

Curso: Compiladores

Ing: Miguel Alejandro Catalán López

Nombre: Manfreed Oswaldo Hernández Mendizábal

Carnet:7590-20-13558

Sección: A

Documentacion detallada del mini-compilador

Gramática Utilizada en el Proyecto

Analizador Léxico (patrones y lexemas)

Un analizador léxico  es la primera fase de un [compilador](https://es.wikipedia.org/wiki/Compilador), consistente en un [programa](https://es.wikipedia.org/wiki/Programa_(inform%C3%A1tica)) que recibe como entrada el [código fuente](https://es.wikipedia.org/wiki/C%C3%B3digo_fuente) de otro programa (secuencia de caracteres) y produce una salida compuesta de [tokens](https://es.wikipedia.org/wiki/Token_(programaci%C3%B3n)) (componentes léxicos) o símbolos. Estos tokens sirven para una posterior etapa del proceso de traducción, siendo la entrada para el [analizador sintáctico](https://es.wikipedia.org/wiki/Analizador_sint%C3%A1ctico) (en inglés parser).

La especificación de un [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) a menudo incluye un conjunto de reglas que definen el [léxico](https://es.wikipedia.org/wiki/L%C3%A9xico). Estas reglas consisten comúnmente en [expresiones regulares](https://es.wikipedia.org/wiki/Expresi%C3%B3n_regular) que indican el conjunto de posibles secuencias de caracteres que definen un token o [lexema](https://es.wikipedia.org/wiki/Lexema).

En algunos lenguajes de programación es necesario establecer patrones para caracteres especiales (como el espacio en blanco) que la gramática pueda reconocer sin que constituya un token en sí.

Token

Un token léxico o simplemente token es una [cadena](https://es.wikipedia.org/wiki/Cadena_de_caracteres) con un significado asignado y, por lo tanto, identificado. Está estructurado como un par que consta de un «nombre de token» y un «valor de token» opcional. El nombre del token es una categoría de unidad léxica.​ Los nombres de token comunes son

[identificador](https://es.wikipedia.org/w/index.php?title=Identificador_(inform%C3%A1tica)&action=edit&redlink=1): nombres que elige el programador;

[palabra clave](https://es.wikipedia.org/wiki/Palabra_reservada): nombres que ya están en el lenguaje de programación;

[separador](https://es.wikipedia.org/w/index.php?title=Delimitador&action=edit&redlink=1) (también conocidos como signos de puntuación): caracteres de puntuación y delimitadores emparejados;

[operador](https://es.wikipedia.org/wiki/Operador): símbolos que operan sobre argumentos y producen resultados;

[literal](https://es.wikipedia.org/w/index.php?title=Literal_(inform%C3%A1tica)&action=edit&redlink=1): literales numéricos, lógicos, textuales, de referencia;

[comentario](https://es.wikipedia.org/wiki/Comentario_(inform%C3%A1tica)): línea, bloque (Depende del compilador si el compilador implementa comentarios como tokens, de lo contrario, se eliminará).

|  |  |
| --- | --- |
| Ejemplos de valores de token | |
| Nombre simbólico | Ejemplos de valores de token |
| identificador | x, color, ARRIBA |
| palabra clave | si, mientras, retorno |
| separador | }, (, ; |
| operador | +, <, = |
| literal | verdadero, 6,02e23, "música" |
| comentario | /\* Recupera datos de usuario \*/, // debe ser negativo |
|  |  |

Patrones y Lexemas del proyecto

Patrón y Lexema .cup utilizados en el proyecto

package codigo;

import java\_cup.runtime.Symbol;

%%

%class LexerCup

%type java\_cup.runtime.Symbol

%cup

%full

%line

%char

L=[a-zA-Z\_]+

D=[0-9]+

espacio=[ ,\t,\r,\n]+

%{

private Symbol symbol (int type, Object value){

return new Symbol (type, yyline,yycolumn,value);

