MANUAL TECNICO DE ANALIZADOR SINTACTICO

by Juan Pablo Orizabal Gil
Carné: 201730318

MANUAL TÉCNICO

El software fue creado en una laptopDell Insipiron n9010, con 4gb de RAM, con un procesador intel core i5. El software fue aprobado por los distintos servicios de calidad que se requieren a nivel internacional.

El software es un analizador lexico y sintactico, esto quiere decir que podemos realizar distintas funciones, como analizar palabras dentro de un texto, reconocer patrones, así como exportarlo a un archivo de texto y ademas podemos analizar sintacticamente las estructuras, es decir el orden en el que vienen apareciendo los tokens.

El software consta de tecnologia Java, para poder analizar y convertir en los tokens adecuados el texto plano que contiene las cadenas de texto que pueden ir incluidas.

PAQUETE ANALISIS SINTACTICO
CLASE Analisis Sintactico

Esta clase es la que contiene a todos los métodos y funciones que se utilizaron para cumplir con la logica que el analizador sintactico requeria. Aqui estan programados todos los distintos estados del automata de pila.

METODOS

inicioAnalisis: solo llama a q0.
Q0: Aqui empieza el analisis poniendo estructura 0
q1: aquí reemplaza por estructura 1 -5 dependiendo cual sea la correccta.
EstadosEstructuraGlobal: Desde aquí llama a los estructuras 1-5.
estadoEstructura1: El estado de estructura 1 es donde empezara la derivacion para que sea la estructura escribir
estadoEstructura2: El estado estructura 2 es donde empezara la derivacion para que sea una estructura repetir
estadoEstructura3: El estado estructura 3 es donde empezara para que la derivacion sea un estructura repetir.
estadoEstructura4:El estado 4 es donde empezara la derivacion para que sea una estructura asignacion
estadoEstructura5: el estado 5 es donde empezara la derivacion para que sea una estructura de expresion
derivacionToken2: Aqui deriva el no terminal token2 que solo deriva en numero o identificador.
VerVecesRepetir: mira cuantas veces dice el token numero que hay que repetir.
VerificarTokenBooleano: verifica que tipo de token booleano es verdadero o falso
hacerEscritura: es donde hará la escritura, mira si lo guarda directamente en el arraylist para escribir o tiene que pasar por otro bufer intermedio de la estructura repetir
derivacionToken: hace la derivacion del no terminal token, en numero, identificador o literal
deriviacionBoolean: hace la derivacion del no terminal boolean en verdadero o falso
derivacionVerEstructuras: hace la derivacion del no terminal ver estructura que deriva en una estructura1 y otra vez ver estructuras o en solo ver estructuras y fin
guardarObjetosRepetir: guarda el bufer intermedio los valores
escribirRepetir: es donde guarda del bufer intermedio al bufer final de donde se escribira
derivacionDeT: hace todo lo necesario para poder derivar el no terminal t
derivacionDeF:hace todo lo necesario para poder derivar el no terminal f
derivacionDeS:hace todo lo necesario para poder derivar el no terminal s
asignarValores:asigna los valores al identificador de expresion y asignacion
cambiarValorIdentificador: cambia el valor de los identificadores.

Clase Constantes

Esta clase contiene las constantes a utilizar de los tokens.

Metodos

Solo contiene los getters de las constantes finales de la clase.

Clase ElementosAuxilares

Esta clase contiene un poyo que servira para poder guardar el historial de los movimientos de la pila y el indice de los tokens.

Metodos

Solo contiene setters y getters de los atributos del poyo.

Clase HacerErrores

Esta clase contiene los métodos para hacer errores, así como los de escribir en el archivo de salida.

Metodos

reportarError: reporta el error sintactico en la lista de la interfaz
mostrarErrores:solo muestra los errores en consola
agregarElementosEscribir: es donde agrega los datos al archivo de salida
escribirTexto: escribe en el archivo de salida
arreglarNumero: si entra un token numero sale su lexema
arreglarIdentificador:si entra un token identificador sale su lexema
arreglarLiteral: si entra un literal sale su lexema sin comillas
quitarComillasALaLinea: le quita las comillas a una cadena de texto.

