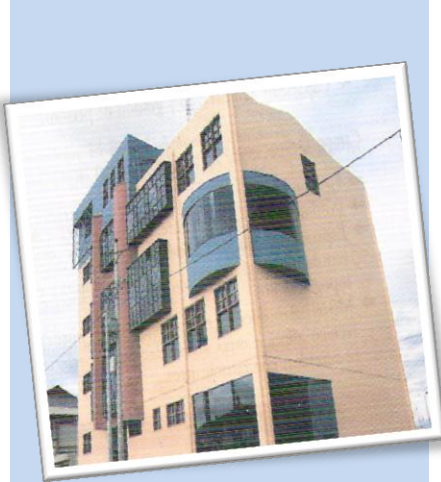


DISEÑO DE LENGUAJE DE PROGRAMACION

COMPILADOR JLEX-CUP



E.F.P

SISTEMAS Y COMPUTACION



DOCENTE:

Ing. Jhon Henry Garcia Ruiz

SEMESTRE: IX

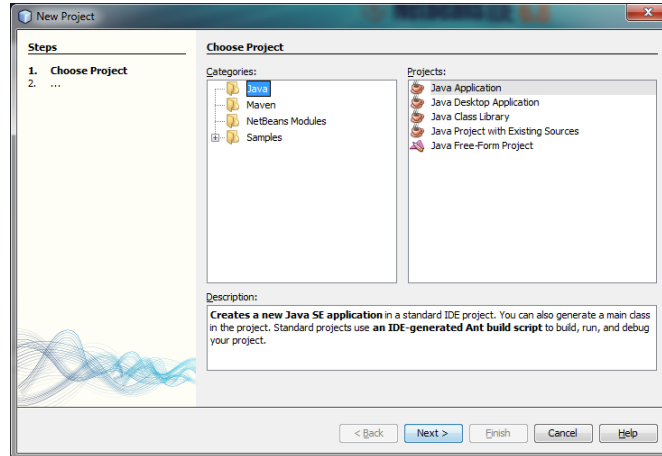
INTEGRANTES:

- BERROSPI ANAYA, ABRAHAM
- CABELLO HIDALGO JULY

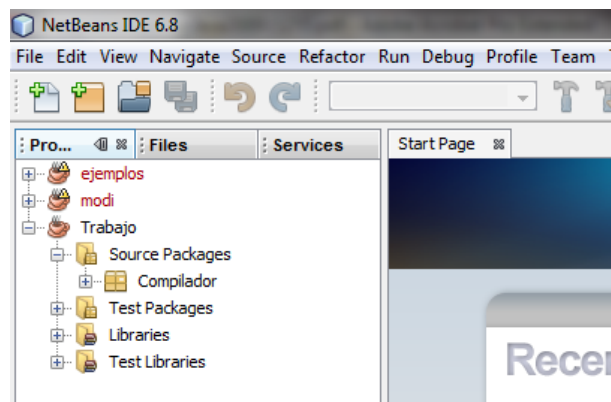
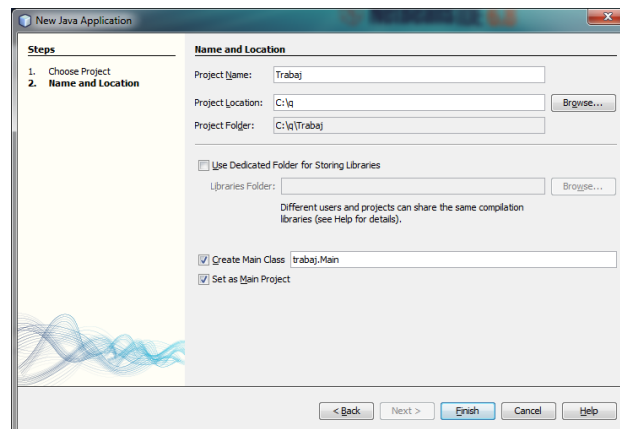
Cerro de Pasco, Junio del 2012

Creación de proyecto en Netbeans

Como paso inicial se procederá a crear un proyecto nuevo en Netbeans.



