

**Colegio Mexiquense Universitario**

**Ingeniería en Sistemas Computacionales**

**Manual Técnico: Analizador Léxico**

**Docente: Vianey Soltero Pérez**

**Alumnos: Axel Isaac Cejudo Román**

**5to Semestre**

**CONTENIDO**

[**Objetivos** 4](#_Toc54097873)

[✓ Realizar un analizador léxico del leguaje C+++ en NetBeans. 4](#_Toc54097874)

[✓ Aplicar conocimientos aprendidos durante el semestre. 4](#_Toc54097875)

[✓ Usar medios de investigación para facilitar la construcción del programa ya sea libros, audiolibros, navegadores web. 4](#_Toc54097876)

[**Instalación de NetBeans IDE 8.2** 5](#_Toc54097877)

[Requisitos de software 5](#_Toc54097878)

[Opciones de descarga del instalador 5](#_Toc54097879)

[Personalización de la instalación 5](#_Toc54097880)

[Inicio de la descarga 6](#_Toc54097881)

[Instalación del software 6](#_Toc54097882)

[Microsoft Windows, Solaris OS y Linux 6](#_Toc54097883)

[Para instalar el software: Una vez finalizada la descarga, ejecute el instalador. 6](#_Toc54097884)

[Mac OS X 8](#_Toc54097885)

[Paquete independiente de la plataforma 8](#_Toc54097886)

[**Pseudocodigo** 9](#_Toc54097887)

[**Diagramas** 10](#_Toc54097888)

[**Archivos del Proyecto** 11](#_Toc54097889)

[Descargar Proyecto 11](#_Toc54097890)

[Abrir Proyecto 12](#_Toc54097891)

[Añadir Librería 13](#_Toc54097892)

[Ejecutar Programa 13](#_Toc54097893)

[**Analizador Léxico** 14](#_Toc54097894)

[Funcionamiento 14](#_Toc54097895)

[Librería JFlex 14](#_Toc54097896)

[Archivos 14](#_Toc54097897)

[**Código Fuente** 15](#_Toc54097898)

[Lexer.flex 15](#_Toc54097899)

[Menú.java 18](#_Toc54097900)

[Lexer.java 24](#_Toc54097901)

[LineNumber.java 41](#_Toc54097902)

[TableData.java 47](#_Toc54097903)

[TableRender.java 47](#_Toc54097904)

[Tokens.java 48](#_Toc54097905)

# **Objetivos**

## ✓ Realizar un analizador léxico del leguaje C+++ en NetBeans.

## ✓ Aplicar conocimientos aprendidos durante el semestre.

## ✓ Usar medios de investigación para facilitar la construcción del programa ya sea libros, audiolibros, navegadores web.

# **Instalación de NetBeans IDE 8.2**

## Requisitos de software

#### Antes de instalar el IDE, debe tener instalado en el sistema la actualización 19 de Java SE Development Kit (JDK) (versión 1.5.0\_19) o la actualización 14 de JDK 6 o posterior. Si no tiene una instalación de JDK, no puede continuar con la instalación. Puede descargar la versión más reciente de JDK en [http://java.sun.com/javase/downloads.](http://java.sun.com/javase/downloads)

#### Para las opciones de descarga de PHP, Ruby y C/C++ puede instalar JRE en lugar de JDK. Sin embargo, si piensa usar cualquiera de las funciones de Java, necesitará JDK.

## Opciones de descarga del instalador

#### En la página de descarga de NetBeans IDE puede obtener uno de los diversos instaladores, cada uno de los cuales contiene el IDE básico y herramientas adicionales.

#### Los siguientes instaladores están disponibles:

#### Java. Proporciona herramientas para desarrollar aplicaciones de Java SE, Java EE y Java ME. Esta opción de descarga también incluye el software GlassFish v3 y Apache Tomcat 6.0.20.

#### Java SE. Proporciona todas las características estándar para el desarrollo de Java SE. También se incluye la compatibilidad con el desarrollo de RCP de la plataforma NetBeans.

#### Ruby. Proporciona herramientas para el desarrollo de Ruby y es compatible con Rails y JRuby. También contiene GlassFish v3.

#### PHP. Proporciona herramientas para el desarrollo de PHP.

#### C/C++. Es compatible con el desarrollo en los lenguajes C, C++, Qt, Fortran y Assembly.

#### All (Todo). Es una opción de descarga completa que contiene todos los tiempos de ejecución y los paquetes disponibles para NetBeans IDE.

## Personalización de la instalación

#### Si realiza la descarga mediante las opciones Java o "All" (Todo), puede seleccionar qué herramientas y tiempos de ejecución que desea instalar. Haga clic en Personalizar en la página de bienvenida del instalador y seleccione las características que desee usar. Para obtener más información, consulte Instalación del software.

#### Si desea agregar funciones a la configuración del NetBeans IDE, utilice NetBeans Plugin Manager. Por ejemplo, suponga que descarga e instala el paquete de Ruby. A continuación, decide que desea probar las funciones de C/C++. Para ello, hay que acceder a NetBeans Plugin Manager desde el IDE (seleccione Herramientas > Plugins) y agregar el paquete C/C++ a la instalación existente.

#### Hay varias notas importantes sobre la instalación y la configuración de algunas funciones de NetBeans IDE:

#### Apache Tomcat 6.0.20 se incluye en las opciones de descarga "Java" y "All" (Todo), pero no se instala de forma predeterminada con ninguna de ellas. Para instalar Apache Tomcat a partir de estas descargas, inicie el instalador y seleccione Apache Tomcat 6.0.20 en el cuadro de diálogo Personalizar la instalación.

#### Para el desarrollo de C/C++, aplicable a los tipos de descarga "C/C++" y "All" (Todo), deberá instalar además compiladores y herramientas. Para obtener más información, consulte Configuración de NetBeans IDE para C/C++/Fortran.

#### En Mac OS X, puede personalizar la instalación de NetBeans IDE en las opciones de descarga de Java, Ruby, y "All" (Todo). Esta posibilidad no está disponible en las descargas JavaSE, PHP y C/C++.

#### También puede descargar un archivo comprimido independiente de la plataforma y ejecutar el archivo ejecutable de NetBeans IDE. Tenga en cuenta que los tiempos de ejecución, como los servidores de aplicaciones de GlassFish y Apache Tomcat, sólo están disponibles con instaladores para plataformas específicas.

## Inicio de la descarga

#### Vaya a la página de descargas de NetBeans IDE.

#### En la parte superior derecha de la página, seleccione el idioma y la plataforma en la lista desplegable. También puede optar por descargar y utilizar el archivo zip independiente de la plataforma.

#### Haga clic en el botón Download (Descargar) de la opción de descarga que desee instalar. Guarde el archivo del instalador en el sistema.

## Instalación del software

#### Siga las instrucciones que aparecen en esta sección para instalar el IDE en el sistema. Estas instrucciones de instalación se aplican a todas las plataformas compatibles. Para obtener la lista de las plataformas compatibles, consulte las notas de la versión.

## Microsoft Windows, Solaris OS y Linux

## Para instalar el software: Una vez finalizada la descarga, ejecute el instalador.

#### El archivo del instalador de Windows tiene la extensión .exe. Haga doble clic en el para ejecutarlo.

#### El archivo del instalador de las plataformas Solaris y Linux tiene la extensión .sh. En estas plataformas debe convertir los archivos del instalador en ejecutables mediante el siguiente comando: chmod +x <nombre-archivo-instalador>

#### Para seleccionar las herramientas y tiempos de ejecución que se van a instalar, siga estos pasos en la página de bienvenida del asistente de instalación:

#### Haga clic en Personalizar.

#### En el cuadro de diálogo Personalizar la instalación, realice las selecciones. Haga clic en Aceptar.

#### En la página de bienvenida del asistente para la instalación, haga clic en Siguiente.

#### En la página del contrato de licencia, revise el contrato, marque la casilla de aceptación y haga clic en Siguiente.

#### En la página de instalación de NetBeans IDE, siga estos pasos:

#### Acepte el directorio de instalación predeterminado de NetBeans IDE o especifique otro directorio. Nota: el directorio de instalación debe estar vacío y el perfil de usuario que utilice para ejecutar el instalador debe disponer de permisos de lectura/escritura en dicho directorio.

#### Acepte la instalación de JDK predeterminada para usarlo con NetBeans IDE o seleccione una instalación diferente en la lista desplegable. Si el asistente de instalación no encuentra una instalación de JDK compatible para usar con NetBeans IDE, no se instalará en la ubicación predeterminada. En este caso, especifique la ruta de un JDK instalado o cancele la instalación actual, instale la versión de JDK requerida y reinicie esta instalación.

#### Haga clic en Siguiente.

#### Si se abre la página de instalación de GlassFish v3, acepte el directorio de instalación predeterminado o especifique otra ubicación para la instalación.

#### Si va a instalar Apache Tomcat: en la página de instalación, acepte el directorio de instalación predeterminado o especifique otra ubicación para la instalación.

#### En la página de resumen, compruebe que la lista de componentes que se van a instalar es correcta y que dispone de espacio suficiente en el sistema para la instalación.

#### Haga clic en Instalar para comenzar la instalación.

#### Nota: si surgen problemas para instalar el software correctamente, consulte Resolución de problemas para obtener descripciones y sugerencias para solucionar problemas no resueltos que pueden afectar al proceso de instalación.