Clase Pila

Esta clase contiene el modelo de la pila que se va a utilizar, tiene todos los métodos para poder usar la pila.

Metodos

push: agrega un elemento a la pila
pop: quita el elemento de hasta arriba de la pila
peek: agarra el valor de hasta arriba de la pila
empty: mira si la pila esta vacia
getPila: obtiene la pila
setPila: asigna una pila ya creada a esta
quitarPoner: quita un elemento y agrega otro por el parametro
clear: limpia la pila

PAQUETE ANALIZADOR LEXICO

CLASE Analisis Lexico

Esta clase contiene los métodos para crear el analisis lexico

METODOS

crearObjetoLexer: crea un objeto lexer
hacerAnalisisLexico: hace el analisis lexico de un objeto lexer
obtenerTokens: obtiene los tokens del analisis
obtenerTokensError: obtiene los tokens de error del analisis.

CLASE Lexer

Esta clase es el analisis lexico como tal, contiene todo lo relacionado al analizador lexico, esta hecho por jflex.

METODOS

Todo lo de jflex que es ajeno al proyecto como tal, ya que es una ayuda solo para poder enfocarse en el analisis sintacticoy no en el lexico.

Clase Token

Esta clase contiene todo los atributos que se necesitan para guardar los tokens y sus lexemas, tiene nombre token, lexema, fila, columna, veces.

Metodos

Tiene 3 constructores sobrecargados para instanciar de distintas formas
y sus respectivos setters y getters de los atributos privados.

PAQUETE GUI

Clase Validaciones

Esta clase es un jframe que tiene jdesktop pane y menú bar, aquí es como la ventana principal de la aplicación y es el front end de todo lo que se hace. Tiene varias default table modelos, y otros strings que serviran para la logica de la aplicación.

Metodos

asignarValoresModelos: asigna valores a los modelos de las tablas
ocultarBotones: oculta los botones de recuento de lexemas y tokens
mostrarBotones: muestra los botones de recuento de lexemas y tokens
asignar: asigna las funcionalidades de la aplicación de copiar,pegar, entre otros.
Guardar: guarda los cambios de la aplicacion
guardarComo: guarda los cambios en un archivo especifico
nuevo: abre una nueva pestaña del editor de texto
borrarDatos: borra los datos de las tablas
vaciarModelos: vacia el modelo de las tablas

Clase Tablas Visuales

Esta clase es un jframe que tiene que tiene una j table dentro y lo unico que hace es mostrar el recuento de lexemas y de tokens de la aplicación, se le manda como parametro el modelo y lo unico que hace este frame es agregarlo a su tabla y lo muestra.

Metodos

No tiene métodos.

PAQUETE LOGICA

Clase Cargar Archivo

Esta clase contiene dos atributos un file y una linea de código privadas.

Metodos

append: agrega texto al final del text area.
LeerArchivo: lee el archivo de entrada y agrega los datos al text area.
numerar: numera las filas que contiene el text area.

Clase Logica Archivo

Esta clase es la encargada de las funcionalidades del editor de texto de la aplicación.

Metodos

guardarArchivo: Agarra los datos del archivo y los almacena en el archivo que tenga seleccionado
guardarComoArchivo: Abre un filechooser para preguntar donde guardar la informacion
nuevoArchivo: Crea una nueva hoja del editor de texto
verificarArchivo: verifica si el archivo ha sufrido cambios para preguntar si desea guardar

Clase My AdjustmentListener

Esta clase contiene dos jscrollpanes que son los del text area y de la numeracion, lo que hace es caputrar un evento para que cuando se mueva un scroll se mueva el otro

Metodos

adjustmentValueChanged: lo que hace es detectar si se mueve uno se mueva el otro scroll pane

