Manual técnico

Compiladores 1

Práctica 1

Autor: Juan Carlos Rodríguez López

Organización del proyecto

App

Scr

Main

Java

Nootbookmovil

Backend

AnalisiMArcado.java

AnalisisCodigo,java

AnalisisLexico,java

AnalisisLexico,jflex

AnalsisSintacticoTexto.cup

Parser.java

Sym.java

Compilador.java

Navigation

AppScreems

AppNavigation.kt

screems

paginaPrincipal.kt

agregarTarea.kt

MyActivity.kt

Libs

Java-cup.11b.jar

Jflex-full-1.9.1.jar

Gramática analizador léxico

```
package com.compiladores_1.notebookmovil.Backend;

import java_cup.runtime.Symbol;

%%


%init{
   yyline=1;
   yycolumn=1;
%init}

%cup
%class analizadorLexico
%public
%line
%char
%column
%full
%debug

SUMA = \+
INDICE = [1-9]*\.[ ]
NUMERO = [0-9]+
ENCABEZADO = \#{1,6}[ ]
TEXTOCONFORMATO = \*{1,3}
PUNTO = \.
LineTerminator = \r|\n|\r\n
WhiteSpace = [ \t\f]+
TEXTO = [a-zA-Z]+(\s)*
TEXTO_PARRAFO = ([^\#\*\n0-9]+(\n)?)+


%%

<YYINITIAL> {WhiteSpace} { /* Ignorar espacios en blanco */ }
<YYINITIAL> {LineTerminator} { /* Ignorar saltos de línea */ }

<YYINITIAL> {SUMA} { return new Symbol(sym.SUMA, yyline, yycolumn, yytext().trim()); }

<YYINITIAL> {INDICE} { return new Symbol(sym.INDICE, yyline, yycolumn, yytext().trim()); }

<YYINITIAL> {PUNTO} { return new Symbol(sym.PUNTO  , yyline, yycolumn, yytext().trim()); }
<YYINITIAL> {NUMERO} { return new Symbol(sym.NUMERO, yyline, yycolumn, yytext()); }
<YYINITIAL> {ENCABEZADO} { return new Symbol(sym.ENCABEZADO, yyline, yycolumn, yytext()); }
<YYINITIAL> {TEXTOCONFORMATO} { return new Symbol(sym.TEXTOCONFORMATO, yyline, yycolumn, yytext()); }
<YYINITIAL> {TEXTO} { return new Symbol(sym.TEXTO, yyline, yycolumn, yytext()); }

<YYINITIAL> \#{1,6} {
   System.out.println("❌ Error de sintaxis: Se esperaba un espacio después del encabezado en línea " + yyline);
   return new Symbol(sym.error, yyline, yycolumn, yytext());
}

<YYINITIAL> . {
   System.out.println("❌ Token no reconocido: " + yytext() + " en la línea " + yyline + ", columna " + yycolumn);
   return new Symbol(sym.error, yyline, yycolumn, yytext());
}
<YYINITIAL> {TEXTO_PARRAFO} { return new Symbol(sym.TEXTO_PARRAFO, yyline, yycolumn, yytext()); }
```

1. package com.compiladores_1.notebookmovil.Backend;

Este es el paquete donde esta mi clase leer, o analizador lexico.

2. import java_cup.runtime.Symbol;

Importacion de la clase Symbol de Cup para los símbolos necesarios.

3.

```
%init{
    yyline=1;
    yycolumn=1;
 %init
```

Aquí están mis contadores de línea y columna.

4. Importaciones propias de jflex

```
%cup
%class analizadorLexico
%public
%line
%char
%column
%full
%debug
```

5.Expresiones regulares

SUMA =\+ : Reconoce el símbolo de suma, usa \, ya que + algo propio de jflex, este se usa como indice de una lista de datos.

INDICE = [1-9]+\.[ ] : Esta expresión pued aceptar números, seguido de un punto que indica que es un índice de una lista de datos.

NUMERO = [0-9]+: Esta reconoce números.

ENCABEZADO = \#{1,6}[ ] : Esta reconoce encabezados que comienzen con # o hasta ######, y pide un espacio después obligatoriamente para trabajar con encabezador.