## Mac OS X

#### Para instalar el software: Una vez finalizada la descarga, ejecute el instalador. La extensión del archivo del instalador es .dmg.

#### En los paneles que se abren, haga clic en el icono del paquete. La extensión del paquete es. mpkg. El asistente de instalación comienza. En la página de bienvenida del asistente para la instalación, haga clic en Continuar.

#### Revise el contrato de licencia y haga clic en Continuar. Haga clic en Aceptar en la ventana emergente para aceptar la licencia.

#### En la página de selección de destino, elija la unidad y haga clic en Continuar.

#### Si ha descargado los paquetes de "All" (Todo), Java o Ruby, puede personalizar la instalación. En el último panel del asistente de instalación, pulse el botón Personalizar de la parte inferior izquierda del panel. Aparece el árbol de productos.

#### Seleccione los productos que desea instalar.

#### Especifique el nombre y la contraseña del administrador del sistema y haga clic en Aceptar para comenzar la instalación.

## Paquete independiente de la plataforma

#### Para ejecutar el software:

#### Cuando finalice la descarga del archivo ZIP independiente de la plataforma, extráigalo a cualquier carpeta del sistema.

#### Ejecute el archivo ejecutable ubicado en el directorio netbeans/bin. Acepte el contrato de licencia. NetBeans IDE se inicia.

#### Nota: tenga en cuenta que los tiempos de ejecución, como el servidor de aplicaciones de GlassFish y Apache Tomcat, sólo están disponibles con instaladores para plataformas específicas.

#### Nota: si no puede encontrar la instalación de JDK compatible, quizá deba agregar la ruta del directorio de instalación de JDK de forma manual:

#### Abra el archivo netbeans.conf que se encuentra en netbeans/etc en un editor de texto.

#### Introduzca la ubicación de la instalación de JDK compatible como valor de la opción netbeans\_jdkhome.

#### Guarde el archivo netbeans.conf y ejecute el archivo ejecutable de netbeans/bin

# **Pseudocodigo**

1. Funcion tokens <- analizador ( codigo,lexemas )
2. caracter <- codigo
3. si caracter == regla\_lexica Entonces
4. token <- regla\_lexica\_token
5. FinSi
6. Fin Funcion
8. Algoritmo Analizador\_Lexico
9. // El usuario decide que hacer
10. Escribir "De sea escribir el codigo 1 o cargar un archivo"
11. Leer entero opcionn
12. Segun opcionn Hacer
13. 1:
14. // Se lee el codigo de un textarea linea por linea
15. Leer Texto codigo
16. Mientras codigo <> null
17. // Analiza y retorna los tokens
18. Escribir analizador(codigo, lexema)
19. FinMientras
20. 2:
21. // se lee un archivo linea por linea
22. Escribir "Abrir explorador de archivos"
23. Leer Archivo
24. Mientras Archivo <> null
25. // Analiza y retorna los tokens
26. Escribir analizador(codigo, lexema)
27. FinMientras
28. De Otro Modo:
29. Escribir "Elija que hacer"
30. Fin Segun
31. FinAlgoritmo

# 

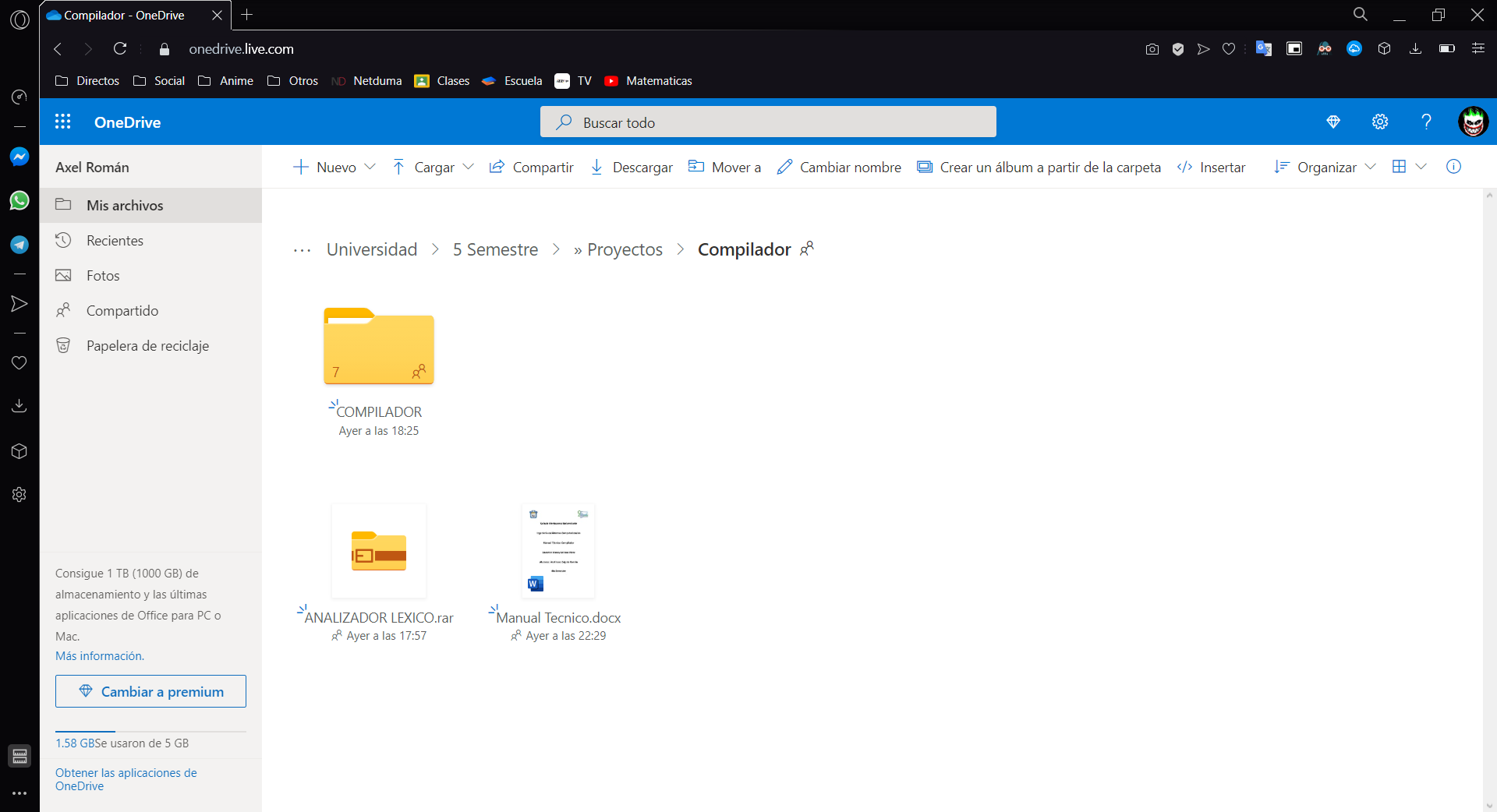
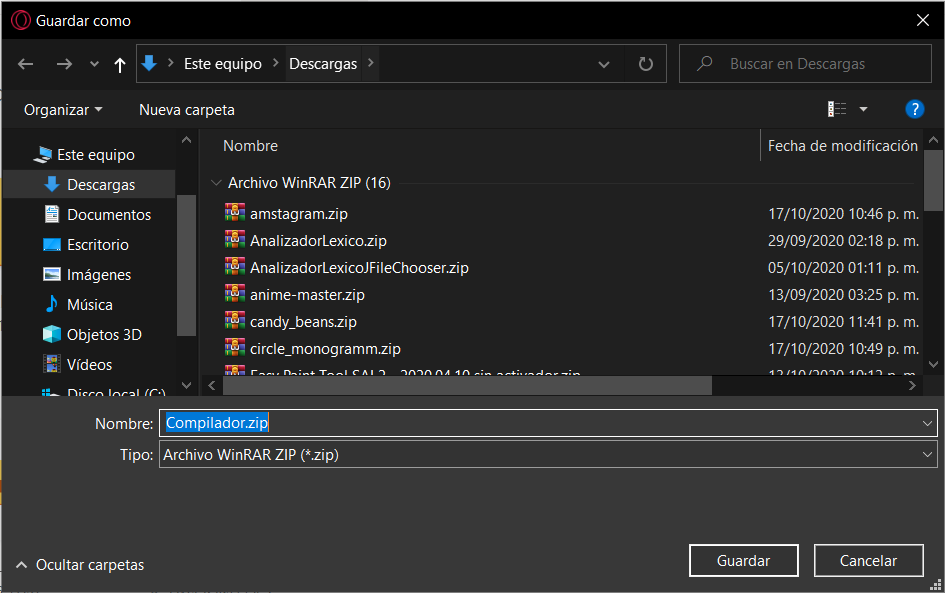
# **Diagramas**

