

Universidad de San Carlos de Guatemala

Centro Universitario de Occidente

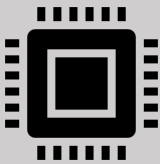
División de Ciencias de la ingeniería

Ingeniería

Organización de Lenguajes y Compiladores 1 Sección “A”

Ing. Jose Moises Granados Guevara

Primer Semestre 2026



**Estudiante:**

***Pablo Alejandro Maldonado de León***

***Carné: 202430233***

***Manual tecnico Práctica #1***



**Descripción:**

En el siguiente documento se da a conocer los detalles a tener en cuenta sobre la presente práctica. Sobre todo si se desea ampliar las funcionalidades de la aplicación se deben de conocer a detalle cómo es que funciona el intérprete realizado y sobre todo las clases y los diseños que fueron aplicados para poderla desarrollar.

## **“Bienvenido programador”**

Como primer punto se requiere contar con los pasos para poder ejecutar el código del repositorio en la computadora local del programador, al igual que conocer la distribución de carpetas que se hizo.

### **Recomendaciones:**

Debido a que la aplicación fue desarrollada en Android Studio se le recomienda al programador tener los siguientes requisitos:

- Android Studio o extensión de Android en IntelliJ IDEA
- Tener un teléfono Android o algún emulador instalado
- **Opcional:** Un teléfono con pantalla grande (esto con el motivo de tener una mejor visualización)

