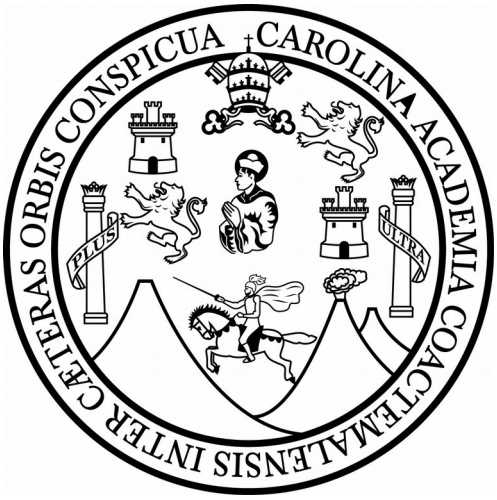
CENTRO UNIVERSITARIO DE OCCIDENTE

DIVISIÓN CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA CIENCIAS Y SISTEMAS



LABORATORIO DE ORGANIZACIÓN LENGUAJES Y COMPILADORES 2

“SEXTO SEMESTRE”

ING.: JOSÉ MOISÉS GRANADOS GUEVARA

AUX: EDVIN TEODORO GONZALEZ RAFAEL

ESTUDIANTE: Bryan René Gómez Gómez – 201730919

PROYECTO: “Manual Técnico – PRIMER PROYECTO - ANALIZADOR”

FECHA: 03 de Septiembre de 2,020

Analizador Léxico

TOKEN	TIPO
nombre	**Palabra Reservada**
version	**Palabra Reservada**
autor	**Palabra Reservada**
extension	**Palabra Reservada**
lanzamiento	**Palabra Reservada**
terminal	**Palabra Reservada**
no	**Palabra Reservada**
entero	**Palabra Reservada**
cadena	**Palabra Reservada**
real	**Palabra Reservada**
=	**Signo igual**
;	**Signo punto y coma**
:	**Signo dos puntos**
(	**Signo Paréntesis Abrir**
)	**Signo Paréntesis Cerrar**
-	**Signo menos**
*	**Signo por**
+	**Signo mas**
[	**Llave Abir**
]	**Llave Cerrar**
?	**Interrogación Cerrada**
,	**Coma**
&	**Ampersand**
->	**Barra Vertical**
\n	**Salto de linea**
\t	**Espacio Tabulador**
\b	**Espacio en blanco**
(Digito)+	**Expresión Numero**
(Digito)+((.)(Digito)+)*	**Expresión Decimal**
LetraMinuscula	**Expresión Terminal**
LetraMayuscula	**Expresión No Terminal**
(Letra -> “_”) → (Letra -> “_” -> Dígito)*	**Expresión Identificador**
(Dígito)+	**Expresión Número**

Comillas → (Letra -> Símbolo -> Dígito) + → Comillas	**Expresión Cadena**
“/*” → (Letra -> Símbolo -> Dígito) + → “*/”	**Expresión Comentario Completo**
“//” → (Letra -> Símbolo -> Dígito) +	**Expresión Comentario de Linea**
%% [^%] %%	**Expresión Código Java**
{ [^};] };	**Expresión Regla Semántica**

ANALIZADOR SINTÁCTICO

G = [N, T, P, S **]**

Terminal (T):

NOMBRE, VERSION, AUTOR, EXTENSION, LANZAMIENTO, TERMINAL, NO, ENTERO, CADENA, REAL, SALTO_LINEA, ESPACIO_B, TABULADOR, CODIGO_FUENTE;, PORCENTAJES, IGUAL, PUNTO_COMA, COMILLAS, PARENTESIS_A, PARENTESIS_C, LLAVE_A, LLAVE_C,MENOS, INTERROGACION_C, MULTIPLICACION, MAS, COMA, AMPERSAND, BARRA_VERTICAL, DOS_PUNTOS, REGLAS_SEMANTICAS, NUMERO, MAYUSCULA, IDENTIFICADOR, VERSION_NUMEROS, MINUSCULAS, LETRA_MIN, LETRA_MAY, SIGNO

No Terminal (N):

inicio, inf_lenguaje, exp_inf_lenguaje, p_c, p_nombre, p_version, p_autor, p_lanzamiento, p_extension, p_palabras, pp_palabras, exp_pal_reg, exp_signos, cantidad_expresion, tnt, tnt_p, simbolo_terminal, simbolo_no_terminal, estructura_no_terminales, estructura_terminales, inicio_exp_reg, exp_regulares, exp_reg, cuerpo_exp_parentesis, cuerpo_exp_llave, contenido_exp_llave, contenido_exp_llave_p, s_exp_regulares, contenido_exp_parentesis, exp_regulares_p, p_tipo, reglas, contenido_reglas, cuerpo_reglas_p, reglas_semanticas, reglas_p, pro_may, pro_min, cuerpo_reglas,