## Analizador de código o archivo

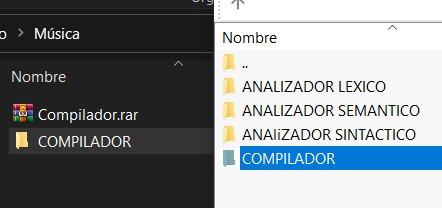
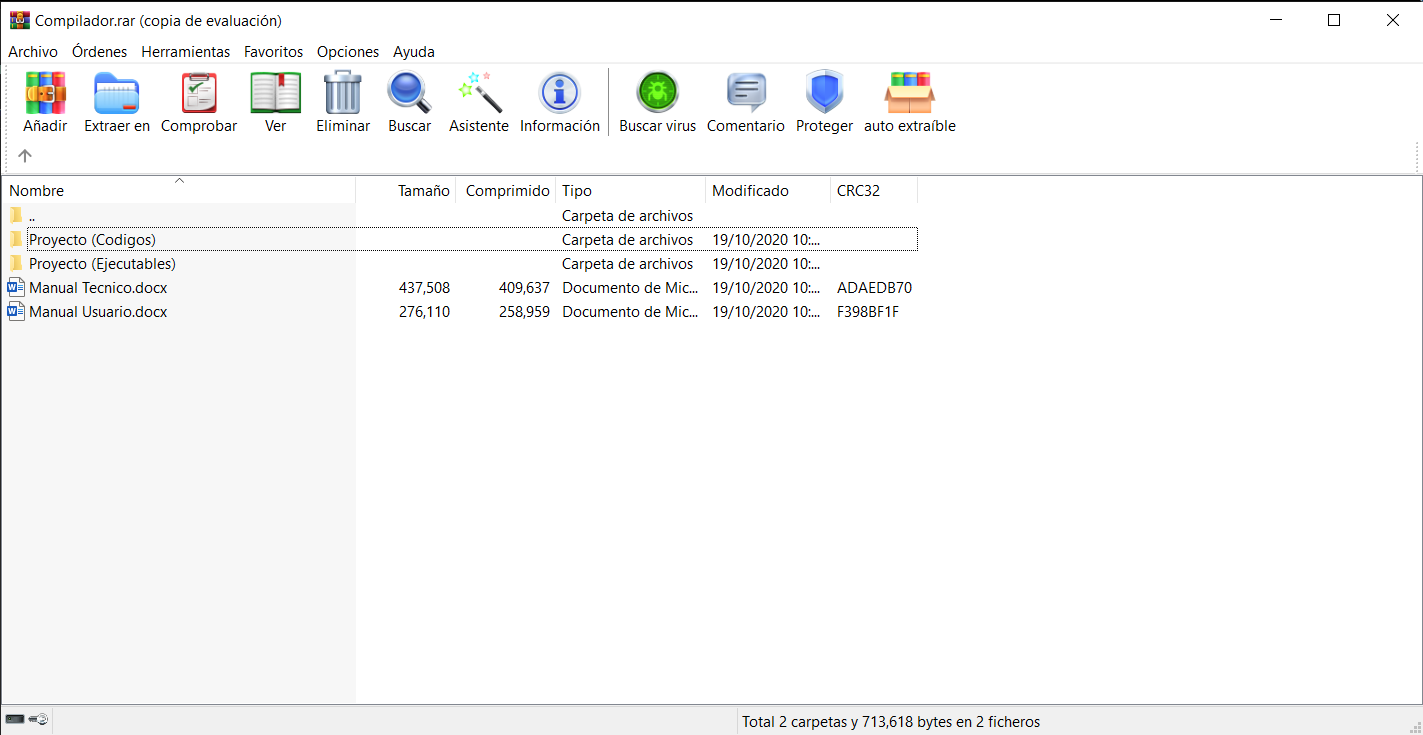
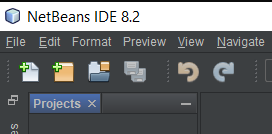
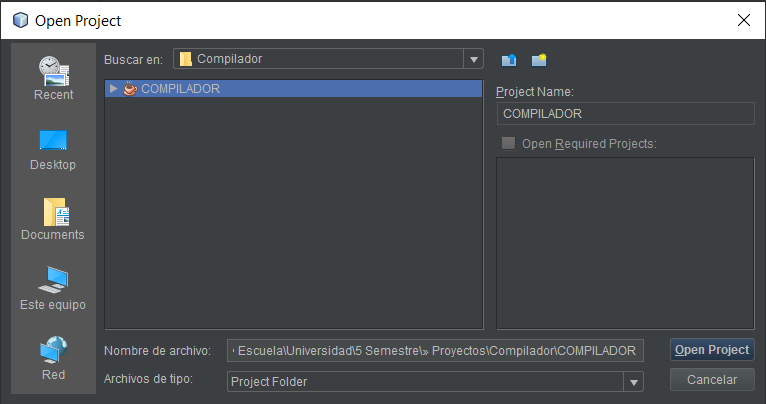
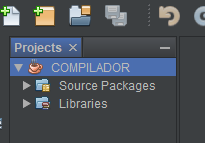
## Analizador de tokens encontrados

# **Archivos del Proyecto**

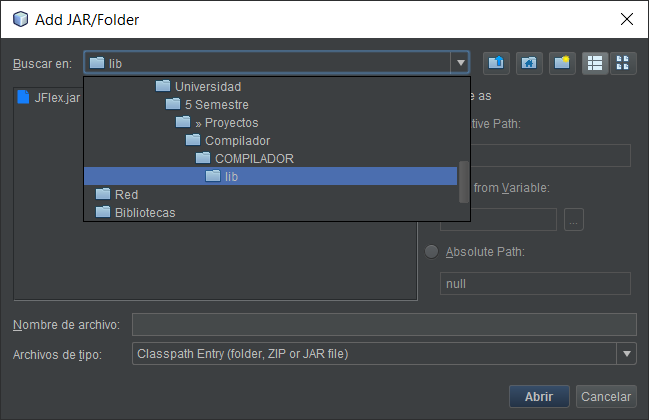
## Descargar Proyecto

1. Abra el enlace donde esta guardado el archivo del proyecto: <https://1drv.ms/u/s!AkReNdQdhFHO5mlt8OlIPKhQ-vHX?e=SHMs72>
2. De un clic en la parte donde indica **Descargar** en la parte superior
3. Elija una ubicación para guardar el archivo y listo.

## Abrir Proyecto

1. Abra el archivo .rar descargado previamente:
2. Seleccione y abra la carpeta de nombre **Proyecto (Códigos)** dentro de la carpeta estará el proyecto con el nombre **COMPILADOR**, selecciónelo y arrástrelo en la carpeta de **NetBeans** o en el lugar donde se prefiera (Los demás archivos son partes por separado del proyecto).
3. Abra el programa de **NetBeans** y de un clic en siguiente icono:
4. Seleccione la ubicación del proyecto, ábralo y listo.

## Añadir Librería

1. De clic derecho en la parte del proyecto que dice **Libraries.**
2. Seleccione la opción **Add JAR/Folder…** y busque dentro de la carpeta del proyecto una carpeta de nombre **lib** ahí se encontrará las librerías que necesita el programa, selecciónelas todas y agréguelas.

## Ejecutar Programa



1. Seleccione el icono siguiente o presione la tecla **F6** de su teclado para ejecutarlo y listo.
2. En caso de algún error y no se cierre adecuadamente el programa puede cerrar el proceso en la parte inferior izquierda dándole clic al tache de la barra de progreso.

# **Analizador Léxico**

## Funcionamiento

Es la primera fase de un compilador y su principal función consiste en leer los caracteres de entrada y elaborar como salida una secuencia de componente léxicos que utiliza el analizador sintáctico para hacer el análisis. Esta interacción suele aplicarse convirtiendo al analizador léxico en una subrutina o corrutina del analizador sintáctico.

## Librería JFlex

Es un generador de analizador léxico escrito en Java. Un generador de analizador léxico toma como entrada una especificación con un conjunto de expresiones regulares y acciones correspondientes.

Genera un programa (un lexer) que lee la entrada, compara la entrada con las expresiones regulares en el archivo de especificaciones y ejecuta la acción correspondiente si una expresión regular coincide.

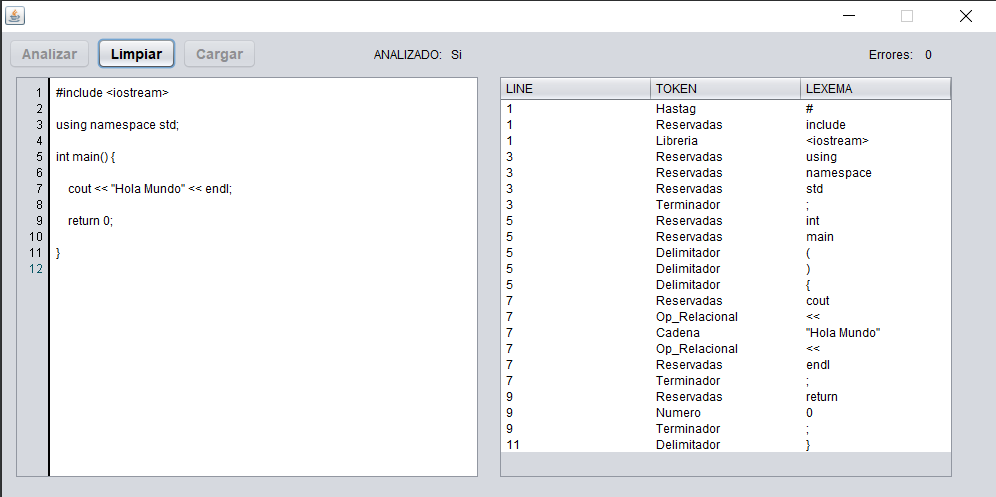
Los lexers JFlex se basan en autómatas finitos deterministas (DFA). Son rápido sin costosos retrocesos.