Luego se procedió a crear el proyecto con el nombre **Trabajo**



Analizador léxico (JLex)

La definición de las reglas para el analizador léxico, se creara en una clase o en un block de notas, el cual se guardara con el nombre que deseamos y con la extensión “*.lex”.

```
1 package Compilador;
2 import java_cup.runtime.Symbol;
3
4 %%
5
6 %cup
7 %public
8 %full
9 %line
10 %char
11 %ignorecase
12 %eofval{
13     return new Symbol(sym.EOF,new String("... Lineas Principales..."));
14 %eofval}
15
16 digito = [0-9]
17 letra = [a-zA-Z]*
18 espacio = \t|\f|" "|\\r|\\n
19
20 %%
21 "#"      {return new Symbol(sym.CAB, yychar, yyline, yytext());}
22 "!"      {return new Symbol(sym.COMIS, yychar, yyline, yytext());}
23 "."      {return new Symbol(sym.PUNTO, yychar, yyline, yytext());}
24 "="      {return new Symbol(sym.IGUAL, yychar, yyline, yytext());}
25 "<>"     {return new Symbol(sym.DIFERENTE, yychar, yyline, yytext());}
26 ">="     {return new Symbol(sym.MAYORIGUAL, yychar, yyline, yytext());}
27 "<="     {return new Symbol(sym.MENORIGUAL, yychar, yyline, yytext());}
28 ">"      {return new Symbol(sym.MAYOR, yychar, yyline, yytext());}
29 "<"      {return new Symbol(sym.MENOR, yychar, yyline, yytext());}
30 "("      {return new Symbol(sym.PARENTI, yychar, yyline, yytext());}
31 ")"      {return new Symbol(sym.PARENTD, yychar, yyline, yytext());}
32 "{"      {return new Symbol(sym.LLAVEI, yychar, yyline, yytext());}
33 "}"      {return new Symbol(sym.LLAVED, yychar, yyline, yytext());}
34 "["      {return new Symbol(sym.MAS, yychar, yyline, yytext());}
35 "-"      {return new Symbol(sym.MENOS, yychar, yyline, yytext());}
36 "*"      {return new Symbol(sym.MULTI, yychar, yyline, yytext());}
37 "/"      {return new Symbol(sym.ENTRE, yychar, yyline, yytext());}
38 "=="     {return new Symbol(sym.MENOSNUM, yychar, yyline, yytext());}
39 "--"     {return new Symbol(sym.MENOSUNO, yychar, yyline, yytext());}
40 "+="     {return new Symbol(sym.MASNUM, yychar, yyline, yytext());}
41 "++"     {return new Symbol(sym.MASUNO, yychar, yyline, yytext());}
42 ";"      {return new Symbol(sym.PYCOMA, yychar, yyline, yytext());}
43 ","      {return new Symbol(sym.COMA, yychar, yyline, yytext());}
44 "entero" {return new Symbol(sym.ENTERO, yychar, yyline, yytext());}
45 "cadena" {return new Symbol(sym.CADENA, yychar, yyline, yytext());}
46 "real"   {return new Symbol(sym.REAL, yychar, yyline, yytext());}
47 "incluir" {return new Symbol(sym.INCLUIR, yychar, yyline, yytext());}
48 "iostream" {return new Symbol(sym.IOSTREAM, yychar, yyline, yytext());}
49 "stdio"  {return new Symbol(sym.STDIO, yychar, yyline, yytext());}
50 "h"      {return new Symbol(sym.H, yychar, yyline, yytext());}
51 "main"   {return new Symbol(sym.MAIN, yychar, yyline, yytext());}
52 "void"   {return new Symbol(sym.VOID, yychar, yyline, yytext());}
53 "escribir" {return new Symbol(sym.ESCRIBIR, yychar, yyline, yytext());}
54 "paq"    {return new Symbol(sym.PAQ, yychar, yyline, yytext());}
55 "ari"     {return new Symbol(sym.ARI, yychar, yyline, yytext());}
56 "sino"    {return new Symbol(sym.SINO, yychar, yyline, yytext());}
57 "entonces" {return new Symbol(sym.ENTONCES, yychar, yyline, yytext());}
58 "kama"    {return new Symbol(sym.KAMA, yychar, yyline, yytext());}
59 "hacer"   {return new Symbol(sym.HACER, yychar, yyline, yytext());}
60 {letra}   {return new Symbol(sym.LETRA, yychar, yyline, yytext());}
61 {digito}+ {return new Symbol(sym.NUMERO, yychar, yyline, new Integer(yytext()));}
62 {espacio} {}
63 .        { System.out.println("Caracter ilegal: " + yytext()); }
```