}

private Symbol symbol (int type){

return new Symbol (type, yyline,yycolumn);

}

%}

%%

BYTE {return new Symbol(sym.BYTE, yychar, yyline,yytext());}

SHORT {return new Symbol(sym.SHORT, yychar, yyline,yytext());}

INT {return new Symbol(sym.INT, yychar, yyline,yytext());}

LONG {return new Symbol(sym.LONG, yychar, yyline,yytext());}

FLOAT {return new Symbol(sym.FLOAT, yychar, yyline,yytext());}

DOUBLE {return new Symbol(sym.DOUBLE, yychar, yyline,yytext());}

BOOLEAN {return new Symbol(sym.BOOLEAN, yychar, yyline,yytext());}

CHAR {return new Symbol(sym.CHAR, yychar, yyline,yytext());}

VERDADERO {return new Symbol(sym.VERDADERO, yychar, yyline,yytext());}

FALSO {return new Symbol(sym.FALSO, yychar, yyline,yytext());}

STRING {return new Symbol(sym.STRING, yychar, yyline,yytext());}

RETURN {return new Symbol(sym.RETURN, yychar, yyline,yytext());}

include {return new Symbol(sym.include, yychar, yyline,yytext());}

using {return new Symbol(sym.using, yychar, yyline,yytext());}

namespace {return new Symbol(sym.namespace, yychar, yyline,yytext());}

class {return new Symbol(sym.classT, yychar, yyline,yytext());}

public {return new Symbol(sym.publicT, yychar, yyline,yytext());}

double {return new Symbol(sym.doubleT, yychar, yyline,yytext());}

char {return new Symbol(sym.charT, yychar, yyline,yytext());}

switch {return new Symbol(sym.switchT, yychar, yyline,yytext());}

case {return new Symbol(sym.caseT, yychar, yyline,yytext());}

return {return new Symbol(sym.returnT, yychar, yyline,yytext());}

default {return new Symbol(sym.defaultT, yychar, yyline,yytext());}

endl {return new Symbol(sym.endlT, yychar, yyline,yytext());}

pragma {return new Symbol(sym.pragmaT, yychar, yyline,yytext());}

once {return new Symbol(sym.onceT, yychar, yyline,yytext());}

while {return new Symbol(sym.whileT, yychar, yyline,yytext());}

continue {return new Symbol(sym.continueT, yychar, yyline,yytext());}

if {return new Symbol(sym.ifT, yychar, yyline,yytext());}

else {return new Symbol(sym.elseT, yychar, yyline,yytext());}

true {return new Symbol(sym.trueT, yychar, yyline,yytext());}

false {return new Symbol(sym.falseT, yychar, yyline,yytext());}

"11/05/2024" {return new Symbol(sym.DAY, yychar, yyline,yytext());}

("=") {return new Symbol(sym.igual, yychar, yyline,yytext());}

{espacio} {/\*Ignore\*/}

"//".\* {/\*Ignore\*/}

( ";" ) {return new Symbol(sym.puntoComa, yychar, yyline, yytext());}

( "{" ) {return new Symbol(sym.llaveApertura, yychar, yyline, yytext());}

( "}" ) {return new Symbol(sym.llaveCierre, yychar, yyline, yytext());}

( ">" ) {return new Symbol(sym.signoMenorQue, yychar, yyline, yytext());}

( "<" ) {return new Symbol(sym.signoMayorQue, yychar, yyline, yytext());}

( ":" ) {return new Symbol(sym.signoDosPuntos, yychar, yyline, yytext());}

( "#" ) {return new Symbol(sym.signoAlmohoadilla, yychar, yyline, yytext());}

( "," ) {return new Symbol(sym.signoComa, yychar, yyline, yytext());}

( "+" ) {return new Symbol(sym.signoMas, yychar, yyline, yytext());}

( "-" ) {return new Symbol(sym.signoMenos, yychar, yyline, yytext());}

( "\*" ) {return new Symbol(sym.signoMultiplicacion, yychar, yyline, yytext());}

( "/" ) {return new Symbol(sym.signoSlash, yychar, yyline, yytext());}

( "(" ) {return new Symbol(sym.parentesisApertura, yychar, yyline, yytext());}

( ")" ) {return new Symbol(sym.parentesisCierre, yychar, yyline, yytext());}

( "." ) {return new Symbol(sym.signoPunto, yychar, yyline, yytext());}

( "|" ) {return new Symbol(sym.signoOr, yychar, yyline, yytext());}

{L}({L}|{D})\* {return new Symbol(sym.Identificador, yychar, yyline, yytext());}

("(-"{D}+")")|{D}+ {return new Symbol(sym.Numero, yychar, yyline, yytext());}

. {return new Symbol(sym.ERROR, yychar, yyline, yytext());}

Se presentan los lexemas que puede reconocer el mini-compilador, el patrón que toma los lexemas o token para poder formar la palabra clave.

Patrón y Lexema .flex utilizados en el proyecto

package codigo;

import static codigo.Tokens.\*;

%%

%class Lexer

%type Tokens

L=[a-zA-Z\_]+

D=[0-9]+

espacio=[ ,\t,\r]+

%{

public String lexeme;