PAQUETE analizador sintactico

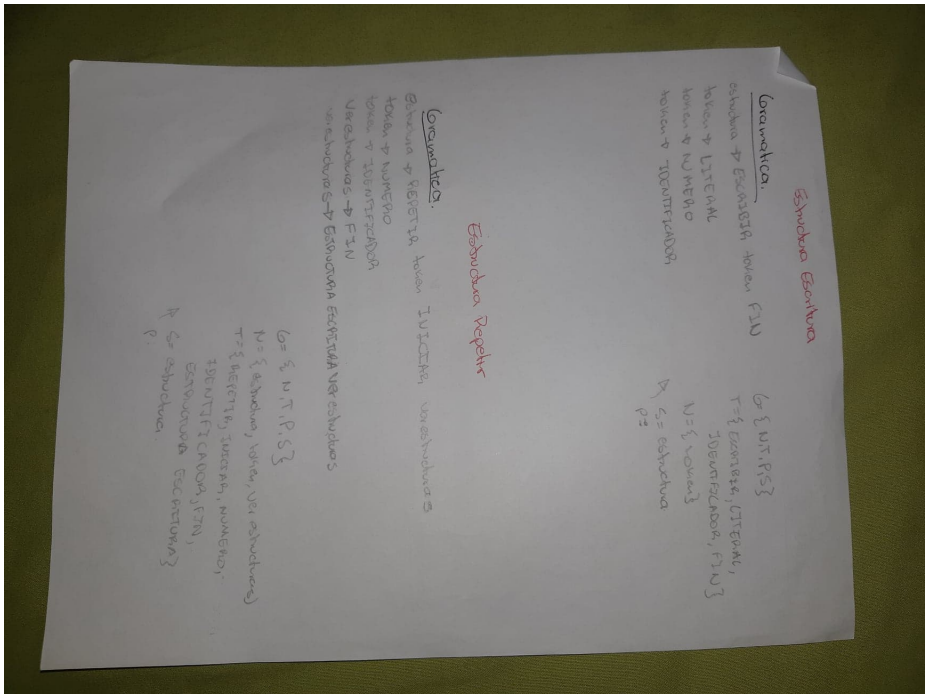
Clase Analizador Sintactico

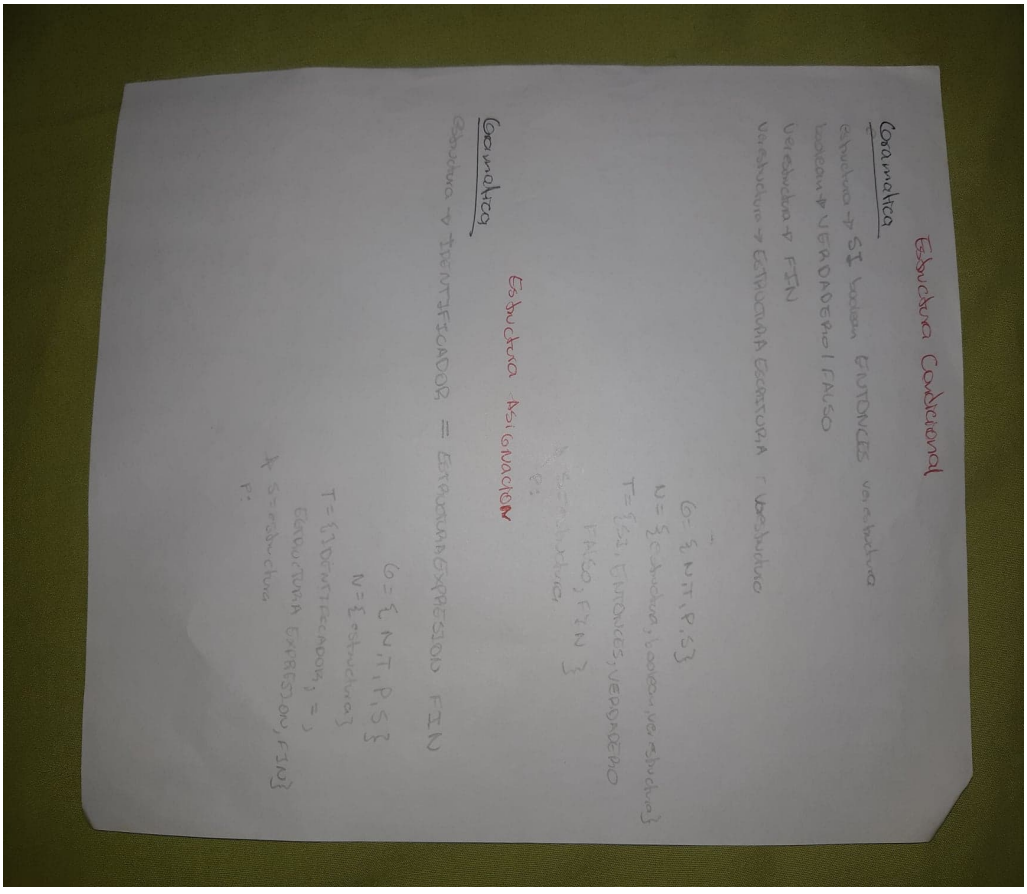
Esta clase es la que contiene el main, de donde parte la aplicación, solo llama al menú principal.