Implementación de Parser utilizando Cup

Para implementar un Parser en Cup es necesario que se cree un archivo con sintaxis cup, conteniendo las reglas de la gramática en notación BNF para que esta sea reconocida por Cup.

Y se crea el siguiente condigo de la misma manera que la anterior, dicho archivo se guardara con la extensión “*.cup”

```
1 package Compilador;
2 import java.io.*;
3 import java_cup.runtime.*;
4
5 parser code
6 {
7     public static void main(String args[]) throws Exception{
8         //new parser(new Yylex(new FileInputStream(args[0]))).parse();
9         new parser(new Yylex(System.in)).parse();
10    }
11    public void syntax_error(Symbol s){
12        report_error("Error de sintaxis. Linea: " + (s.right + 1) +
13            " Columna: " + s.left + ". Texto: \"" + s.value + "\"", null);
14    }
15 }
16
17 terminal String ARI, ENTONCES, IGUAL, PARENTI, PARENTD, H, PUNTO, PAQ;
18 terminal String CAB, INCLUIR, IOSTREAM, STDIO, MAIN, VOID, ESCRIBIR, COMIS;
19 terminal String MENOSNUM, MENOSUNO, MASNUM, MASUNO, COMA, PYCOMA;
20 terminal String DIFERENTE, MAYORIGUAL, MENORIGUAL, MAYOR, MENOR;
21 terminal String KAMA, HACER, SINO, MAS ,MENOS, MULTI, ENTRE;
22 terminal String ENTERO, CADENA, REAL;
23 terminal String LLAVEI, LLAVED, LETRA;
24 terminal integer NUMERO;
25 non terminal Lista, Sentencia, Expression, inter, inte, Incre;
26 non terminal Regla, Cabecera, Expresionl, ExpresionP;
27 non terminal Vin, Con, Vfi;
28 non terminal Sin, Sinta, Signo, Ins, Instruccion, Var, A, B, Tvar, C, D, E, F, G, HH, I, J, K, L;
29
30 Sin ::= Regla A LLAVEI Tvar Lista LLAVED { : System.out.println("Sentencia analizada correctamente. Sin errores."); : }
31 | Regla A LLAVEI Tvar Ins Lista LLAVED { : System.out.println("Sentencia analizada correctamente. Sin errores."); : }
32 | Regla A LLAVEI Tvar Ins LLAVED { : System.out.println("Sentencia analizada correctamente. Sin errores."); : }
33 | Regla A LLAVEI LLAVED { : System.out.println("Sentencia analizada correctamente. Sin errores."); : }
34 | Regla A LLAVEI Tvar LLAVED { : System.out.println("Sentencia analizada correctamente. Sin errores."); : }
35 | error { : System.out.println("Error sintactico en la sentencia"); : }
36 ;
37 A ::= MAIN PARENTI PARENTD PYCOMA
38 | VOID PARENTI PARENTD PYCOMA
39 ;
40 Tvar ::= Tvar C
41 | C
42 ;
43 C ::= CADENA LETRA PYCOMA
44 | ENTERO LETRA PYCOMA
45 | REAL LETRA PYCOMA
46 | CADENA LETRA D PYCOMA
47 | ENTERO LETRA D PYCOMA
48 | REAL LETRA D PYCOMA
49 ;
50 D ::= D E
51 | E
52 ;
53 E ::= COMA LETRA
54 ;
55 Regla ::= Regla Cabecera
56 | Cabecera
57 ;
58 Cabecera ::= CAB INCLUIR MENOR Expresionl PUNTO H MAYOR { : System.out.println("Sentencia analizada correctamente. Sin ex:
59 | error { : System.out.println("Error sintactico en la sentencia"); : }
60 ;
61 Expresionl ::= IOSTREAM
62 | STDIO
63 ;
64 Lista ::= Lista Sentencia
```

```

98      ;
99      J := J K
100      | K
101      ;
102      K := ARI PARENTI Expression PARENTD ENTONCES LLAVEI J LLAVED
103      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI LLAVED
104      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI Ins LLAVED
105      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI J Ins LLAVED
106      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI Ins J LLAVED
107      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI LLAVED SINO LLAVEI LLAVED