TEXTOCONFORMATO = *{1,3} : Este tiene la misma función solo que este haceta ya sea * o hasta ***, para trabajar con formato de texto.

finLinea = \r|\n|\r\n: Este indica sobre un salto de línea, de carro , o carro y salto de líneas.

espaBlanco = [\t\f]+: Esta nos permite tomas los espacion en blanco.

TEXTO = [a-zA-Z]+(\s)*: Esta nos permite, encontrar palabra, con matusculas o minúsculas.

TEXTO_PARRAFO = ([^\#*\n0-9]+(\n)?)+: Esta expresión no permite encontrar párrafos, cuando no hay ni encabezador#, ni números, ni formatos de texto**.

6. Sección de retorno

En esta sección, se defique se hará cuando se encuentra sierto patron, y que se enviara a nuestro archivo cup.

```
<YYINITIAL> {espBalco} { /* Ignorar espacios en blanco */ }
<YYINITIAL> {finLineas} { /* Ignorar saltos de línea */ }
<YYINITIAL> {SUMA} { return new Symbol(sym.SUMA, yyline, yycolumn, yytext().trim()); }
<YYINITIAL> {INDICE} { return new Symbol(sym.INDICE, yyline, yycolumn, yytext().trim()); }
<YYINITIAL> {NUMERO} { return new Symbol(sym.NUMERO, yyline, yycolumn, yytext()); }
<YYINITIAL> {ENCABEZADO} { return new Symbol(sym.ENCABEZADO, yyline, yycolumn, yytext()); }
<YYINITIAL> {TEXTOCONFORMATO} { return new Symbol(sym.TEXTOCONFORMATO, yyline, yycolumn, yytext()); }
<YYINITIAL> {TEXTO} { return new Symbol(sym.TEXTO, yyline, yycolumn, yytext()); }
<YYINITIAL> \#{1,6} {
  System.out.println("❌ Error de sintaxis: Se esperaba un espacio después del encabezado en línea " + yyline);
  return new Symbol(sym.error, yyline, yycolumn, yytext());
```

```
}
<YYINITIAL> . {
   System.out.println("❌ Token no reconocido: " + yytext() + " en la línea " + yyline + ", columna " + yycolumn);
   return new Symbol(sym.error, yyline, yycolumn, yytext());
}
<YYINITIAL> {TEXTO_PARRAFO} { return new Symbol(sym.TEXTO_PARRAFO, yyline, yycolumn, yytext()); }
```