%}

%%

BYTE {lexeme=yytext(); return BYTE;}

SHORT {lexeme=yytext(); return SHORT;}

INT {lexeme=yytext(); return INT;}

LONG {lexeme=yytext(); return LONG;}

FLOAT {lexeme=yytext(); return FLOAT;}

DOUBLE {lexeme=yytext(); return DOUBLE;}

BOOLEAN {lexeme=yytext(); return BOOLEAN;}

CHAR {lexeme=yytext(); return CHAR;}

VERDADERO {lexeme=yytext(); return VERDADERO;}

FALSO {lexeme=yytext(); return FALSO;}

STRING {lexeme=yytext(); return STRING;}

RETURN {lexeme=yytext(); return RETURN;}

include {lexeme=yytext(); return include;}

using {lexeme=yytext(); return using;}

namespace {lexeme=yytext(); return namespace;}

class {lexeme=yytext(); return classT;}

public {lexeme=yytext(); return publicT;}

double {lexeme=yytext(); return doubleT;}

char {lexeme=yytext(); return charT;}

switch {lexeme=yytext(); return switchT;}

case {lexeme=yytext(); return caseT;}

return {lexeme=yytext(); return returnT;}

default {lexeme=yytext(); return defaultT;}

endl {lexeme=yytext(); return endlT;}

pragma {lexeme=yytext(); return pragmaT ;}

once {lexeme=yytext(); return onceT ;}

while {lexeme=yytext(); return whileT;}

continue {lexeme=yytext(); return continueT;}