Documentación: <https://jflex.de/manual.pdf>

## Archivos

Es la parte inicial del proyecto del compilador guardado por separado en caso de necesitarlo o analizar el código fuente directamente.

Analizador Léxico: https://1drv.ms/u/s!AkReNdQdhFHO5yXE5sI6t2MecKYP?e=IzS8Jz

****

# **Código Fuente**

## Lexer.flex

1. /\*
2. \* ARCHIVO DONDE SE ESCRIBEN TODAS LAS REGLAS LEXICAS
3. \*/
5. /\* Ubicacion del paquete donde se creara el archivo .flex \*/
6. package code;
8. /\* Nombre del metodo que se usara para importarlo\*/
9. import static code.Tokens.\*;
11. /\* Separador del archivo \*/
12. %%
14. /\* Nombre del archivo .flex \*/
15. %class Lexer
17. /\* Asignamos el tipo de valor que manejara el archivo \*/
18. %type Tokens
20. /\* Linea del lexema que se este analizando \*/
21. %line
23. /\* Se define una variable para el argumento de detectar letras \*/
24. L=[a-zA-Z\_]+
26. /\* Se define una variable para el argumento de detectar numeros \*/
27. D=[0-9]+
29. /\* Se define una variable para el argumento de detectar librerias con .h \*/
30. lib = {L}(".h")
32. /\* Se define una variable para el argumento de detectar algun salto de linea, tabulacion, etc \*/
33. espacio=[ ,\t,\r,\n]+
35. /\* Se define una variable que almacenara todos los datos del lexema y tokens junto a un archivo que
36. guardara todas las ubicaciones de en que linea se encuentran los lexemas \*/
37. %{
38. TableData data = new TableData();
39. %}
41. /\* Separador del archivo \*/
42. %%
44. /\* Regla que ignora algunos caracteres y la variable ya definida \*/
45. {espacio} {/\*Ignore\*/}
46. "//".\* {/\*Ignore\*/}
48. /\* Regla que detecta todas las palabras reservadas de la lista \*/
49. asm | auto | break | case | catch | class |
50. const | const\_cast | continue | default | delete | do |
51. dynamic\_cast | else | enum | explicit | extern | false |
52. for | friend | goto | if | inline | mutable |
53. namespace | new | operator | private | protected | public |
54. register | reinterpret\_cast | return | signed | sizeof | static |
55. static\_cast | struct | switch | template | this | throw |
56. true | try | typedef | typeid | typename | union |
57. unsigned | using | virtual | void | volatile | while |
58. printf | scanf | include | iostream | stdio | endl |
59. cout | cin | main | std
60. {data.linea=yyline; data.lexema=yytext(); return Reservadas;}
62. /\* Regla para detectar los tipos de datos \*/
63. bool | boolean | char | double | float | int | long | short
64. {data.linea=yyline; data.lexema=yytext(); return T\_Dato;}
66. /\* Regla para detectar las librerias del codigo \*/
67. ("<"){lib}(">") | ("<"){L}(">") {data.linea=yyline; data.lexema=yytext(); return Libreria;}
69. /\* Regla para detectar los identificadores \*/
70. {L}({L}|{D})\* {data.linea=yyline; data.lexema=yytext(); return Identificador;}
72. /\* Regla para detectar numeros enteros \*/
73. ("(-"{D}+")")|{D}+ {data.linea=yyline; data.lexema=yytext(); return N\_Entero;}
75. /\* Regla para detectar numeros decimales \*/
76. {D}("."){D} {data.linea=yyline; data.lexema=yytext(); return N\_Decimal;}
78. /\* Regla para detectar cadenas de texto \*/
79. ("\"")([^\n\"]\*([.])\*)\*("\"") {data.linea=yyline; data.lexema=yytext(); return Cadena;}
81. /\* Regla para detectar caracteres \*/
82. ("'"){L}("'") | ("'").("'") {data.linea=yyline; data.lexema=yytext(); return Caracter;}
84. /\* Regla para detectar operadores aritmeticos \*/
85. "+" | "-" | "\*" | "/" {data.linea=yyline; data.lexema=yytext(); return OpAritmeticos;}
87. /\* Regla para detectar delimitadores de codigo \*/
88. "(" {data.linea=yyline; data.lexema=yytext(); return Parentesis\_a;}
89. ")" {data.linea=yyline; data.lexema=yytext(); return Parentesis\_c;}
90. "{" {data.linea=yyline; data.lexema=yytext(); return Llave\_a;}
91. "}" {data.linea=yyline; data.lexema=yytext(); return Llave\_c;}
92. "[" {data.linea=yyline; data.lexema=yytext(); return Corchetes\_a;}
93. "]" {data.linea=yyline; data.lexema=yytext(); return Corchetes\_c;}
95. /\* Regla para detectar ciertos caracteres especiales \*/
96. "\n" {data.linea=yyline; data.lexema=yytext(); return Linea;}
97. "°" {data.linea=yyline; data.lexema=yytext(); return Grado;}
98. "¡" {data.linea=yyline; data.lexema=yytext(); return Admiracion;}
99. "¨" {data.linea=yyline; data.lexema=yytext(); return dieresis;}
100. "#" {data.linea=yyline; data.lexema=yytext(); return Hastag;}
101. "´" {data.linea=yyline; data.lexema=yytext(); return Acento;}
102. "`" {data.linea=yyline; data.lexema=yytext(); return AcentoGrave;}
103. "&" {data.linea=yyline; data.lexema=yytext(); return And;}
104. "." {data.linea=yyline; data.lexema=yytext(); return Punto;}
105. ":" {data.linea=yyline; data.lexema=yytext(); return DosPuntos;}
106. "%" {data.linea=yyline; data.lexema=yytext(); return Porcentaje;}
107. "\\" {data.linea=yyline; data.lexema=yytext(); return BarraInvertida;}
108. "~" {data.linea=yyline; data.lexema=yytext(); return Virgulilla;}
109. "@" {data.linea=yyline; data.lexema=yytext(); return Arroba;}
110. "$" {data.linea=yyline; data.lexema=yytext(); return Dinero;}
111. "¿" {data.linea=yyline; data.lexema=yytext(); return Interrogacion\_a;}
112. "?" {data.linea=yyline; data.lexema=yytext(); return Interrogacion\_c;}
113. "'" {data.linea=yyline; data.lexema=yytext(); return ComillaSimple;}
114. ";" {data.linea=yyline; data.lexema=yytext(); return OpFin;}
115. "=" {data.linea=yyline; data.lexema=yytext(); return OpAsignacion;}
116. ("\"") {data.linea=yyline; data.lexema=yytext(); return Comilla;}
118. /\* Regla para detectar operadores relacionales \*/
119. ">" | "<" | "==" | "!=" | ">=" | "<=" | "<<" | ">>" {data.linea=yyline; data.lexema=yytext(); return OpRelacional;}
121. /\* Regla para detectar operadores logicos \*/
122. "&&" | "||" | "!" | "&" | "|" | "^" | "¬" {data.linea=yyline; data.lexema=yytext(); return OpLogico;}
124. /\* Regla para detectar operadores de atribucion \*/
125. "+=" | "-=" | "\*=" | "/=" | "%=" {data.linea=yyline; data.lexema=yytext(); return OpAtribucion;}
127. /\* Regla para detectar operadores de incremento y decremento \*/
128. "++" | "--" {data.linea=yyline; data.lexema=yytext(); return OpIncremento;}
130. /\* Regla para detectar cualquier otro caracter fuera de las reglas no definido \*/
131. . {data.linea=yyline; data.lexema=yytext(); return ERROR;}