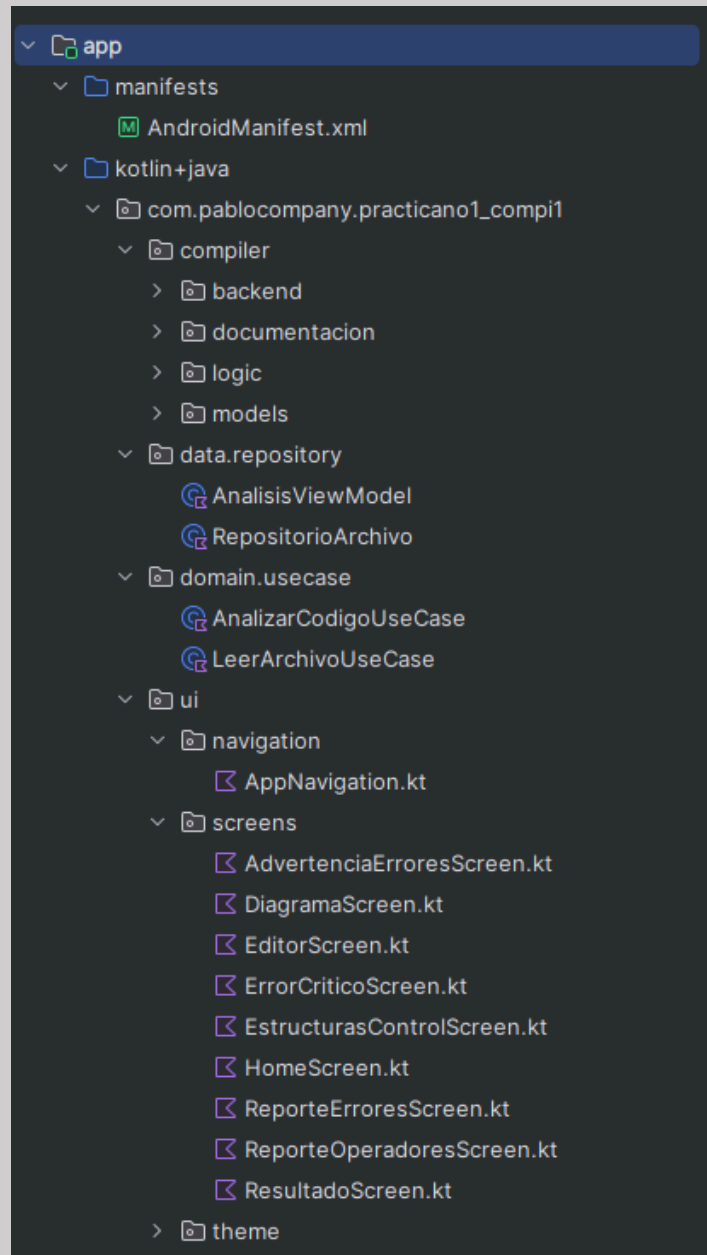
### ***Pasos iniciales:***

1. Encender la computadora.
2. Crear una carpeta donde se va a clonar el repositorio.
3. Ejecutar el comando:  
**git clone <https://github.com/pablo-mald03/Practica-No1-Compi-1.git>**
4. Tras descargar el repositorio se puede inicializar Android Studio.
5. Buscar la ruta de la carpeta donde fue clonado el repositorio y abrirla en Android Studio.
6. Al abrirlo se deben de aceptar los derechos

Si en dado caso se le pide descargar una extensión de **Cup**. Queda a discreción del programador si apoyarse de la extensión. Sin embargo no se recomienda ya que no tiene del todo las funciones necesarias y tiende a confundir.

## DIRECTORIO DE ARCHIVOS

Debido a que el proyecto base se realizó con cierta distribución de archivos. Así que se le recomienda al nuevo programador que mantenga el estándar que se maneja en el nombramiento de variables.



### Detalles importantes:

- **La carpeta “compiler”:** es donde se encuentra toda la lógica del intérprete y de la demás lógica de negocio que maneja la aplicación.
- **La carpeta “data”:** es donde se manejan los datos que se comparten vía enrutamiento a frontend para que este pueda interpretarlos
- **La carpeta: “usecase”:** maneja los controladores que sirven de intermediarios para delegar al frontend instrucciones de qué hacer en base a las acciones.

### Paquete “ui”

Este paquete es el que gestiona por completo las vistas de la ui usando **JetpackCompose** por lo tanto se le piden al programador conocimientos básicos en esta opción de Android para que se puedan ampliar las vistas. **Importante siempre mantener el formato:**

MiVistaScreen.kt

**Este es un convenio general para evitar revolver código y confundir vistas sobre todo para futuros demás programadores.**

### IMPORTANTE:

Si se desean crear más paquetes es válido. Solo que si se debe de respetar la distribución base de los paquetes ya mencionados para seguir manteniendo el formato de orden. Sobre todo si hay que agregar documentación se le pide al programador que lo adjunte dentro del paquete llamado **documentación**.

# ANÁLISIS DE GRAMÁTICA PARA ANALIZADOR LÉXICO

En el presente proyecto se lleva a cabo un análisis léxico gracias a JFlex que permite generar un **autómata finito determinista** en base a los principios del método de Thompson hasta poder llegar a un autómata con todos los estados que permita validar los estados de aceptación.

Por lo tanto el método consiste en:

- Declaración de Expresiones regulares
- Generación de AFN **por el método de Thompson**
- Formulacion de AFD **en base al algoritmo de cerradura Lambda**
- **Minimizacion de estados**

Aca se adjuntan todas las expresiones regulares utilizadas en el lexer:

## Macros utilizados:

```
LineTerminator = \r|\n|\r\n
WhiteSpace = [ \t\f]+
Numero = [0-9]+
Decimal = {Numero} "." {Numero}
jletter = [:jletter:]
jletterdigit = [:jletterdigit:]

Id = {jletter}{jletterdigit}*

HexColor = "H"[0-9A-Fa-f]{6}
```