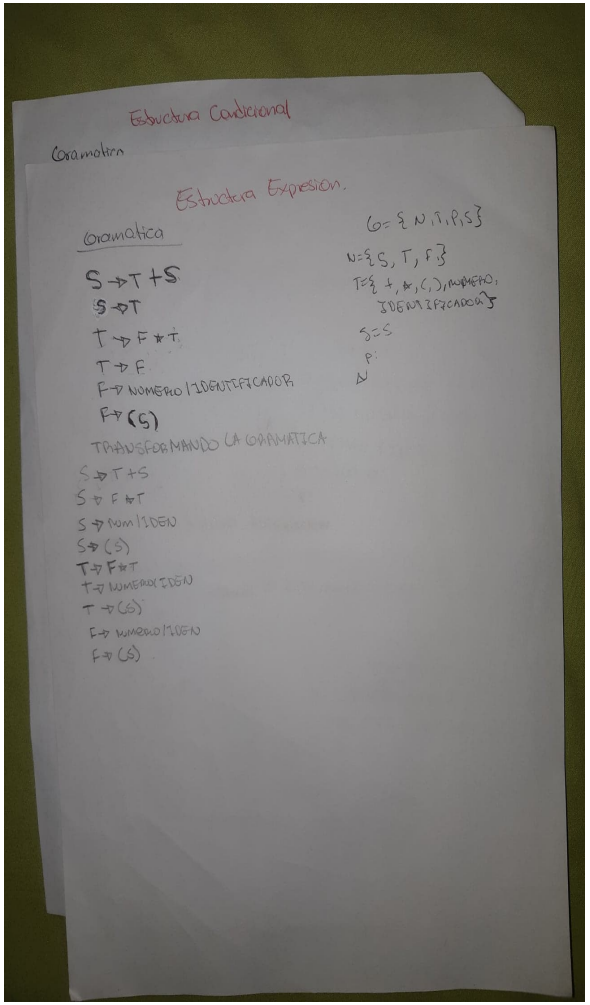
Metodos

main: donde empieza la aplicación.

TRABAJO TEORICO PRACTICO







Estructura Condicional

Gramatica

Estructura Expresion.