if {lexeme=yytext(); return ifT;}

else {lexeme=yytext(); return elseT;}

true {lexeme=yytext(); return trueT ;}

false {lexeme=yytext(); return falseT ;}

"11/05/2024" {lexeme=yytext(); return DAY ;}

{espacio} {/\*Ignore\*/}

"//".\* {/\*Ignore\*/}

"\n" {return linea;}

";" {lexeme=yytext(); return puntoComa;}

("=") {lexeme=yytext(); return igual;}

("{") {lexeme=yytext(); return llaveApertura;}

("}") {lexeme=yytext(); return llaveCierre;}

("<") {lexeme=yytext(); return signoMenorQue;}

(">") {lexeme=yytext(); return signoMayorQue;}

(":") {lexeme=yytext(); return signoDosPuntos;}

("#") {lexeme=yytext(); return signoAlmohoadilla;}

(",") {lexeme=yytext(); return signoComa;}

("+") {lexeme=yytext(); return signoMas;}

("-") {lexeme=yytext(); return signoMenos;}

("\*") {lexeme=yytext(); return signoMultiplicacion;}

("/") {lexeme=yytext(); return signoSlash;}

("(") {lexeme=yytext(); return parentesisApertura;}

(")") {lexeme=yytext(); return parentesisCierre;}

(".") {lexeme=yytext(); return signoPunto;}

("|") {lexeme=yytext(); return signoOr;}

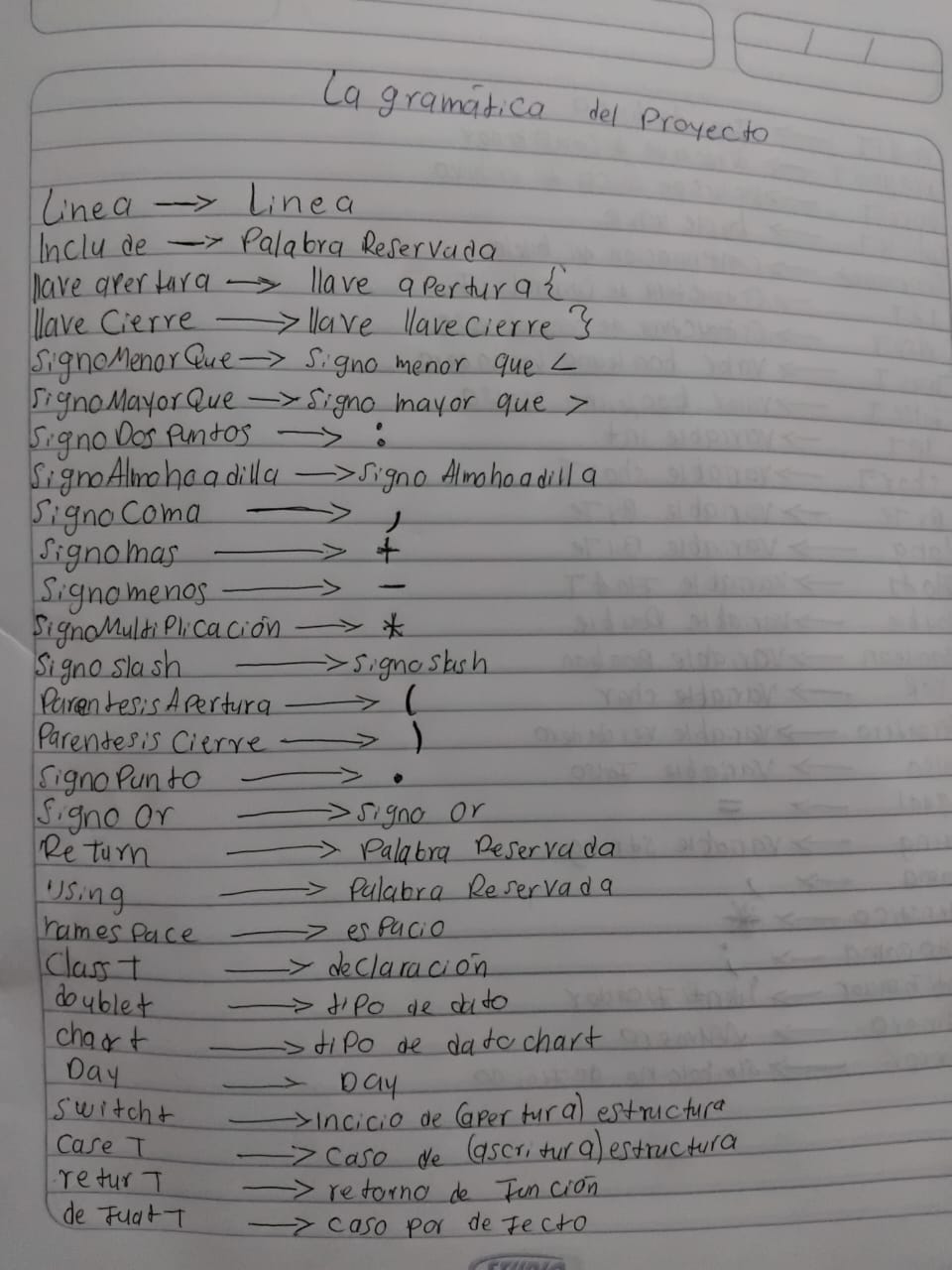
{L}({L}|{D})\* {lexeme=yytext(); return Identificador;}