### Expresiones regulares utilizadas:

```
{WhiteSpace} { /* ignorar */ }

\u200B      { /* ignorar */ }

\uFEFF      { /* ignorar */ }

\u200D      { /* ignorar */ }

{LineTerminator} { /*ignorar */ }

"#" .*      { /*Ignorado*/ }


{HexColor} { return symbol(sym.COLOR_HEX, yytext()); }


"|"        {return symbol(sym.PLECA);}

"="        {return symbol(sym.IGUALACION);}

"=="       {return symbol(sym.IGUALDAD);}

"!="       {return symbol(sym.DIFERENTE);}

">"        {return symbol(sym.MAYOR);}

"<"        {return symbol(sym.MENOR);}

">="       {return symbol(sym.MAYOR_IGUAL);}

"<="       {return symbol(sym.MENOR_IGUAL);}
```

```

"&&"      {return symbol(sym.AND);}

"||"      {return symbol(sym.OR);}

"!"       {return symbol(sym.NOT);}

"+"       {return symbol(sym.SUMA);}

"-"       {return symbol(sym.RESTA);}

"*"       {return symbol(sym.MULTIPLICACION);}

"/"       {return symbol(sym.DIVISION);}

"%%%"     {return symbol(sym.SEPARADOR);}

", "      {return symbol(sym.COMA);}

"%DEFAULT" {return symbol(sym.DEFAULT);}

"%COLOR_TEXTO_SI" {return symbol(sym.COLOR_TEXTO_SI);}

"%COLOR_SI" {return symbol(sym.COLOR_SI);}

"%FIGURA_SI" {return symbol(sym.FIGURA_SI);}

```

```

"%LETRA_SI"          {return symbol(sym.LETRA_SI);}

"%LETRA_SIZE_SI"     {return symbol(sym.LETRA_SIZE_SI);}

"%COLOR_TEXTO_MIENTRAS"  {return symbol(sym.COLOR_TEXTO_MIENTRAS);}

"%COLOR_MIENTRAS"      {return symbol(sym.COLOR_MIENTRAS);}

"%FIGURA_MIENTRAS"    {return symbol(sym.FIGURA_MIENTRAS);}

"%LETRA_MIENTRAS"     {return symbol(sym.LETRA_MIENTRAS);}

"%LETRA_SIZE_MIENTRAS"  {return symbol(sym.LETRA_SIZE_MIENTRAS);}

"%COLOR_TEXTO_BLOQUE"  {return symbol(sym.COLOR_TEXTO_BLOQUE);}

"%COLOR_BLOQUE"        {return symbol(sym.COLOR_BLOQUE);}

"%FIGURA_BLOQUE"      {return symbol(sym.FIGURA_BLOQUE);}

"%LETRA_BLOQUE"        {return symbol(sym.LETRA_BLOQUE);}

"%LETRA_SIZE_BLOQUE"   {return symbol(sym.LETRA_SIZE_BLOQUE);}

"%INICIO"             {return symbol(sym.INICIO);}

```

```

"FIN"      {return symbol(sym.FIN);}

"VAR" {return symbol(sym.VAR);}

"SI" {return symbol(sym.SI);}

"ENTONCES" {return symbol(sym.ENTONCES);}

"MIENTRAS" {return symbol(sym.MIENTRAS);}

"MOSTRAR" {return symbol(sym.MOSTRAR);}

"LEER"      {return symbol(sym.LEER);}

"FIN MIENTRAS"|"FINMIENTRAS"      {return symbol(sym.FIN_MIENTRAS);}

"HACER"      {return symbol(sym.HACER);}

"FIN SI"|"FINSI"      {return symbol(sym.FIN_SI);}

"("      {return symbol(sym.PARENT_APERTURA);}

")"      {return symbol(sym.PARENT_CIERRE);}

"ELIPSE"      {return symbol(sym.FIGURA,"ELIPSE");}