## Menú.java

1. /\*
2. ARCHIVO PRINCIPAL
3. \*/
5. /\* Paquete \*/
6. package code;
8. /\* Librerias \*/
9. import java.io.\*;
10. import javax.swing.JFileChooser;
11. import javax.swing.table.DefaultTableModel;
13. /\* Clase y Extenciones \*/
14. public class Menu extends javax.swing.JFrame{
16. /\* Constructor \*/
17. public Menu() {
18. /\* Valores iniciales de los objetos \*/
19. initComponents();
20. /\* No rescalable \*/
21. setResizable(false);
22. /\* Ubicacion central \*/
23. setLocationRelativeTo(null);
24. /\* Asigna el nuevo diseño al jScrollPanel del TextArea \*/
25. LineNumber desing = new LineNumber(txtCode);
26. ScrollPanel\_txtCode.setRowHeaderView(desing);
27. /\* Analiza y Crea el archivo lexer \*/
28. GenerateLexer();
29. }
31. /\* Valores Iniciales de los Objetos \*/
32. @SuppressWarnings("unchecked")
33. // <editor-fold defaultstate="collapsed" desc="Generated Code">
34. private void initComponents() {
36. btnCargar = new javax.swing.JButton();
37. btnAnalizar = new javax.swing.JButton();
38. btnLimpiar = new javax.swing.JButton();
39. labelTokens = new javax.swing.JLabel();
40. jLabel3 = new javax.swing.JLabel();
41. txtState = new javax.swing.JLabel();
42. ScrollPanel\_tableInfo = new javax.swing.JScrollPane();
43. tableInfo = new javax.swing.JTable();
44. ScrollPanel\_txtCode = new javax.swing.JScrollPane();
45. txtCode = new javax.swing.JTextArea();
46. labelErrores = new javax.swing.JLabel();
47. txtErrores = new javax.swing.JLabel();
49. setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);
50. setPreferredSize(new java.awt.Dimension(500, 500));
52. btnCargar.setFont(new java.awt.Font("Dialog", 1, 14)); // NOI18N
53. btnCargar.setText("Cargar");
54. btnCargar.addActionListener(new java.awt.event.ActionListener() {
55. public void actionPerformed(java.awt.event.ActionEvent evt) {
56. btnCargarActionPerformed(evt);
57. }
58. });
60. btnAnalizar.setFont(new java.awt.Font("Dialog", 1, 14)); // NOI18N
61. btnAnalizar.setText("Analizar");
62. btnAnalizar.addActionListener(new java.awt.event.ActionListener() {
63. public void actionPerformed(java.awt.event.ActionEvent evt) {
64. btnAnalizarActionPerformed(evt);
65. }
66. });
68. btnLimpiar.setFont(new java.awt.Font("Dialog", 1, 14)); // NOI18N
69. btnLimpiar.setText("Limpiar");
70. btnLimpiar.addActionListener(new java.awt.event.ActionListener() {
71. public void actionPerformed(java.awt.event.ActionEvent evt) {
72. btnLimpiarActionPerformed(evt);
73. }
74. });
76. labelTokens.setToolTipText("");
77. labelTokens.setEnabled(false);
79. jLabel3.setText("ANALIZADO: ");
80. jLabel3.setToolTipText("");
82. txtState.setText("No");
83. txtState.setToolTipText("");
85. tableInfo.setModel(new javax.swing.table.DefaultTableModel(
86. new Object [][] {
87. {null, null, null, null},
88. {null, null, null, null},
89. {null, null, null, null},
90. {null, null, null, null}
91. },
92. new String [] {
93. "Title 1", "Title 2", "Title 3", "Title 4"
94. }
95. ));
96. ScrollPanel\_tableInfo.setViewportView(tableInfo);
98. txtCode.setColumns(20);
99. txtCode.setRows(5);
100. ScrollPanel\_txtCode.setViewportView(txtCode);
102. labelErrores.setText("Errores:");
103. labelErrores.setToolTipText("");
105. txtErrores.setText("No");
106. txtErrores.setToolTipText("");
108. javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
109. getContentPane().setLayout(layout);
110. layout.setHorizontalGroup(
111. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
112. .addGroup(layout.createSequentialGroup()
113. .addContainerGap()
114. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
115. .addGroup(layout.createSequentialGroup()
116. .addGap(88, 88, 88)
117. .addComponent(labelTokens))
118. .addGroup(layout.createSequentialGroup()
119. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
120. .addGroup(layout.createSequentialGroup()
121. .addComponent(btnAnalizar)
122. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
123. .addComponent(btnLimpiar)
124. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
125. .addComponent(btnCargar)
126. .addGap(117, 117, 117)
127. .addComponent(jLabel3)
128. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
129. .addComponent(txtState, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE))
130. .addGroup(layout.createSequentialGroup()
131. .addGap(6, 6, 6)
132. .addComponent(ScrollPanel\_txtCode)))
133. .addGap(18, 18, 18)
134. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
135. .addComponent(ScrollPanel\_tableInfo, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
136. .addGroup(layout.createSequentialGroup()
137. .addComponent(labelErrores)
138. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
139. .addComponent(txtErrores, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)))))
140. .addContainerGap(19, Short.MAX\_VALUE))
141. );
142. layout.setVerticalGroup(
143. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
144. .addGroup(layout.createSequentialGroup()
145. .addContainerGap()
146. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
147. .addComponent(btnAnalizar)
148. .addComponent(btnLimpiar)
149. .addComponent(jLabel3)
150. .addComponent(txtState)
151. .addComponent(btnCargar)
152. .addComponent(labelErrores)
153. .addComponent(txtErrores))
154. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
155. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
156. .addComponent(ScrollPanel\_tableInfo, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE)
157. .addComponent(ScrollPanel\_txtCode, javax.swing.GroupLayout.DEFAULT\_SIZE, 412, Short.MAX\_VALUE))
158. .addGap(18, 18, 18)
159. .addComponent(labelTokens))
160. );
162. pack();
163. }// </editor-fold>
165. /\* Metodos del boton Analizar \*/
166. // <editor-fold defaultstate="collapsed" desc="Boton Analizar">
167. private void btnAnalizarActionPerformed(java.awt.event.ActionEvent evt) {
168. String text = txtCode.getText();
169. if(!text.equals("")){
170. int numTokens = 0;
171. txtState.setText("Si");
172. btnAnalizar.setEnabled(false);
173. labelTokens.setEnabled(true);
174. txtCode.setEditable(false);
175. this.setSize(1000, 500);
176. String url = "file.txt";
177. File file = new File(url);
178. PrintWriter writer;
179. try {
180. writer = new PrintWriter(file);
181. writer.print(txtCode.getText());
182. writer.close();
183. } catch (FileNotFoundException e) { System.err.println(e); }
184. try {
185. Reader reader = new BufferedReader(new FileReader(url));
186. Lexer lexer = new Lexer(reader);
187. String token = "";
188. int error=0;
189. DefaultTableModel model;
190. String com[] = {"LINE","TOKEN", "LEXEMA"};
191. String row[] = new String[3];
192. model = new DefaultTableModel(null, com);
193. while (true) {
194. TableData.token = lexer.yylex();
195. if(TableData.token != null){
196. row[0] = Integer.toString(1+TableData.linea);
197. switch (TableData.token) {
198. case ERROR:
199. row[1] = "NO DEFINIDO";
200. error++;
201. break;
202. default:
203. row[1] = "" + TableData.token;
204. break;
205. }
206. row[2] = TableData.lexema;
207. model.addRow(row);
208. } else {
209. break;
210. }
211. txtErrores.setText(Integer.toString(error));
212. tableInfo.setModel(model);
213. tableInfo.getColumnModel().getColumn(0).setCellRenderer(new TableRender());
214. tableInfo.getColumnModel().getColumn(1).setCellRenderer(new TableRender());
215. tableInfo.getColumnModel().getColumn(2).setCellRenderer(new TableRender());
216. }
217. } catch (Exception e) { System.err.println(e); }
218. }
220. }
221. // </editor-fold>
223. /\* Meotdos del boton limpiar \*/
224. // <editor-fold defaultstate="collapsed" desc="Boton Limpiar">
225. private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
226. txtCode.setText("");
227. txtState.setText("No");
228. btnAnalizar.setEnabled(true);
229. btnCargar.setEnabled(true);
230. labelTokens.setEnabled(false);
231. txtCode.setEditable(true);
232. this.setSize(500, 500);
233. }
234. // </editor-fold>
236. /\* Metodos del boton cargar \*/
237. // <editor-fold defaultstate="collapsed" desc="Boton Cargar">
238. private void btnCargarActionPerformed(java.awt.event.ActionEvent evt) {
239. JFileChooser chooser = new JFileChooser();
240. chooser.showOpenDialog(null);
241. File file = chooser.getSelectedFile();
242. try {
243. FileReader fr = new FileReader(file);
244. BufferedReader br = new BufferedReader(fr);
245. String text="";
246. String code="";
247. while ((code=br.readLine())!=null) {
248. text+=code+"\n";
249. }
250. txtCode.setText(text);
251. } catch (Exception e) { }
252. btnCargar.setEnabled(false);
253. btnAnalizar.setEnabled(true);
254. }
255. // </editor-fold>
257. /\* Metodo para generar el archivo lexer \*/
258. // <editor-fold defaultstate="collapsed" desc="Generate Lexer">
259. public static void GenerateLexer(){
260. File file = new File("src/code/Lexer.flex").getAbsoluteFile();
261. JFlex.Main.generate(file);
262. }// </editor-fold>
264. /\* Metodo Principal \*/
265. // <editor-fold defaultstate="collapsed" desc="Principal Main">
267. public static void main(String args[]) {
268. try {
269. for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
270. if ("Nimbus".equals(info.getName())) {
271. javax.swing.UIManager.setLookAndFeel(info.getClassName());
272. break;
273. }
274. }
275. } catch (ClassNotFoundException ex) {
276. java.util.logging.Logger.getLogger(Menu.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
277. } catch (InstantiationException ex) {
278. java.util.logging.Logger.getLogger(Menu.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
279. } catch (IllegalAccessException ex) {
280. java.util.logging.Logger.getLogger(Menu.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
281. } catch (javax.swing.UnsupportedLookAndFeelException ex) {
282. java.util.logging.Logger.getLogger(Menu.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
283. }
285. /\* Create and display the form \*/
286. java.awt.EventQueue.invokeLater(new Runnable() {
287. public void run() {
288. new Menu().setVisible(true);
289. }
290. });
291. }// </editor-fold>