Gramática analizador sintáctico

```
package com.compiladores_1.notebookmovil.Backend;

import java_cup.runtime.Symbol;

action code
{:
   public boolean instruccionValida = false;
   public boolean encabezado= false;
   public boolean lista_numerica = false;
   public boolean parrafo = false;
   public boolean lista = false;
   public boolean tipoFormsto = false;
   public String mensaje="";
   String instruccion="";
:}

   terminal String TEXTO, NUMERO, ENCABEZADO, TEXTOCONFORMATO,INDICE, PUNTO,TEXTO_PARRAFO,SUMA;

   nonterminal INICIO;
   nonterminal INSTRUCCION;
   nonterminal String cadena,elemento1,elementolista,elementolista2;

   start with INICIO;

   INICIO ::= INSTRUCCION{:
                  RESULT = instruccion;
               :}
         ;

INSTRUCCION ::= ENCABEZADO:E cadena:e2
            {:
               instruccion=E+e2;
               System.out.println("✓ Instrucción reconocida: "+instruccion+"\n");
               instruccionValida=true;
               encabezado = true;

            :}
             | ENCABEZADO:E
             {:  System.out.println("❌ Error de sintaxis se esperaba un header");
               mensaje="❌ ERROR DE SINTAXIS";
               instruccion="❌ Header no definido "+E+" ^^^^^^^^^^ ";
               instruccionValida=false;
               encabezado = false;
               RESULT = null;
            :}
             ;

INSTRUCCION ::= elementolista;
INSTRUCCION ::= elementolista2;

INSTRUCCION::=error
         {:
            instruccion="❌ Error de sintaxis es una instrccion imvalida";
              instruccionValida=false;

            RESULT = null;

         :}
          ;

   INSTRUCCION ::= TEXTOCONFORMATO:tf1 cadena:t TEXTOCONFORMATO:tf2
            {:
             if (!tf1.equals(tf2)) {
                mensaje="❌ ERROR DE SINTAXIS";
                instruccion="❌ Cierre no coicide: "+tf1+" "+t+" →"+ tf2+"←, esperado: "+ tf1;
                instruccionValida=false;
                tipoFormsto = false;
                RESULT = null;
             }else{
```

```
                instruccion=tf1+t+tf2;
                System.out.println("✓ Instrucción reconocida: "+instruccion+"\n");
                instruccionValida=true;
                tipoFormsto = true;
                RESULT = instruccion;
              }

             :}
              | TEXTOCONFORMATO:tf1 cadena:t  {:
                  mensaje="❌ ERROR DE SINTAXIS";
                  instruccion="❌ Cierre no presente: "+tf1+" "+t+" →^^^←"+", esperado: →"+ tf1;
                  instruccionValida=false;
                  tipoFormsto = false;
                  RESULT = null;
             :}
             | cadena:t TEXTOCONFORMATO:tf1  {:
               mensaje="❌ ERROR DE SINTAXIS";
               instruccion="❌ Inicio no presente: "+" →^^^←"+" "+t+tf1+", esperado: →"+ tf1;
              instruccionValida=false;
              tipoFormsto = false;
               RESULT = null;
            :}
            |  TEXTOCONFORMATO:tf1 error {:
                  mensaje="❌ ERROR DE SINTAXIS";
                  instruccion="❌ Texto no definido, Cierre no definido: "+tf1+" →^^^^^^^^^^^^^^←  →^^^←, cierre esperado: "+tf1;
                  instruccionValida=false;
                  tipoFormsto = false;
                  RESULT = null;
             :}
             ;

 INSTRUCCION ::= TEXTO_PARRAFO:p
             {:
                instruccion = p;
                System.out.println("✓ Instrucción reconocida: " + instruccion + "\n");
                 instruccionValida=true;
                  parrafo = true;
                RESULT = instruccion;
            :}
            |TEXTO:t
              {:
                instruccion+= t;
                System.out.println("✓ Instrucción reconocida: " + instruccion + "\n");
                 instruccionValida=true;
                 parrafo = true;
                RESULT = instruccion;
             :}
;


elementolista ::= INDICE:i cadena:c
            {:
               instruccion = i + c+" ";
               System.out.println("✓ Instrucción reconocida: " + instruccion + "\n");
               instruccionValida=true;
               lista_numerica =true;
               RESULT = instruccion;
            :}
            |INDICE:i cadena:c elementolista:ed
             {:
                instruccion = i + c +" "+ ed;
                System.out.println("✓ Instrucción reconocida: " + instruccion + "\n");
                instruccionValida=true;
                lista_numerica =true;
                RESULT = instruccion;
            :}
            |INDICE:i1 INDICE:i2
             {:
                mensaje="❌ ERROR DE SINTAXIS";
                instruccion="❌ Elemento de lista no definido: "+i1+" ^^^^^^^^^^ "+i2+"^^^^^^^^^^";
                instruccionValida=false;
                lista_numerica =false;
                RESULT = instruccion;
             :}
             |INDICE:i
             {:
                mensaje="❌ ERROR DE SINTAXIS";
                instruccion="❌ Elemento de lista no definido: "+i+" ^^^^^^^^^^";
```

```
                instruccionValida=false;
                lista_numerica =false;
                RESULT = instruccion;
            :}
            ;

elementolista2 ::= SUMA:i cadena:c
            {:
               instruccion = i + c+" ";
               System.out.println("✓ Instrucción reconocida: " + instruccion + "\n");
               instruccionValida=true;
               lista_numerica =true;
               RESULT = instruccion;
            :}
            |SUMA:i cadena:c elementolista2:ed
             {:
                instruccion = i + c +" "+ ed;
                System.out.println("✓ Instrucción reconocida: " + instruccion + "\n");
                instruccionValida=true;
                lista_numerica =true;
                RESULT = instruccion;
            :}
            |SUMA:i1 SUMA:i2
             {:
                mensaje="❌ ERROR DE SINTAXIS";
                instruccion="❌ Elemento de lista no definido: "+i1+" ^^^^^^^^^^ "+i2+"^^^^^^^^^^";
                instruccionValida=false;
                lista_numerica =false;
                RESULT = instruccion;
            :}
             |SUMA:i
             {:
                mensaje="❌ ERROR DE SINTAXIS";
                instruccion="❌ Elemento de lista no definido: "+i+" ^^^^^^^^^^";
                instruccionValida=false;
                lista_numerica =false;
                RESULT = instruccion;
            :}
            ;

cadena ::= elemento1:e1    {: RESULT = e1; :}
      | elemento1:e1 cadena:c    {: RESULT = e1+" "+c; :}
      ;

 elemento1 ::=TEXTO:t      {: RESULT = t; :}
        |NUMERO:n      {: RESULT = n; :}
        |TEXTO_PARRAFO:tp  {: RESULT = tp;:}
        ;
```