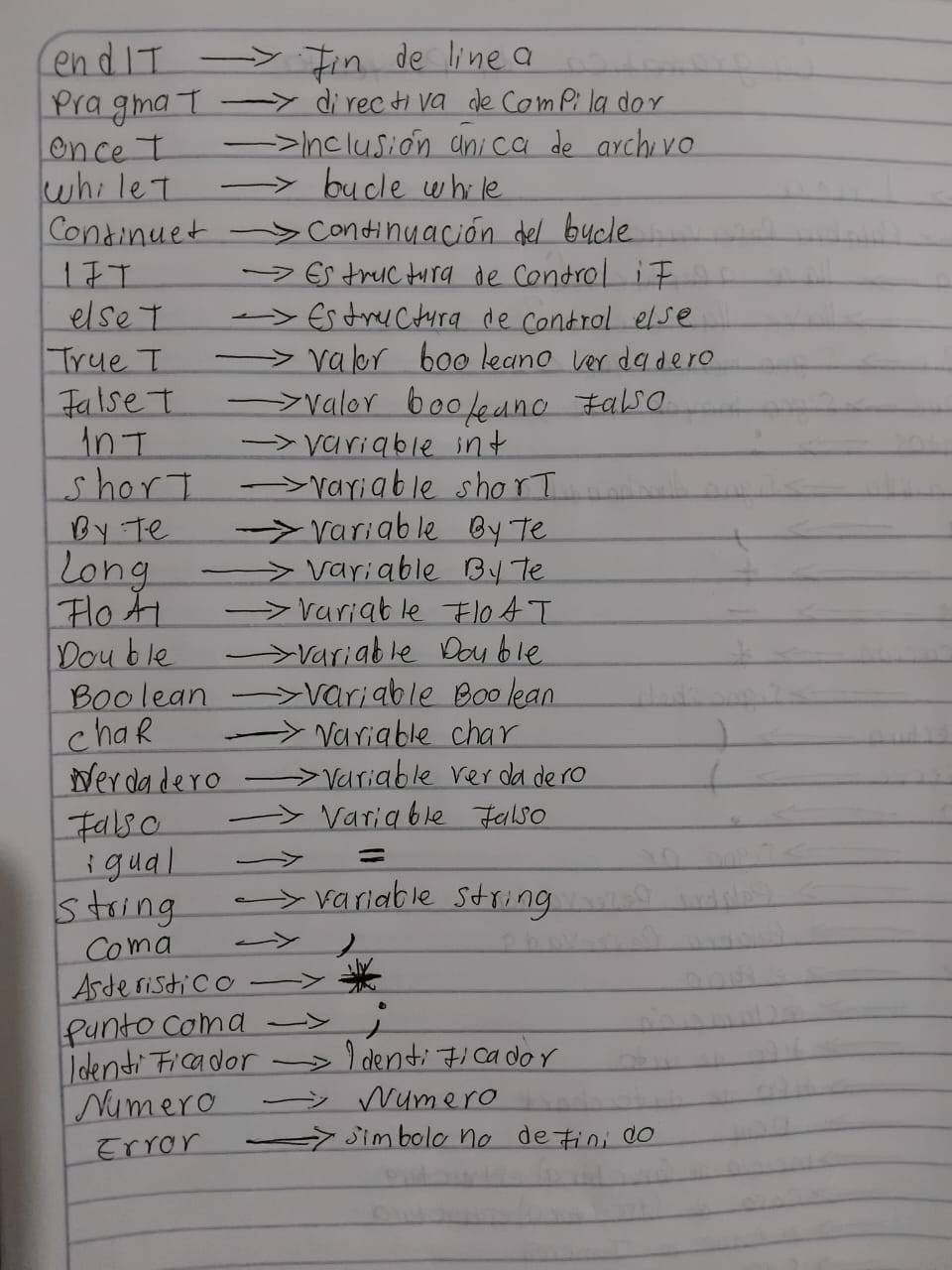
("(-"{D}+")")|{D}+ {lexeme=yytext(); return Numero;}