294. // Variables declaration - do not modify
295. private javax.swing.JScrollPane ScrollPanel\_tableInfo;
296. private javax.swing.JScrollPane ScrollPanel\_txtCode;
297. private javax.swing.JButton btnAnalizar;
298. private javax.swing.JButton btnCargar;
299. private javax.swing.JButton btnLimpiar;
300. private javax.swing.JLabel jLabel3;
301. private javax.swing.JLabel labelErrores;
302. private javax.swing.JLabel labelTokens;
303. private static javax.swing.JTable tableInfo;
304. private javax.swing.JTextArea txtCode;
305. private javax.swing.JLabel txtErrores;
306. private javax.swing.JLabel txtState;
307. // End of variables declaration
309. }

## Lexer.java

(Este archive se genera solo con el archive Lexer.flex)

1. /\* The following code was generated by JFlex 1.4.3 on 20/10/20 02:28 PM \*/
3. /\*
4. \* ARCHIVO DONDE SE ESCRIBEN TODAS LAS REGLAS LEXICAS
5. \*/
7. /\* Ubicacion del paquete donde se creara el archivo .flex \*/
8. package code;
10. /\* Nombre del metodo que se usara para importarlo \*/
11. import static code.Tokens.\*;
13. /\* Separador del archivo \*/
15. /\*\*
16. \* This class is a scanner generated by
17. \* <a href="http://www.jflex.de/">JFlex</a> 1.4.3
18. \* on 20/10/20 02:28 PM from the specification file
19. \* <tt>C:/Users/Restf/OneDrive/• Escuela/Universidad/5 Semestre/» Proyectos/Analizador Lexico/ANALIZADOR LEXICO/src/code/Lexer.flex</tt>
20. \*/
21. class Lexer {
23. /\*\* This character denotes the end of file \*/
24. public static final int YYEOF = -1;
26. /\*\* initial size of the lookahead buffer \*/
27. private static final int ZZ\_BUFFERSIZE = 16384;
29. /\*\* lexical states \*/
30. public static final int YYINITIAL = 0;
32. /\*\*
33. \* ZZ\_LEXSTATE[l] is the state in the DFA for the lexical state l
34. \* ZZ\_LEXSTATE[l+1] is the state in the DFA for the lexical state l
35. \* at the beginning of a line
36. \* l is of the form l = 2\*k, k a non negative integer
37. \*/
38. private static final int ZZ\_LEXSTATE[] = {
39. 0, 0
40. };
42. /\*\*
43. \* Translates characters to character classes
44. \*/
45. private static final String ZZ\_CMAP\_PACKED =
46. "\11\0\1\5\1\7\2\0\1\5\22\0\1\5\1\65\1\45\1\52"+
47. "\1\63\1\57\1\54\1\46\1\42\1\44\1\50\1\47\1\5\1\43"+
48. "\1\3\1\6\12\2\1\56\1\53\1\40\1\55\1\41\1\64\1\62"+
49. "\32\1\1\51\1\60\1\51\1\67\1\25\1\0\1\10\1\16\1\23"+
50. "\1\27\1\21\1\30\1\34\1\4\1\26\1\1\1\22\1\17\1\12"+
51. "\1\24\1\15\1\33\1\1\1\20\1\11\1\14\1\13\1\36\1\35"+
52. "\1\32\1\31\1\37\1\51\1\66\1\51\1\61\uff81\0";
54. /\*\*
55. \* Translates characters to character classes
56. \*/
57. private static final char [] ZZ\_CMAP = zzUnpackCMap(ZZ\_CMAP\_PACKED);
59. /\*\*
60. \* Translates DFA states to action switch labels.
61. \*/
62. private static final int [] ZZ\_ACTION = zzUnpackAction();
64. private static final String ZZ\_ACTION\_PACKED\_0 =
65. "\1\0\1\1\1\2\1\3\1\4\1\5\1\6\23\2"+
66. "\2\7\1\10\1\6\1\10\1\11\1\12\2\6\1\13"+
67. "\1\14\1\15\1\16\1\17\1\20\1\21\1\22\1\23"+
68. "\1\24\1\25\3\26\1\5\1\27\40\2\2\30\13\2"+
69. "\1\0\1\7\1\0\1\31\1\0\1\32\2\0\4\2"+
70. "\1\30\52\2\1\0\1\33\1\0\1\34\37\2\1\0"+
71. "\1\3\7\2\1\30\34\2";
73. private static int [] zzUnpackAction() {
74. int [] result = new int[224];
75. int offset = 0;
76. offset = zzUnpackAction(ZZ\_ACTION\_PACKED\_0, offset, result);
77. return result;
78. }
80. private static int zzUnpackAction(String packed, int offset, int [] result) {
81. int i = 0; /\* index in packed string \*/
82. int j = offset; /\* index in unpacked array \*/
83. int l = packed.length();
84. while (i < l) {
85. int count = packed.charAt(i++);
86. int value = packed.charAt(i++);
87. do result[j++] = value; while (--count > 0);
88. }
89. return j;
90. }