1. package com.compiladores_1.notebookmovil.Backend;

Este es el paquete donde esta mi analizado sintáctico.

2. import java_cup.runtime.Symbol;

importación de la clase Symbol de cup.

3. Action code

```
action code
{:
   public boolean instruccionValida = false;
   public boolean encabezado= false;
   public boolean lista_numerica = false;
   public boolean parrafo = false;
   public boolean lista = false;
   public boolean tipoFormsto = false;
   public String mensaje="";
   String instruccion="";
:}
```

Aquí defini variables que me ayudan como bandernes, y me informa si algo salió bien o mal.

4. Términos terminales

terminal String TEXTO, NUMERO, ENCABEZADO, TEXTOCONFORMATO,INDICE, PUNTO,TEXTO_PARRAFO,SUMA;

Aquí defino los tokens que defini en mi archivo .lex.

5. Términos no terminales

```
nonterminal INICIO;
nonterminal INSTRUCCION;
nonterminal String cadena,elemento1,elementolista,elementolista2;
```

Aquí defino los termino o variables por así decirlo que usar en el archivo cup. Exclusivamente en el cup.

6. start with INICIO;

Aquí establecemos quien inicia.

```
INICIO ::= INSTRUCCION{:
                RESULT = instrucción;
             :}
          ;
```

Aquí como el programa es de una instrucción por compilación, propuse que inicio solo tendrá como inico caso instrucción.

7. Gramática de encabezado.

```
INSTRUCCION ::= ENCABEZADO:E cadena:e2
          {:
            instruccion=E+e2;
            System.out.println("✓ Instrucción reconocida: "+instruccion+"\n");
            instruccionValida=true;
            encabezado = true;

          :}
           | ENCABEZADO:E
           {:  System.out.println("❌ Error de sintaxis se esperaba un header");
            mensaje="❌ ERROR DE SINTAXIS";
            instruccion="❌ Header no definido "+E+" ^^^^^^^^^^ ";
            instruccionValida=false;
            encabezado = false;
            RESULT = null;
          :}
           ;

cadena ::= elemento1:e1    {: RESULT = e1; :}
      | elemento1:e1 cadena:c    {: RESULT = e1+" "+c; :}
      ;

 elemento1 ::=TEXTO:t      {: RESULT = t; :}
         |NUMERO:n     {: RESULT = n; :}
         |TEXTO_PARRAFO:tp {: RESULT = tp;:}
         ;
```

