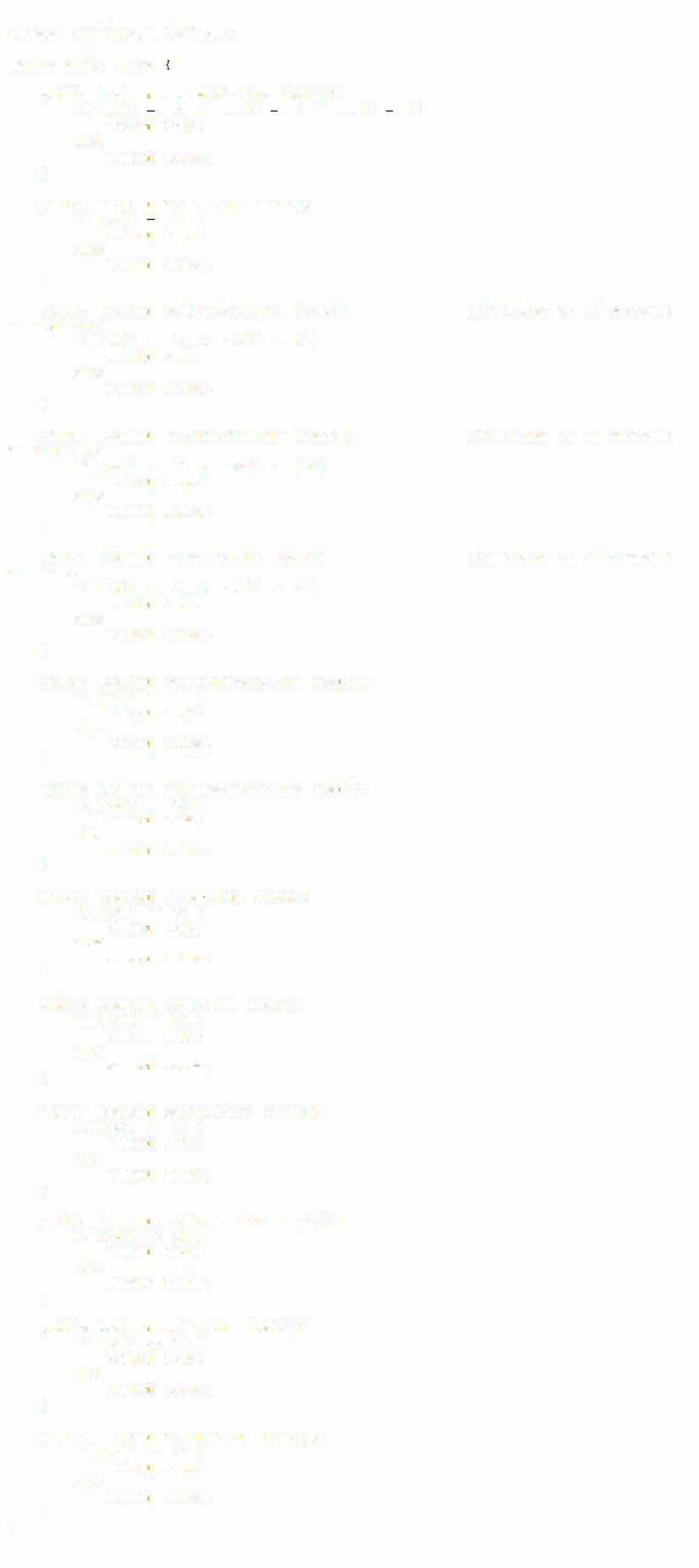
if(ascii >= 49 && ascii <= 57) elsereturn true;

returrn false;

//verifica si el caracter

//verifica si el caracter

//verifica si el caracter

}

public boolean esParentesisl(int ascíi){ íf(ascíí == 40)

elsereturrn true;

returrn false;

}

public boolearn esParentesis2(int ascíi){ íf(ascíí == 41)

return true; else

return false;

}

public boolearn esComa(int ascii){ íf(ascíí == 44)

elsereturn true; returrn false;

}

public boolearn esPyC(int ascii){ íf(ascíí == 59 )

elsereturzn true;

return false;

}

public boolearn esigual(int ascii){ íf(ascíí == 61)

elsereturrn true;

return false;

}

public boolean esDPuntos(int ascii){ íf(ascíí == 58)

elsereturrn true;

retur:n false;

}

public boolean esMas(int ascii){ íf(ascíí == 43)

elsereturrn true; returrn false;

}

public boolearn esMenos(int ascii){ íf(ascíí == 45 )

elsereturn true;

returrn false;

}

}