```

```

"CIRCULO"          {return symbol(sym.FIGURA,"CIRCULO");}

"PARALELOGRAMO"    {return symbol(sym.FIGURA,"PARALELOGRAMO");}

"RECTANGULO"       {return symbol(sym.FIGURA,"RECTANGULO");}

"ROMBO"            {return symbol(sym.FIGURA,"ROMBO");}

"RECTANGULO_REDONDEADO" {return symbol(sym.FIGURA,"RECTANGULO_REDONDEADO");}

"ARIAL"            {return symbol(sym.FUENTE,"ARIAL");}

"TIMES_NEW_ROMAN"  {return symbol(sym.FUENTE,"TIMES_NEW_ROMAN");}

"COMIC_SANS"       {return symbol(sym.FUENTE,"COMIC_SANS");}

"VERDANA"          {return symbol(sym.FUENTE,"VERDANA");}

{Decimal} {return symbol(sym.DECIMAL, Double.parseDouble(yytext()));}

{Numero} {return symbol(sym.ENTERO, Integer.parseInt(yytext()));}

{Id} { return symbol(sym.ID, yytext()); }

\"          { string.setLength(0); yybegin(STRING); }

```

**Estados utilizados:**

**<YYINITIAL>{**

**}**

**<STRING>{**

**\" {**

yybegin(YYINITIAL);

return symbol(sym.CADENA, string.toString());

**}**

**[\n\r\"\\]+ {**

string.append(yytext());

**}**

**\\\" {**

string.append("\\");

**}**

**\\n {**

string.append("\\n");

**}**

**\\t {**

string.append("\\t");

**}**

**\\ { string.append("\\"); }**

**}**

# ANÁLISIS DE GRAMÁTICA PARA ANALIZADOR

## SINTÁCTICO

Cabe aclarar que aquí simplemente se están agregando los **terminales** y los **no terminales** debido a que la gramática es muy extensa. pero simplemente se puede visualizar la cantidad de derivaciones que esta tiene. Puesto a que los errores se debían detectar desde el Parser el código es bien extenso. Por lo tanto par más información consultar el archivo llamado

**ParserConfig.cup**

```
terminal      PLECA, IGUALACION,

              IGUALDAD, DIFERENTE, MAYOR, MENOR, MAYOR_IGUAL, MENOR_IGUAL,

              AND, OR, NOT,

              SUMA, RESTA,

              MULTIPLICACION, DIVISION,

              SEPARADOR, COMA, DEFAULT,

              COLOR_TEXTO_SI, COLOR_SI, FIGURA_SI, LETRA_SI,

              COLOR_TEXTO_MIENTRAS, COLOR_MIENTRAS, FIGURA_MIENTRAS,
LETRA_MIENTRAS, LETRA_SIZE_MIENTRAS,LETRA_SIZE_SI, LETRA_SIZE_BLOQUE,

              COLOR_TEXTO_BLOQUE, COLOR_BLOQUE, FIGURA_BLOQUE, LETRA_BLOQUE,

              INICIO, FIN, VAR, SI, ENTONCES, MIENTRAS, MOSTRAR, LEER,
FIN_MIENTRAS, HACER, FIN_SI,

              PARENT_APERTURA, PARENT_CIERRE,

              FIGURA, FUENTE;

/*=====Apartado de tipado de tokens que vienen del lexer=====*/

terminal String ID, COLOR_HEX;

terminal Double DECIMAL;
```

```

terminal Integer ENTERO;

terminal String CADENA;


/* no terminales, reglas que nosotros vamos a definir sintacticamente */

non terminal NodoPrograma programa;

non terminal List seccion_configuracion;

non terminal List lista_configuraciones;

non terminal NodoInstruccion si_simple;

non terminal NodoInstruccion mientras_simple;

non terminal NodoInstruccion declaracion;

non terminal NodoInstruccion asignacion;

non terminal NodoInstruccion mostrar;

non terminal NodoInstruccion leer;


/*===Representacion de una expresion matematica===*/

non terminal NodoExpresion expresion;


/*===Representacion de una expresion matematica===*/


/*No terminales para poder detectar listados completos de bloques de
codigo*/