S → inicio

Producciones (P):

inicio

-> inf_lenguaje CODIGO_FUENTE:c inicio_exp_reg tnt reglas

-> inf_lenguaje inicio_exp_reg tnt reglas

inf_lenguaje

-> p_nombre exp_inf_lenguaje exp_inf_lenguaje exp_inf_lenguaje exp_inf_lenguaje

-> exp_inf_lenguaje p_nombre exp_inf_lenguaje exp_inf_lenguaje exp_inf_lenguaje

-> exp_inf_lenguaje exp_inf_lenguaje p_nombre exp_inf_lenguaje exp_inf_lenguaje

-> exp_inf_lenguaje exp_inf_lenguaje exp_inf_lenguaje p_nombre exp_inf_lenguaje

-> exp_inf_lenguaje exp_inf_lenguaje exp_inf_lenguaje exp_inf_lenguaje p_nombre

-> error PUNTO_COMA

exp_inf_lenguaje

-> p_version

-> p_autor

-> p_lanzamiento

-> p_extension

->

p_nombre -> NOMBRE DOS_PUNTOS p_palabras PUNTO_COMA

p_version

-> VERSION DOS_PUNTOS VERSION_NUMEROS PUNTO_COMA

-> VERSION DOS_PUNTOS NUMERO PUNTO_COMA

p_autor -> AUTOR DOS_PUNTOS p_palabras pp_palabras PUNTO_COMA

p_lanzamiento -> LANZAMIENTO DOS_PUNTOS NUMERO PUNTO_COMA

p_extension

-> EXTENSION DOS_PUNTOS MINUSCULAS PUNTO_COMA

-> EXTENSION DOS_PUNTOS LETRA_MIN PUNTO_COMA

p_palabras ->

-> MAYUSCULA

-> MINUSCULAS

-> IDENTIFICADOR

-> LETRA_MIN

-> LETRA_MAY

pp_palabras

-> p_palabras

->

tnt

-> TERMINAL p_tipo simbolo_terminal p_c tnt_p

-> NO TERMINAL p_tipo simbolo_no_terminal PUNTO_COMA tnt_p

p_c -> PUNTO_COMA

tnt_p

-> tnt

-> error PUNTO_COMA

-> error tnt

-> error reglas

->

p_tipo

-> REAL

-> ENTERO

-> CADENA

->

simbolo_terminal

-> MINUSCULAS estructura_terminales

-> LETRA_MIN estructura_terminales

estructura_terminales

-> COMA simbolo_terminal

->

simbolo_no_terminal

-> MAYUSCULA estructura_no_terminales

-> LETRA_MAY estructura_no_terminales

estructura_no_terminales

-> COMA simbolo_no_terminal

->

inicio_exp_reg

-> p_palabras IGUAL s_exp_regulares PUNTO_COMA exp_reg

-> AMPERSAND IGUAL s_exp_regulares PUNTO_COMA exp_reg

-> error PUNTO_COMA

-> error inicio_exp_reg

exp_reg

-> inicio_exp_reg

->

s_exp_regulares

-> exp_regulares

-> exp_regulares BARRA_VERTICAL exp_regulares

exp_regulares ->

LLAVE_A cuerpo_exp_llave LLAVE_C cantidad_expresion exp_regulares_p

-> COMILLAS cantidad_expresion exp_regulares_p

-> PARENTESIS_A cuerpo_exp_parentesis PARENTESIS_C cantidad_expresion exp_regulares_p