Gramatica

G = { N, T, P, S }

S → T + S

S → T

T → F * T

T → F

F → NUMERO | IDENTIFICADOR

F → (S)

N = { S, T, F }

T = { +, *, (,), NUMERO, IDENTIFICADOR }

S = S

P:

TRANSFORMANDO LA GRAMATICA

S → T + S

S → F * T

S → NUM | IDEN

S → (S)

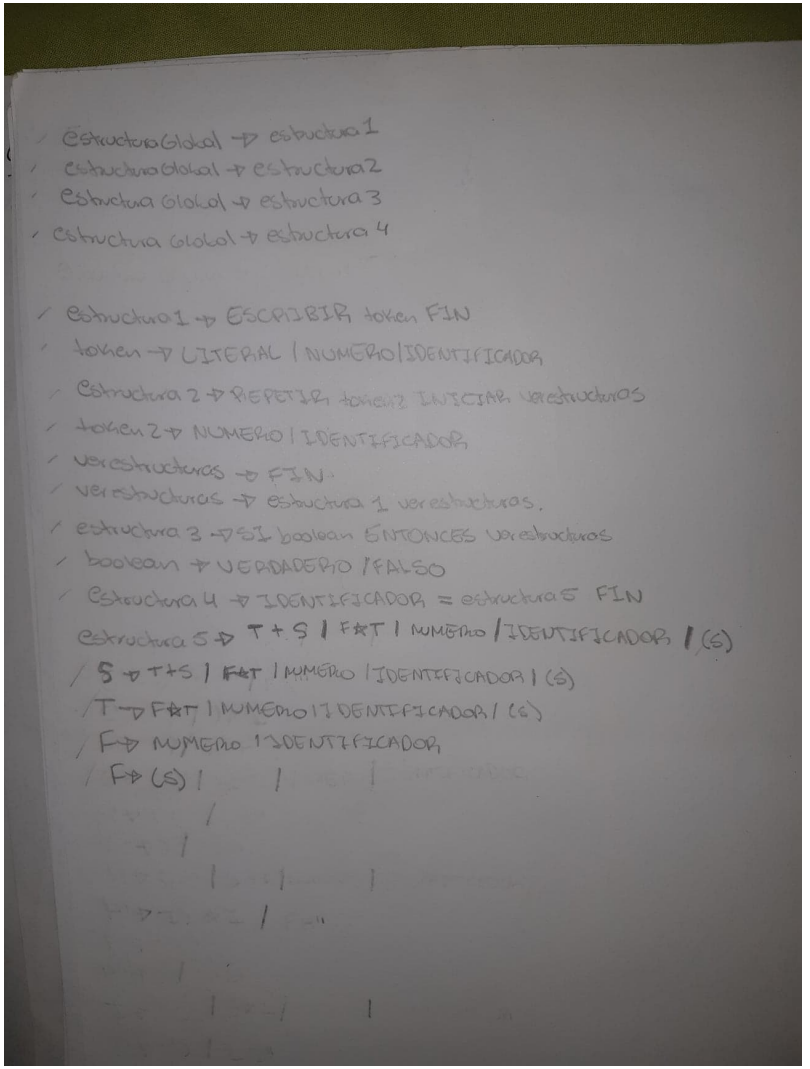
T → F * T

T → NUMERO | IDEN

T → (S)

F → NUMERO | IDEN

F → (S)



- EstructuraGlobal → estructura 1
- EstructuraGlobal → estructura 2
- EstructuraGlobal → estructura 3
- EstructuraGlobal → estructura 4

- estructura 1 → ESCRIBIR token FIN
- token → LITERAL | NUMERO | IDENTIFICADOR
- Estructura 2 → REPETIR token2 INICIAR verestructuras
- token 2 → NUMERO | IDENTIFICADOR
- verestructuras → FIN
- verestructuras → estructura 1 verestructuras
- estructura 3 → SI boolean ENTONCES verestructuras
- boolean → VERDADERO / FALSO
- Estructura 4 → IDENTIFICADOR = estructura 5 FIN
- estructura 5 → T + S | F * T | NUMERO | IDENTIFICADOR | (S)
- S → T + S | F * T | NUMERO | IDENTIFICADOR | (S)
- T → F * T | NUMERO | IDENTIFICADOR | (S)
- F → NUMERO | IDENTIFICADOR
- F → (S) |