8. Gramática de listas.

```
INSTRUCCION ::= elementolista;
INSTRUCCION ::= elementolista2;

elementolista ::= INDICE:i cadena:c
            {:
              instruccion = i + c+" ";
              System.out.println("✓ Instrucción reconocida: " + instruccion + "\n");
              instruccionValida=true;
              lista_numerica =true;
              RESULT = instruccion;
            :}
            |INDICE:i cadena:c elementolista:ed
             {:
               instruccion = i + c +" "+ ed;
               System.out.println("✓ Instrucción reconocida: " + instruccion + "\n");
               instruccionValida=true;
               lista_numerica =true;
               RESULT = instruccion;
             :}
             |INDICE:i1 INDICE:i2
              {:
                 mensaje="❌ ERROR DE SINTAXIS";
```

```
                    instruccion="❌ Elemento de lista no definido: "+i1+" ^^^^^^^^^^ "+i2+"^^^^^^^^^^";
                    instruccionValida=false;
                    lista_numerica =false;
                    RESULT = instruccion;
                :}
                |INDICE:i
                {:
                    mensaje="❌ ERROR DE SINTAXIS";
                    instruccion="❌ Elemento de lista no definido: "+i+" ^^^^^^^^^^";
                    instruccionValida=false;
                    lista_numerica =false;
                    RESULT = instruccion;
                :}
                ;

elementolista2 ::= SUMA:i cadena:c
            {:
                instruccion = i + c+" ";
                System.out.println("✓ Instrucción reconocida: " + instruccion + "\n");
                instruccionValida=true;
                lista_numerica =true;
                RESULT = instruccion;
            :}
            |SUMA:i cadena:c elementolista2:ed
            {:
                instruccion = i + c +" "+ ed;
                System.out.println("✓ Instrucción reconocida: " + instruccion + "\n");
                instruccionValida=true;
                lista_numerica =true;
                RESULT = instruccion;
            :}
            |SUMA:i1 SUMA:i2
            {:
                mensaje="❌ ERROR DE SINTAXIS";
                instruccion="❌ Elemento de lista no definido: "+i1+" ^^^^^^^^^^ "+i2+"^^^^^^^^^^";
                instruccionValida=false;
                lista_numerica =false;
                RESULT = instruccion;
            :}
            |SUMA:i
            {:
                mensaje="❌ ERROR DE SINTAXIS";
                instruccion="❌ Elemento de lista no definido: "+i+" ^^^^^^^^^^";
                instruccionValida=false;
                lista_numerica =false;
                RESULT = instruccion;
            :}
            ;
```

9. Gramática de texto con formato.

```
INSTRUCCION ::= TEXTOCONFORMATO:tf1 cadena:t TEXTOCONFORMATO:tf2
        {:
        if (!tf1.equals(tf2)) {
            mensaje="❌ ERROR DE SINTAXIS";
            instruccion="❌ Cierre no coicide: "+tf1+" "+t+" →"+ tf2+"←, esperado: "+ tf1;
            instruccionValida=false;
            tipoFormsto = false;
            RESULT = null;
        }else{
            instruccion=tf1+t+tf2;
            System.out.println("✓ Instrucción reconocida: "+instruccion+"\n");
            instruccionValida=true;
            tipoFormsto = true;
            RESULT = instruccion;
        }

        :}
        | TEXTOCONFORMATO:tf1 cadena:t  {:
            mensaje="❌ ERROR DE SINTAXIS";
            instruccion="❌ Cierre no presente: "+tf1+" "+t+" →^^^←"+", esperado: →"+ tf1;
            instruccionValida=false;
            tipoFormsto = false;
            RESULT = null;
        :}
        | cadena:t TEXTOCONFORMATO:tf1 {:
          mensaje="❌ ERROR DE SINTAXIS";
```

```
                instruccion="❌ Inicio no presente: "+" →^^^←"+" "+t+tf1+", esperado: →"+ tf1;
                instruccionValida=false;
                tipoFormsto = false;
                RESULT = null;
            :}
            | TEXTOCONFORMATO:tf1 error {:
                mensaje="❌ ERROR DE SINTAXIS";
                instruccion="❌ Texto no definido, Cierre no definido: "+tf1+" →^^^^^^^^^^^^^^← →^^^←, cierre esperado: "+tf1;
                instruccionValida=false;
                tipoFormsto = false;
                RESULT = null;
            :}
            ;
```

10. Gramática de párrafo

```
INSTRUCCION ::= TEXTO_PARRAFO:p
            {:
              instruccion = p;
              System.out.println("✓ Instrucción reconocida: " + instruccion + "\n");
               instruccionValida=true;
                parrafo = true;
              RESULT = instruccion;
            :}
            |TEXTO:t
              {:
                instruccion+= t;
                System.out.println("✓ Instrucción reconocida: " + instruccion + "\n");
                 instruccionValida=true;
                 parrafo = true;
              RESULT = instruccion;
             :}
;
```

11. Gramática de error

```
INSTRUCCION::=error
        {:
          instruccion="❌ Error de sintaxis es una instrccion imvalida";
            instruccionValida=false;

          RESULT = null;
```