108      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI Ins LLAVED SINO LLAVEI Ins LLAVED
109      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI Ins LLAVED SINO LLAVEI J Ins LLAVED
110      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI Ins LLAVED SINO LLAVEI Ins J LLAVED
111      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI Ins LLAVED SINO LLAVEI J LLAVED
112      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI J LLAVED SINO LLAVEI Ins LLAVED
113      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI Ins LLAVED SINO LLAVEI LLAVED
114      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI LLAVED SINO LLAVEI Ins LLAVED
115      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI LLAVED SINO LLAVEI J Ins LLAVED
116      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI LLAVED SINO LLAVEI Ins J LLAVED
117      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI LLAVED SINO LLAVEI J LLAVED
118      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI J LLAVED SINO LLAVEI LLAVED
119      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI J Ins LLAVED SINO LLAVEI LLAVED
120      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI J Ins LLAVED SINO LLAVEI Ins LLAVED
121      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI J Ins LLAVED SINO LLAVEI J Ins LLAVED
122      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI J Ins LLAVED SINO LLAVEI Ins J LLAVED
123      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI Ins J LLAVED SINO LLAVEI LLAVED
124      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI Ins J LLAVED SINO LLAVEI Ins LLAVED
125      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI Ins J LLAVED SINO LLAVEI J Ins LLAVED
126      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI Ins J LLAVED SINO LLAVEI Ins J LLAVED
127      | ARI PARENTI Expression PARENTD ENTONCES LLAVEI Ins J LLAVED SINO LLAVEI J LLAVED
128      | KAMA PARENTI Expression PARENTD HACER LLAVEI Ins LLAVED
129      | KAMA PARENTI Expression PARENTD HACER LLAVEI J Ins LLAVED
129      | KAMA PARENTI Expression PARENTD HACER LLAVEI J Ins LLAVED
130      | KAMA PARENTI Expression PARENTD HACER LLAVEI Ins J LLAVED
131      | KAMA PARENTI Expression PARENTD HACER LLAVEI LLAVED
132      | KAMA PARENTI Expression PARENTD HACER J LLAVEI J LLAVED
133      | KAMA PARENTI Expression PARENTD HACER J LLAVEI LLAVED
134      | KAMA PARENTI Expression PARENTD HACER LLAVEI J LLAVED
135      | PAQ PARENTI ExpressionP PARENTD LLAVEI Ins LLAVED
136      | PAQ PARENTI ExpressionP PARENTD LLAVEI J Ins LLAVED
137      | PAQ PARENTI ExpressionP PARENTD LLAVEI Ins J LLAVED
138      | PAQ PARENTI ExpressionP PARENTD LLAVEI LLAVED
139      | PAQ PARENTI ExpressionP PARENTD LLAVEI J LLAVED
140      ;
141
142      Ins := Ins Instruccion
143      | Instruccion
144      ;
145      Instruccion := ESCRIBIR PARENTI COMIS HH COMIS PARENTD PYCOMA
146      | ESCRIBIR PARENTI COMIS COMIS PARENTD PYCOMA
147      | Var PYCOMA
148      ;
149      HH := HH I
150      | I
151      ;
152      I := LETRA
153      | NUMERO
154      ;
155      Var := LETRA IGUAL LETRA
156      | LETRA IGUAL NUMERO
157      | LETRA IGUAL NUMERO Signo NUMERO
158      | LETRA IGUAL NUMERO Signo LETRA
159      | LETRA IGUAL LETRA Signo NUMERO
160      | LETRA IGUAL LETRA Signo LETRA