. {return ERROR;}

Se presentan los lexemas que puede reconocer el documento de .flex del mini-compilador, el patrón que toma los lexemas o token para poder formar la palabra clave.

Gramática Utilizada en el proyecto Analizador Léxico.





La sintaxis del analizador Léxico

package codigo;

import java.io.File;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

public class Principal {

public static void main(String[] args) throws Exception {

String ruta1 = "src/codigo/Lexer.flex";

String ruta2 = "src/codigo/LexerCup.flex";

String[] rutaS={"-parser","Sintax","src/codigo/Sintax.cup"};

generar(ruta1, ruta2,rutaS);

}

//crearea mi analizador lexico

public static void generar(String ruta1, String ruta2, String[] rutaS) throws IOException, Exception {

File archivo;

archivo= new File(ruta1);

JFlex.Main.generate(archivo);

archivo= new File(ruta2);

JFlex.Main.generate(archivo);

java\_cup.Main.main(rutaS);

Path rutaSym = Paths.get("src/codigo/sym.java");

if (Files.exists(rutaSym)) {

Files.delete(rutaSym);

}

Files.move(

Paths.get("sym.java"),

Paths.get("src/codigo/sym.java")

);

Path rutaSin = Paths.get("src/codigo/Sintax.java");

if (Files.exists(rutaSin)) {

Files.delete(rutaSin);

}

Files.move(

Paths.get("Sintax.java"),

Paths.get("src/codigo/Sintax.java")

);

FrmPrincipal form = new FrmPrincipal();

form.setVisible(true);