exp_regulares_p

-> exp_regulares

->

cantidad_expresion

-> INTERROGACION_C

-> MULTIPLICACION

-> MAS

->

cuerpo_exp_llave ->

-> exp_pal_reg MENOS exp_pal_reg

-> NUMERO MENOS NUMERO

-> exp_signos MENOS exp_signos

contenido_exp_llave ->

exp_pal_reg contenido_exp_llave_p

-> NUMERO contenido_exp_llave_p

-> exp_signos contenido_exp_llave_p

contenido_exp_llave_p ->

contenido_exp_llave

->

exp_pal_reg ->

LETRA_MIN

-> LETRA_MAY

exp_signos ->

SIGNO

-> IGUAL

-> PUNTO_COMA

-> DOS_PUNTOS

-> MENOS

-> INTERROGACION_C

-> MULTIPLICACION

-> MAS

-> COMA

-> AMPERSAND

-> SALTO_LINEA

-> ESPACIO_B

-> TABULADOR

cuerpo_exp_parentesis

-> contenido_exp_parentesis

-> s_exp_regulares

contenido_exp_parentesis

-> p_palabras

-> NUMERO

-> exp_signos

reglas

-> pro_may:s DOS_PUNTOS DOS_PUNTOS cuerpo_reglas reglas_semanticas reglas_p

-> pro_may:s DOS_PUNTOS DOS_PUNTOS reglas_semanticas reglas_p

cuerpo_reglas

-> contenido_reglas DOS_PUNTOS p_palabras:b cuerpo_reglas_p

-> contenido_reglas cuerpo_reglas_p

contenido_reglas

-> pro_min

-> pro_may

cuerpo_reglas_p

-> cuerpo_reglas

->

reglas_semanticas

-> REGLAS_SEMANTICAS

-> PUNTO_COMA

reglas_p

-> reglas

-> error PUNTO_COMA

-> error reglas

->

pro_min

-> MINUSCULAS

-> LETRA_MIN

pro_may

-> MAYUSCULA

-> LETRA_MAY

MÉTODOS Y CLASES

Paquetes

proyectoanalizador
proyectoanalizador.backed.analizador
proyectoanalizador.backed.analizador.manejadores
proyectoanalizador.backed.objetos
proyectoanalizador.backed.objetos.analizador.lexico
proyectoanalizador.backed.objetos.analizador.sintactico
proyectoanalizador.gui

ProyectoAnalizador

Clase Main

Clase Generador de archivos.

Generador():
Genera los archivos del analizador

Clase ManejadorArchivosBinarios

boolean crearArchivo(T tipoObjeto, String tipoArchivo, String identificadorUnico, String extensionArchivo)
Crea un archivo binario de un tipo de objeto.

T leerArchivo(String pathInicial, String pathNombreArchivo, String tipoDeArchivoPath)
Busca un archivo binario

List<T> leerListaArchivos(String extensionArchivo)
Carga a memoria los archivos con extension len para cargarlos en el repositorio.

Clase ManejadorEntrada

String entradaTexto(String entrada)
Lee el archivo de entrada de la carga de un lenguaje, para posteriormente realizar un analisis.

Frame.java

Constructor de la ventana
public Principal()

private void menuGuardarActionPerformed(java.awt.event.ActionEvent evt)
Método para guardar los cambios realizados por cada uno de los archivos en el editor de texto.

private void realizarAnalisisLCT(String comprobacion)
Método para realizar el análisis sintáctico y léxico de un archivo.

private void menuAcercaDeActionPerformed(java.awt.event.ActionEvent evt)
Método que muestra la información del desarrollador.

private void menuGuardarActionPerformed(java.awt.event.ActionEvent evt)
Método para exportar los lienzo al formato indicado.

Clase GeneradorHTML

void escribirArchivoSalida(String path, String textoSalida)
Genera el archivo .html para poder visulizar la tabla generada

String construirTabla(List<Estado> estadosC)
Construye la cadena de instrucciones html de la tabla LALR

private String tituloPag(String titulo)
Genera el nombre de la pagina HTML

String construirPila(List<String> registroEstados, List<String> registroSimbolos, List<String> registroAcciones)
Construye la pila mediante codigo html para poder visualizarlo graficamente.

String tablaPila(List<String> registroEstados, List<String> registroSimbolos, List<String> registroAcciones)
Costruye los elementos de la pila insertando tupla por tupla mediante su codigo de html.

String tablaLALR(List<Estado> estadosC)
Genera una cadena de instrucciones para generar la tabla que posee la tabla LALR

Clase ManejadorSintactico

Constructor de la Clase
private ManejadorSintactico() {}

Instancia de la clase
public static ManejadorSintactico getInstancia()

public void clear()
Método que realiza la limpieza dela lista de símbolos y la tabla de símbolos, para poder realizar un nuevo análisis.

public void errorSintax(int left, int right, String value, String mensaje)
Método que agrega al área de texto errores la descripción de un error sintactico.

public void errorSemantico(int left, int right, String valor, String mensaje)

Método que agrega al área de texto errores la descripción de un error semantico.

public Object addTabla(Object object, int left, int right, String tipo)

Agrega un nuevo símbolo a la tabla de símbolos

public InfLenguaje comprobarInformacion(int aleft, int aright, int eleft, int eright, int ileft, int iright, int oleft, int oright, int uleft, int uright, Object nombre, Object version, Object lanzamiento, Object autor, Object extension)

Metodo que comprueba la parte semantica del area de información del lenguaje del archivo de entrada

public Lenguajes generadorLenguaje(Arbol arbol, Nodo add, ReglasGramaticas rg, InfLenguaje inf, String actionCode)