```

```

161 | LETRA IGUAL NUMERO Signo NUMERO F
162 | LETRA IGUAL NUMERO Signo LETRA F
163 | LETRA IGUAL LETRA Signo NUMERO F
164 | LETRA IGUAL LETRA Signo LETRA F
165 ;
166 F ::= F G
167 | G
168 ;
169 G ::= Signo NUMERO
170 |Signo LETRA
171 ;
172 Expression ::= LETRA inter NUMERO
173 | NUMERO inter NUMERO
174 | LETRA inter LETRA
175 | NUMERO inter LETRA
176 ;
177 inter ::= DIFERENTE
178 | MAYORIGUAL
179 | MENORIGUAL
180 | MAYOR
181 | MENOR
182 | IGUAL
183 ;
184 inte ::= MAYORIGUAL
185 | MENORIGUAL
186 | MAYOR
187 | MENOR
188 ;
189 ExpressionP ::= Vin PYCOMA Con PYCOMA Vfi
190 ;
191 Vin ::= LETRA IGUAL NUMERO
192 ;
193 Con ::= LETRA inte NUMERO
194 ;
195 Vfi ::= LETRA Incre
196 | LETRA IGUAL LETRA MAS NUMERO
197 | LETRA IGUAL LETRA MENOS NUMERO
198 ;
199 Incre ::= MENOSNUM NUMERO
200 | MENOSUNO
201 | MASNUM NUMERO
202 | MASUNO
203 ;
204 Signo ::= MAS
205 |MENOS
206 |ENTRE
207 |MULTI
208 ;

```

Ambos archivos Scanner.java y Parser.java deben ser compilados mediante comandos de consola.

Estos archivos han sido guardados en una carpeta donde se guardó el proyecto dentro de la carpeta **C:\Trabajo\src\Compilado**.

Para lograr la compilación de los archivos será necesario ingresar las siguientes líneas de comando en la terminal de Windows.


```
C:\Windows\system32\cmd.exe

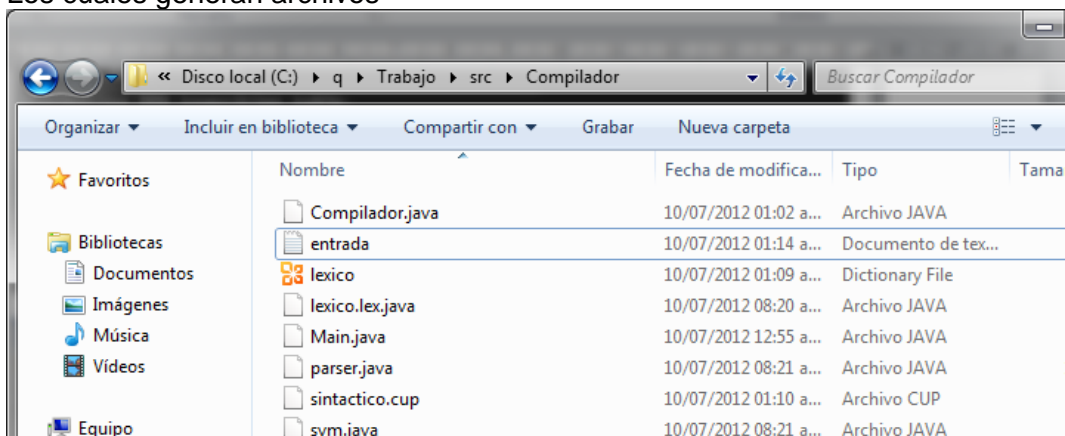
C:\q\Trabajo\src\Compilador>java JLex.Main lexico.lex
Processing first section -- user code.
Processing second section -- JLex declarations.
Processing third section -- lexical rules.
Creating NFA machine representation.
NFA comprised of 224 states.
Working on character classes.....
.....
NFA has 44 distinct character classes.
Creating DFA transition table.
Working on DFA states.....
.....
Minimizing DFA transition table.
102 states after removal of redundant states.
Outputting lexical analyzer code.

C:\q\Trabajo\src\Compilador>
```

```
C:\Windows\system32\cmd.exe

C:\q\Trabajo\src\Compilador>java java_cup.Main sintactico.cup
Opening files...
Parsing specification from standard input...
Checking specification...
Warning: Non terminal "$inta" was declared but never used
Warning: Non terminal "L" was declared but never used
Warning: Non terminal "B" was declared but never used
Building parse tables...
  Computing non-terminal nullability...
  Computing first sets...
  Building state machine...
  Filling in tables...
  Checking for non-reduced productions...
Writing parser...
Closing files...
----- CUP v0.10k Parser Generation Summary -----
0 errors and 3 warnings
43 terminals, 33 non-terminals, and 150 productions declared,
producing 326 unique parse states.
3 terminals declared but not used.
0 non-terminals declared but not used.
0 productions never reduced.
0 conflicts detected (0 expected).
Code written to "parser.java", and "sym.java".
----- (v0.10k)
```