}

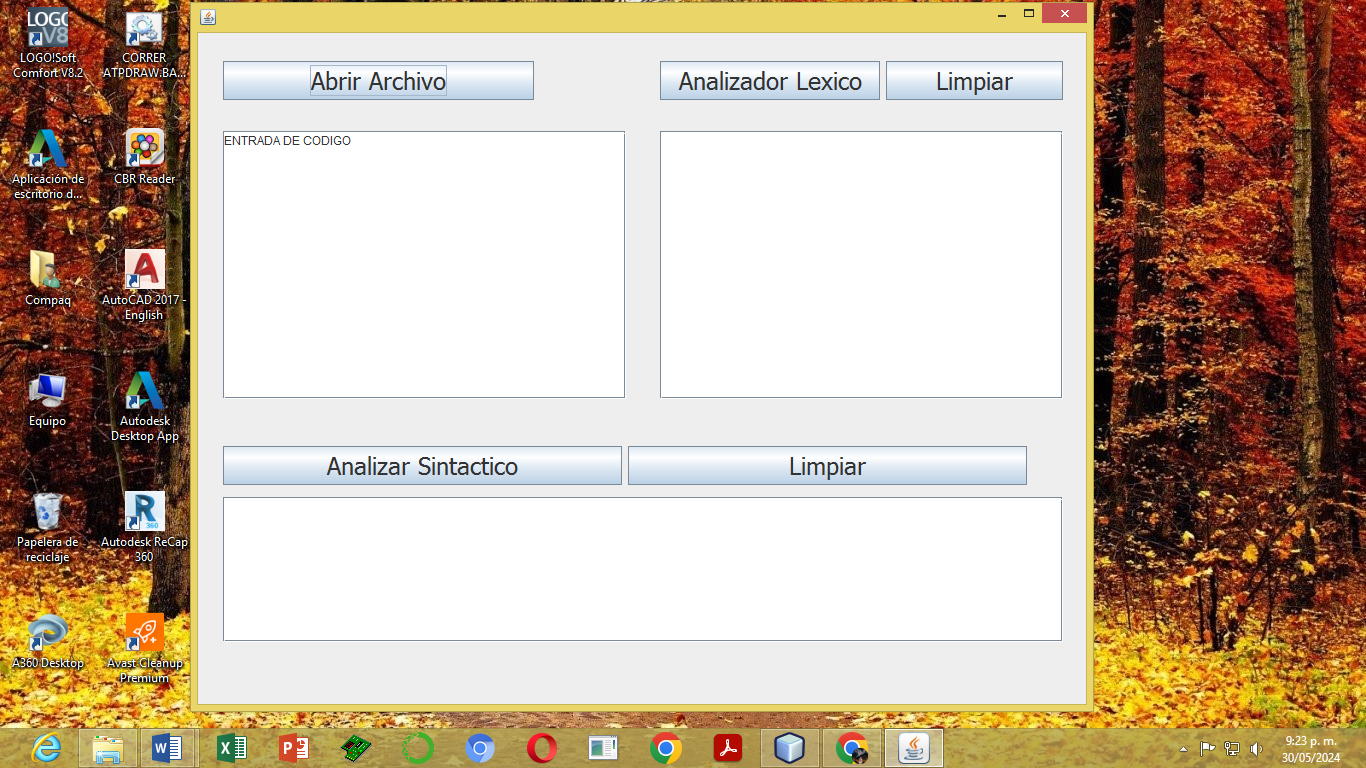
}

Desafíos del proyecto

Fue un gran desafío como comenzar el proyecto ya que al principio no tenía una idea clara de cómo empezarlo, unos de los mayores desafíos fue el tiempo, ya que se dificulto mucho por el trabajo, también fue un desafío la codificación y el análisis que llevo el codigo, los errores que tenía el codigo, las pruebas fallidas que hubo.

Decisiones de Diseño

Una vez definido como comenzaría a realizar el proyecto, comencé a estructural el mini-compilador, con una pantalla que tuviera los botones siguientes:



Esta pantalla esta realizado con un “JPanel From.Java”, donde tiene los botones: Abrir Archivo, Analizador Léxico, Limpiar, Analizador Sintáctico y Limpiar y cada botón tiene su codigo para el funcionamiento.

Esta pantalla fue un desafío ya que lo que se complicó más fue el codigo para el buen funcionamiento de los botones, hubo varios intentos ya que las pruebas que se hacían no jalaban o el codigo daba varios errores y cuando le daba un clik a un botón este no funcionaba, esta parte fue tediosa y frustrante, porque no funcionaba y por más que intentaba, hasta que al fin pude hacer que los botones funcionaran, pero fue un gran desafío llegar a ese punto.

Lecciones Aprendidas Durante el Desarrollo del Proyecto.

1. Tener bien claro los conceptos de cada tema que seda en la clase de compiladores.
2. Administrar bien el tiempo ya que si no puede ser un factor fatal para la persona que esté trabajando en el proyecto.
3. Tener el entorno listo para empezar a trabajar en el proyecto, sería el lenguaje de programación, librerías, el Jflex y el Cup tenerlos listos. Ya que, sino esto nos puede consumir tiempo en buscar, descargar e instalar cada uno de los entornos de trabajo.
4. Tener bien definido que es lo que se está trabajando y a donde queremos llegar.
5. No hay que desesperarse y no enojarse si no el proyecto nunca saldrá, me paso a mí, sí que recomiendo no hacer esto.
6. Tener subidas solo las librerías a utilizarse y seria lo primero que hay que subir.