93. /\*\*
94. \* Translates a state to a row index in the transition table
95. \*/
96. private static final int [] ZZ\_ROWMAP = zzUnpackRowMap();
98. private static final String ZZ\_ROWMAP\_PACKED\_0 =
99. "\0\0\0\70\0\160\0\250\0\70\0\340\0\u0118\0\u0150"+
100. "\0\u0188\0\u01c0\0\u01f8\0\u0230\0\u0268\0\u02a0\0\u02d8\0\u0310"+
101. "\0\u0348\0\u0380\0\u03b8\0\u03f0\0\u0428\0\u0460\0\u0498\0\u04d0"+
102. "\0\u0508\0\u0540\0\u0578\0\u05b0\0\u05e8\0\u0620\0\70\0\u0658"+
103. "\0\u0690\0\u06c8\0\u0700\0\70\0\70\0\u0738\0\u0770\0\70"+
104. "\0\u0700\0\70\0\70\0\70\0\70\0\70\0\u0770\0\u07a8"+
105. "\0\70\0\u07e0\0\70\0\u0818\0\u0850\0\u0888\0\u08c0\0\u08f8"+
106. "\0\u0930\0\u0968\0\u09a0\0\u09d8\0\u0a10\0\u0a48\0\u0a80\0\u0ab8"+
107. "\0\u0af0\0\u0b28\0\u0b60\0\u0b98\0\u0bd0\0\u0c08\0\u0c40\0\u0c78"+
108. "\0\u0cb0\0\u0ce8\0\u0d20\0\u0d58\0\u0d90\0\u0dc8\0\u0e00\0\u0e38"+
109. "\0\u0e70\0\u0ea8\0\u0ee0\0\160\0\u0f18\0\u0f50\0\u0f88\0\u0fc0"+
110. "\0\u0ff8\0\u1030\0\u1068\0\u10a0\0\u10d8\0\u1110\0\u1148\0\u1180"+
111. "\0\u11b8\0\70\0\u11f0\0\70\0\u0658\0\70\0\u1228\0\u1260"+
112. "\0\u1298\0\u12d0\0\u1308\0\u1340\0\u1378\0\u13b0\0\u13e8\0\u1420"+
113. "\0\u1458\0\u1490\0\u14c8\0\u1500\0\u1538\0\u1570\0\u15a8\0\u15e0"+
114. "\0\u1618\0\u1650\0\u1688\0\u16c0\0\u16f8\0\u1730\0\u1768\0\u17a0"+
115. "\0\u17d8\0\u1810\0\u1848\0\u1880\0\u18b8\0\u18f0\0\u1928\0\u1960"+
116. "\0\u1998\0\u19d0\0\u1a08\0\u1a40\0\u1a78\0\u1ab0\0\u1ae8\0\u1b20"+
117. "\0\u1b58\0\u1b90\0\u1bc8\0\u1c00\0\u1c38\0\u1c70\0\u1ca8\0\u1ce0"+
118. "\0\70\0\u1d18\0\70\0\u1d50\0\u1d88\0\u1dc0\0\u1df8\0\u1e30"+
119. "\0\u1e68\0\u1ea0\0\u1ed8\0\u1f10\0\u1f48\0\u1f80\0\u1fb8\0\u1ff0"+
120. "\0\u2028\0\u2060\0\u2098\0\u20d0\0\u2108\0\u2140\0\u2178\0\u21b0"+
121. "\0\u21e8\0\u2220\0\u2258\0\u2290\0\u22c8\0\u2300\0\u2338\0\u2370"+
122. "\0\u23a8\0\u23e0\0\u2418\0\70\0\u2450\0\u2488\0\u24c0\0\u24f8"+
123. "\0\u2530\0\u2568\0\u25a0\0\u25d8\0\u2610\0\u2648\0\u2680\0\u26b8"+
124. "\0\u26f0\0\u2728\0\u2760\0\u2798\0\u27d0\0\u2808\0\u2840\0\u2878"+
125. "\0\u28b0\0\u28e8\0\u2920\0\u2958\0\u2990\0\u29c8\0\u2a00\0\u2a38"+
126. "\0\u2a70\0\u2aa8\0\u2ae0\0\u25d8\0\u2b18\0\u2b50\0\u2b88\0\u2bc0";
128. private static int [] zzUnpackRowMap() {
129. int [] result = new int[224];
130. int offset = 0;
131. offset = zzUnpackRowMap(ZZ\_ROWMAP\_PACKED\_0, offset, result);
132. return result;
133. }
135. private static int zzUnpackRowMap(String packed, int offset, int [] result) {
136. int i = 0; /\* index in packed string \*/
137. int j = offset; /\* index in unpacked array \*/
138. int l = packed.length();
139. while (i < l) {
140. int high = packed.charAt(i++) << 16;
141. result[j++] = high | packed.charAt(i++);
142. }
143. return j;
144. }
146. /\*\*
147. \* The transition table of the DFA
148. \*/
149. private static final int [] ZZ\_TRANS = zzUnpackTrans();
151. private static final String ZZ\_TRANS\_PACKED\_0 =
152. "\1\2\1\3\1\4\1\5\1\3\1\6\1\7\1\6"+
153. "\1\10\1\11\1\12\1\13\1\14\1\15\1\16\1\17"+
154. "\1\20\1\21\1\3\1\22\1\23\1\3\1\24\1\25"+
155. "\1\26\2\3\1\27\1\30\1\31\1\32\1\3\1\33"+
156. "\1\34\1\35\1\36\1\37\1\40\1\41\1\42\1\43"+
157. "\1\37\1\44\1\45\1\46\1\47\1\50\1\51\1\52"+
158. "\1\53\1\54\1\55\1\56\1\57\1\60\1\61\71\0"+
159. "\2\3\1\0\1\3\3\0\30\3\32\0\1\4\72\0"+
160. "\1\6\1\0\1\6\66\0\1\62\46\0\1\63\13\0"+
161. "\2\3\1\0\1\3\3\0\1\3\1\64\1\3\1\65"+
162. "\24\3\31\0\2\3\1\0\1\66\3\0\4\3\1\67"+
163. "\6\3\1\70\2\3\1\71\6\3\1\72\2\3\31\0"+
164. "\2\3\1\0\1\3\3\0\1\73\2\3\1\74\24\3"+
165. "\31\0\2\3\1\0\1\3\3\0\1\3\1\75\12\3"+
166. "\1\76\13\3\31\0\2\3\1\0\1\77\3\0\10\3"+
167. "\1\100\1\101\7\3\1\102\6\3\31\0\2\3\1\0"+
168. "\1\3\3\0\23\3\1\103\4\3\31\0\2\3\1\0"+
169. "\1\3\3\0\5\3\1\104\2\3\1\105\17\3\31\0"+
170. "\2\3\1\0\1\3\3\0\5\3\1\106\22\3\31\0"+
171. "\2\3\1\0\1\3\3\0\11\3\1\107\16\3\31\0"+
172. "\2\3\1\0\1\3\3\0\7\3\1\110\4\3\1\111"+
173. "\5\3\1\112\5\3\31\0\2\3\1\0\1\113\3\0"+
174. "\1\114\4\3\1\115\1\3\1\116\6\3\1\117\11\3"+
175. "\31\0\2\3\1\0\1\3\3\0\1\120\10\3\1\121"+
176. "\16\3\31\0\2\3\1\0\1\3\3\0\5\3\1\122"+
177. "\6\3\1\123\3\3\1\124\7\3\31\0\2\3\1\0"+
178. "\1\3\3\0\5\3\1\125\3\3\1\126\7\3\1\127"+
179. "\6\3\31\0\2\3\1\0\1\3\3\0\1\130\4\3"+
180. "\1\131\1\3\1\132\1\133\17\3\31\0\2\3\1\0"+
181. "\1\3\3\0\3\3\1\134\4\3\1\135\17\3\31\0"+
182. "\2\3\1\0\1\3\3\0\5\3\1\65\22\3\31\0"+
183. "\2\3\1\0\1\136\3\0\30\3\31\0\2\3\1\0"+
184. "\1\3\3\0\5\3\1\137\10\3\1\140\11\3\31\0"+
185. "\1\141\2\0\1\141\3\0\30\141\1\142\14\0\1\142"+
186. "\53\0\1\142\13\0\1\142\55\0\1\143\67\0\1\144"+
187. "\11\0\1\63\12\0\7\145\1\0\35\145\1\146\22\145"+
188. "\1\147\1\150\2\147\1\150\2\147\1\0\30\150\30\147"+
189. "\47\0\1\144\5\0\1\63\67\0\1\63\66\0\1\61"+
190. "\70\0\1\142\100\0\1\61\1\0\7\62\1\0\60\62"+
191. "\1\0\2\3\1\0\1\3\3\0\2\3\1\124\25\3"+
192. "\31\0\2\3\1\0\1\3\3\0\4\3\1\151\23\3"+
193. "\31\0\2\3\1\0\1\3\3\0\5\3\1\152\22\3"+
194. "\31\0\2\3\1\0\1\3\3\0\1\153\7\3\1\154"+
195. "\6\3\1\155\10\3\31\0\2\3\1\0\1\3\3\0"+
196. "\1\156\27\3\31\0\2\3\1\0\1\3\3\0\24\3"+
197. "\1\157\2\3\1\160\31\0\2\3\1\0\1\3\3\0"+
198. "\16\3\1\161\11\3\31\0\2\3\1\0\1\3\3\0"+
199. "\16\3\1\117\11\3\31\0\2\3\1\0\1\3\3\0"+
200. "\4\3\1\162\23\3\31\0\2\3\1\0\1\3\3\0"+
201. "\16\3\1\106\11\3\31\0\2\3\1\0\1\3\3\0"+
202. "\1\3\1\163\14\3\1\164\11\3\31\0\2\3\1\0"+
203. "\1\3\3\0\10\3\1\165\5\3\1\166\11\3\31\0"+
204. "\2\3\1\0\1\3\3\0\3\3\1\167\15\3\1\124"+
205. "\6\3\31\0\2\3\1\0\1\3\3\0\2\3\1\170"+
206. "\25\3\31\0\2\3\1\0\1\3\3\0\23\3\1\171"+
207. "\4\3\31\0\2\3\1\0\1\3\3\0\11\3\1\172"+
208. "\16\3\31\0\2\3\1\0\1\3\3\0\5\3\1\173"+
209. "\22\3\31\0\2\3\1\0\1\3\3\0\11\3\1\174"+
210. "\16\3\31\0\2\3\1\0\1\3\3\0\14\3\1\175"+
211. "\13\3\31\0\2\3\1\0\1\3\3\0\4\3\1\176"+
212. "\11\3\1\177\5\3\1\200\3\3\31\0\2\3\1\0"+
213. "\1\3\3\0\1\3\1\167\26\3\31\0\2\3\1\0"+
214. "\1\3\3\0\3\3\1\64\13\3\1\173\10\3\31\0"+
215. "\2\3\1\0\1\3\3\0\4\3\1\201\16\3\1\202"+
216. "\4\3\31\0\2\3\1\0\1\3\3\0\1\131\27\3"+