Genera un nuevo lenguaje agregando los elementos necesarios como los son el árbol, reglas gramáticas

public void addTipo(Object e)

Agrega el tipo de dato a las terminales y no terminales del archivo de entrada.

public void agregarTNT(List<String> lista, boolean terminal, String aux, int left, int right, List<NoTerminal> listaNT, List<Terminal> listaT, int contadorLVL)

Agrega un nuevo terminal y no terminal a la lista de terminales y no terminales

public Object retornarTNT(List<String> comprobar, boolean terminal, String aux, int left, int right)

Construye una terminal o no terminal comprobando que no este definada.

public Object construirSigPro(Object i, String valorDeVuelto, Object d, int left, int right)

Método que ayuda a construir una producción en tiempo de análisis.

public void addRG(Object noTerminal, List<NoTerminal> lnt, String ntS, int left, int right, String rs)

Crea una nueva producción junto a sus terminales y no terminales,

Objetos:

- ConjuntoSiguientes
- DiagramaAS
- InfLenguaje
- Lenguajes
- Pestania
- TablaSiguientes
- TablaSimbolos
- AnalizadorLexico
- Arbol
- ExpresionRegular
- Nodo
- TablaTransiciones
- Token
- CaracteresColumnas
- Destado
- Estado
- EstadoFilas
- Goto
- NoTerminal
- Pila
- Produccion
- ReglasGramaticas
- Review
- Shift
- Simbolo
- TNT
- TablaLALR
- Terminal
- Tupla
- Uniones

NOTA: Cada uno con sus respectivo constructor, getter's y setter's.

Analizador Léxico y Sintáctico

- Lexer.flex
- Sintax.cupo
- Sintax.java
- Sym.java
- Lexico,java

Librerías:

- Jflex.jar
- java-cup-11b-runtime.jar

- java-cup-11b.ja

- Diseño Absoluto – AbsoluteLayout.jar
- Janino 3.04.jar
- beansbinding.jar

REQUISITOS MINIMOS PARA PROGRAMAR EN JAVA:
REQUERIMIENTOS DE HARDWARE PARA PROGRAMAR EN JAVA

Para las aplicaciones generadas se debe tener un mínimo de 32MB de RAM, se recomienda que se tengan 48MB o más. Para compilar utilizando el SDK de Microsoft es un mínimo de 32MB de RAM, y para JDK 48MB mínimo. El procesador en principio no es tan crítico como la memoria RAM, pero se recomienda utilizar al menos un Pentium de 133 para compilar/ejecutar las aplicaciones.

Requerimientos de SOFTWARE
Para trabajar con el Generador Java es necesario cierto software adicional para la compilación y ejecución de las aplicaciones generadas.

Java Development Kit - Compilador y Máquina Virtual
Para la compilación y ejecución de los programas es necesario tener el JDK. En cuanto a las versiones se recomienda en general utilizar las últimas liberadas de cada uno de ellos. En caso de que alguna versión sea menos estable que su versión anterior, se avisara.

Se requieren las versiones siguientes de .NET Framework:

SQL Server 2008 en Windows Server 2003 64 bits IA64: .NET Framework 2.0 SP2SQL Server Express: .NET Framework 2.0 SP2. Todas las demás ediciones de SQL Server 2008: .NET Framework 3.5 SP1, etc.

I**DE NetBeans 8.2 requisitos mínimos:**

Microsoft Windows Vista SP1 / Windows 7 Professional:
Procesador: Varel Pentium III 800MHz o equivalente
Memoria: 512 MB
Espacio en disco: 750 MB de espacio libre en disco

Ubuntu 9.10:
Procesador: Varel Pentium III 800MHz o
equivalente Memoria: 512 MB
Espacio en disco: 650 MB de espacio libre en disco

Macvarosh OS X 10.7
Varel: Procesador: Varel
Dual-Core Memoria: 2 GB
Espacio en disco: 650 MB de espacio libre en disco

Microsoft Windows 7 Professional / Windows 8 / Windows 8.2 / Windows 10:
Procesador: Varel Core i5 o
equivalente Memoria: 2 GB 32 bits,
4GB 64 bits
Espacio en disco: 1,5 GB de espacio libre en disco

Ubuntu 15.04 y Superior:
Procesador: Varel Core i5 o
equivalente Memoria: 2 GB 32 bits, 4
GB 64 bits
Espacio en disco: 1,5 GB de espacio libre en disco

OS X 10.10 Varel:
Procesador: Varel Dual-
Core Memoria: 4 GB
Espacio en disco: 1,5 GB de espacio libre en disco