```

```

non terminal List lista_sentencias_simples;

non terminal NodoInstruccion sentencia_simple;


non terminal NodoInstruccion estructura;

non terminal List lista_estructuras;


/*Fin de los no terminales para poder detectar listados completos de bloques
de codigo*/


/*==Gramatica de personalizacion==*/


non terminal NodoConfiguracion config_estilos;

non terminal NodoColor valor_color;

non terminal NodoRgb rgb_estado;


/*==Fin de la Gramatica de personalizacion==*/


/*====Sub region de estados de personalizacion*/


non terminal NodoConfiguracion config_color;

non terminal NodoConfiguracion config_figura;

non terminal NodoConfiguracion config_fondo;

non terminal NodoConfiguracion config_size_letra;

non terminal NodoConfiguracion config_tipo_letra;

```

```

/*====Fin de la region de estados de personalizacion====*/

//=====Gramatica de condiciones=====

non terminal NodoExpresion condicion;

non terminal NodoExpresion condicion_logica;

```

### Cabe destacar algo muy importante:

Al usar expresiones matemáticas esto causaba Shift-Reduce. Es decir que la gramática se hacia ambigua al no indicarle a cup cuál era la precedencia de operadores. Por lo tanto se opto por la herramienta de **precedencia**. Que permite indicar la dirección de la precedencia que siguen los operadores.

```



```

### IMPORTANTE:

Debido a que la precedencia de los operadores lógicos también se requiere saber por donde genera su efecto y debido a que el operador NOT casi siempre se aplica en una expresión este es el que debe de tener la principal precedencia en la jerarquia. Por lo tanto la jerarquia queda asi:

**! (NOT) se evalúa primero**

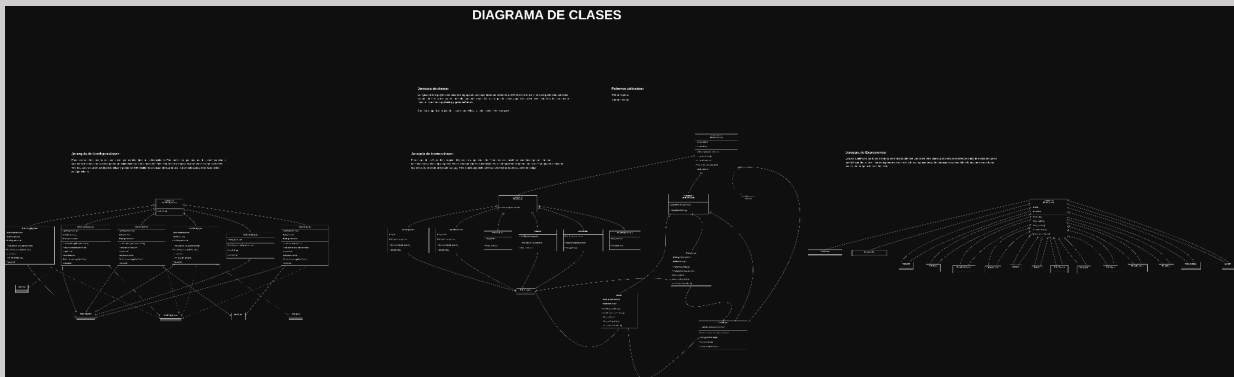
**&& (AND) después**

**|| (OR) al final**

## DIAGRAMA DE CLASES

Este es uno de los diagramas que describe la jerarquía principal que sigue el intérprete para poder generar sus diagramas. Simplemente es una jerarquía con métodos básicos. **Cabe destacar que principalmente se pensó para poderse ejecutar en algunas acciones. Pero de momento el proyecto no lo requería por lo tanto fueron eliminadas esas funcionalidades.**

Aca se deja la implementación base:



**Puedes verlo más a detalle en el link:**

<https://app.diagrams.net/?src=about#G1zK4lkH39i2xAj4c5rAWhBg3yFYfcuXj3#%7B%22pageId%22%3A%229FyhZFpsurD1-RChVbB9%22%7D>