Los cuales generan archivos



Nombre	Fecha de modifica...	Tipo	Tama
Compilador.java	10/07/2012 01:02 a...	Archivo JAVA	
entrada	10/07/2012 01:14 a...	Documento de tex...	
lexico	10/07/2012 01:09 a...	Dictionary File	
lexico.lex.java	10/07/2012 08:20 a...	Archivo JAVA	
Main.java	10/07/2012 12:55 a...	Archivo JAVA	
parser.java	10/07/2012 08:21 a...	Archivo JAVA	
sintactico.cup	10/07/2012 01:10 a...	Archivo CUP	
sym.java	10/07/2012 08:21 a...	Archivo JAVA	

Se cambia de nombre el archivo “lexico.lex.java” por “Yylex.java”

A la clase principal de este proyecto, llamada JLexCup.java le he agregado una instancia a la clase parser, clase en la cual se encuentra el método de carga de archivo de entrada

```
12 public class Compilador {
13     public static void main(String[] args) {
14         try {
15             parser p = new parser(new Yylex(new java.io.FileInputStream("C:\\g\\Trabajo\\src\\Compilador\\entrada.txt")));
16             p.parse();
17         }
18         catch(Exception e) { System.out.println(e.getMessage());}
19     }
20 }
```

Teniendo el archivo de texto Entrada la siguiente información:

```
1 #incluir<iostream.h>
2 main();
3 {
4     cadena re;
5     s=3;
6     ari(a=a) entonces{
7         d=4+3;
8     }
9     paq(a=3;sz<=3;s++){
10         escribir('');
11         escribir(' 2estamos en el programa 23 de3');
12         s=3;
13         d=a;
14     }
15 }
16 }
17 ari(a=a) entonces{
18     d=4+3;
19     escribir('');
20     escribir(' 2estamos en el programa 23 de3');
21     s=3;
```

La salida en la consola de Netbeans, sino se ha programado e implementado de manera correcta la gramática para el reconocimiento de nuestra estructura de ejemplo, deberá de ser la siguiente:

```
Output - modi (run)
Sentencia analizada correctamente. Sin errores.
Sentencia analizada correctamente. Sin errores.
Sentencia analizada correctamente. Sin errores.
Sentencia analizada correctamente. Sin errores.
Sentencia analizada correctamente. Sin errores.
Sentencia analizada correctamente. Sin errores.
Sentencia analizada correctamente. Sin errores.
BUILD SUCCESSFUL (total time: 0 seconds)
```