217. "\31\0\2\3\1\0\1\3\3\0\1\3\1\167\2\3"+
218. "\1\203\23\3\31\0\2\3\1\0\1\3\3\0\3\3"+
219. "\1\204\10\3\1\205\13\3\31\0\2\3\1\0\1\3"+
220. "\3\0\1\206\27\3\31\0\2\3\1\0\1\3\3\0"+
221. "\14\3\1\124\13\3\31\0\2\3\1\0\1\3\3\0"+
222. "\2\3\1\207\25\3\31\0\2\3\1\0\1\3\3\0"+
223. "\25\3\1\124\2\3\31\0\2\3\1\0\1\3\3\0"+
224. "\1\3\1\210\26\3\31\0\2\3\1\0\1\3\3\0"+
225. "\4\3\1\124\2\3\1\211\3\3\1\212\14\3\31\0"+
226. "\2\3\1\0\1\3\3\0\3\3\1\213\24\3\31\0"+
227. "\2\3\1\0\1\3\3\0\7\3\1\214\10\3\1\215"+
228. "\7\3\31\0\2\3\1\0\1\3\3\0\14\3\1\216"+
229. "\13\3\31\0\2\3\1\0\1\3\3\0\7\3\1\110"+
230. "\20\3\31\0\2\3\1\0\1\3\3\0\10\3\1\124"+
231. "\17\3\31\0\2\3\1\0\1\3\3\0\5\3\1\217"+
232. "\22\3\31\0\2\3\1\0\1\3\3\0\16\3\1\220"+
233. "\11\3\31\0\2\3\1\0\1\3\3\0\6\3\1\221"+
234. "\21\3\31\0\2\3\1\0\1\3\3\0\5\3\1\222"+
235. "\10\3\1\223\11\3\31\0\2\3\1\0\1\3\3\0"+
236. "\16\3\1\224\11\3\31\0\2\3\1\0\1\3\3\0"+
237. "\7\3\1\225\6\3\1\226\11\3\31\0\2\3\1\0"+
238. "\1\3\3\0\10\3\1\227\17\3\31\0\1\141\1\0"+
239. "\1\230\1\141\3\0\30\141\1\0\1\231\30\0\1\232"+
240. "\133\0\1\233\22\0\1\150\2\0\1\150\3\0\30\150"+
241. "\6\0\1\233\22\0\2\3\1\0\1\3\3\0\5\3"+
242. "\1\124\22\3\31\0\2\3\1\0\1\3\3\0\10\3"+
243. "\1\204\17\3\31\0\2\3\1\0\1\3\3\0\4\3"+
244. "\1\234\23\3\31\0\2\3\1\0\1\3\3\0\3\3"+
245. "\1\235\24\3\31\0\2\3\1\0\1\3\3\0\16\3"+
246. "\1\151\11\3\31\0\2\3\1\0\1\3\3\0\14\3"+
247. "\1\236\13\3\31\0\2\3\1\0\1\3\3\0\14\3"+
248. "\1\237\13\3\31\0\2\3\1\0\1\3\3\0\11\3"+
249. "\1\240\16\3\31\0\2\3\1\0\1\3\3\0\4\3"+
250. "\1\203\23\3\31\0\2\3\1\0\1\3\3\0\1\213"+
251. "\27\3\31\0\2\3\1\0\1\3\3\0\16\3\1\241"+
252. "\11\3\31\0\2\3\1\0\1\3\3\0\5\3\1\117"+
253. "\22\3\31\0\2\3\1\0\1\3\3\0\5\3\1\121"+
254. "\22\3\31\0\2\3\1\0\1\3\3\0\1\3\1\124"+
255. "\26\3\31\0\2\3\1\0\1\3\3\0\11\3\1\124"+
256. "\16\3\31\0\2\3\1\0\1\3\3\0\23\3\1\242"+
257. "\4\3\31\0\2\3\1\0\1\3\3\0\11\3\1\243"+
258. "\16\3\31\0\2\3\1\0\1\3\3\0\10\3\1\244"+
259. "\17\3\31\0\2\3\1\0\1\3\3\0\7\3\1\124"+
260. "\20\3\31\0\2\3\1\0\1\3\3\0\1\245\27\3"+
261. "\31\0\2\3\1\0\1\3\3\0\24\3\1\124\3\3"+
262. "\31\0\2\3\1\0\1\3\3\0\3\3\1\246\24\3"+
263. "\31\0\2\3\1\0\1\3\3\0\14\3\1\247\13\3"+
264. "\31\0\2\3\1\0\1\3\3\0\16\3\1\250\11\3"+
265. "\31\0\2\3\1\0\1\3\3\0\11\3\1\246\16\3"+
266. "\31\0\2\3\1\0\1\3\3\0\7\3\1\251\20\3"+
267. "\31\0\2\3\1\0\1\3\3\0\13\3\1\252\14\3"+
268. "\31\0\2\3\1\0\1\3\3\0\4\3\1\124\23\3"+
269. "\31\0\2\3\1\0\1\3\3\0\1\3\1\253\2\3"+
270. "\1\254\23\3\31\0\2\3\1\0\1\3\3\0\1\3"+
271. "\1\166\26\3\31\0\2\3\1\0\1\3\3\0\11\3"+
272. "\1\255\16\3\31\0\2\3\1\0\1\3\3\0\4\3"+
273. "\1\256\23\3\31\0\2\3\1\0\1\3\3\0\16\3"+
274. "\1\257\11\3\31\0\2\3\1\0\1\3\3\0\7\3"+
275. "\1\260\20\3\31\0\2\3\1\0\1\3\3\0\6\3"+
276. "\1\224\21\3\31\0\2\3\1\0\1\3\3\0\11\3"+
277. "\1\261\16\3\31\0\2\3\1\0\1\3\3\0\1\262"+
278. "\27\3\31\0\2\3\1\0\1\3\3\0\1\263\27\3"+
279. "\31\0\2\3\1\0\1\3\3\0\1\204\27\3\31\0"+
280. "\2\3\1\0\1\3\3\0\11\3\1\264\16\3\31\0"+
281. "\2\3\1\0\1\3\3\0\7\3\1\265\20\3\31\0"+
282. "\2\3\1\0\1\3\3\0\4\3\1\266\23\3\31\0"+
283. "\2\3\1\0\1\3\3\0\14\3\1\267\11\3\1\270"+
284. "\1\3\31\0\2\3\1\0\1\3\3\0\7\3\1\167"+
285. "\20\3\31\0\2\3\1\0\1\3\3\0\1\271\27\3"+
286. "\31\0\2\3\1\0\1\3\3\0\17\3\1\124\10\3"+
287. "\31\0\2\3\1\0\1\3\3\0\4\3\1\272\23\3"+
288. "\34\0\1\273\65\0\1\232\41\0\1\274\24\0\2\3"+
289. "\1\0\1\3\3\0\16\3\1\275\11\3\31\0\2\3"+
290. "\1\0\1\3\3\0\13\3\1\204\14\3\31\0\2\3"+
291. "\1\0\1\3\3\0\20\3\1\124\7\3\31\0\2\3"+
292. "\1\0\1\3\3\0\11\3\1\226\16\3\31\0\2\3"+
293. "\1\0\1\3\3\0\5\3\1\236\22\3\31\0\2\3"+
294. "\1\0\1\3\3\0\24\3\1\157\3\3\31\0\2\3"+
295. "\1\0\1\3\3\0\7\3\1\270\20\3\31\0\2\3"+
296. "\1\0\1\3\3\0\14\3\1\276\1\3\1\226\1\277"+
297. "\10\3\31\0\2\3\1\0\1\3\3\0\1\300\27\3"+
298. "\31\0\2\3\1\0\1\3\3\0\12\3\1\124\15\3"+
299. "\31\0\2\3\1\0\1\3\3\0\10\3\1\117\17\3"+
300. "\31\0\2\3\1\0\1\3\3\0\4\3\1\301\23\3"+
301. "\31\0\2\3\1\0\1\3\3\0\1\3\1\302\26\3"+
302. "\31\0\2\3\1\0\1\3\3\0\16\3\1\303\11\3"+
303. "\31\0\2\3\1\0\1\124\3\0\30\3\31\0\2\3"+
304. "\1\0\1\3\3\0\4\3\1\304\23\3\31\0\2\3"+
305. "\1\0\1\3\3\0\16\3\1\305\11\3\31\0\2\3"+
306. "\1\0\1\3\3\0\1\3\1\306\26\3\31\0\2\3"+
307. "\1\0\1\3\3\0\10\3\1\307\17\3\31\0\2\3"+
308. "\1\0\1\3\3\0\14\3\1\167\13\3\31\0\2\3"+
309. "\1\0\1\3\3\0\3\3\1\310\24\3\31\0\2\3"+
310. "\1\0\1\3\3\0\4\3\1\167\23\3\31\0\2\3"+
311. "\1\0\1\3\3\0\3\3\1\311\24\3\31\0\2\3"+
312. "\1\0\1\3\3\0\2\3\1\312\25\3\31\0\2\3"+
313. "\1\0\1\3\3\0\14\3\1\226\13\3\31\0\2\3"+
314. "\1\0\1\3\3\0\16\3\1\313\11\3\31\0\2\3"+
315. "\1\0\1\3\3\0\11\3\1\314\16\3\31\0\2\3"+
316. "\1\0\1\3\3\0\4\3\1\236\23\3\31\0\2\3"+
317. "\1\0\1\3\3\0\1\261\27\3\31\0\2\3\1\0"+
318. "\1\3\3\0\4\3\1\136\23\3\31\0\2\3\1\0"+
319. "\1\3\3\0\3\3\1\315\24\3\71\0\1\231\27\0"+
320. "\2\3\1\0\1\3\3\0\13\3\1\304\14\3\31\0"+
321. "\2\3\1\0\1\3\3\0\1\316\27\3\31\0\2\3"+
322. "\1\0\1\3\3\0\11\3\1\236\16\3\31\0\2\3"+
323. "\1\0\1\3\3\0\4\3\1\317\23\3\31\0\2\3"+
324. "\1\0\1\3\3\0\11\3\1\320\16\3\31\0\2\3"+
325. "\1\0\1\3\3\0\4\3\1\321\23\3\31\0\2\3"+
326. "\1\0\1\3\3\0\13\3\1\322\14\3\31\0\2\3"+
327. "\1\0\1\3\3\0\15\3\1\323\12\3\31\0\2\3"+
328. "\1\0\1\3\3\0\14\3\1\324\13\3\31\0\2\3"+
329. "\1\0\1\3\3\0\23\3\1\325\4\3\31\0\2\3"+
330. "\1\0\1\3\3\0\11\3\1\326\16\3\31\0\2\3"+
331. "\1\0\1\3\3\0\17\3\1\167\10\3\31\0\2\3"+
332. "\1\0\1\3\3\0\7\3\1\204\20\3\31\0\2\3"+
333. "\1\0\1\3\3\0\16\3\1\327\11\3\31\0\2\3"+
334. "\1\0\1\3\3\0\13\3\1\124\14\3\31\0\2\3"+
335. "\1\0\1\3\3\0\13\3\1\330\14\3\31\0\2\3"+
336. "\1\0\1\3\3\0\1\173\27\3\31\0\2\3\1\0"+
337. "\1\3\3\0\2\3\1\167\25\3\31\0\2\3\1\0"+
338. "\1\3\3\0\5\3\1\131\22\3\31\0\2\3\1\0"+
339. "\1\3\3\0\10\3\1\331\17\3\31\0\2\3\1\0"+
340. "\1\3\3\0\11\3\1\131\16\3\31\0\2\3\1\0"+
341. "\1\3\3\0\16\3\1\204\11\3\31\0\2\3\1\0"+
342. "\1\3\3\0\13\3\1\332\14\3\31\0\2\3\1\0"+
343. "\1\3\3\0\3\3\1\167\24\3\31\0\2\3\1\0"+
344. "\1\3\3\0\1\333\27\3\31\0\2\3\1\0\1\3"+
345. "\3\0\1\64\27\3\31\0\2\3\1\0\1\3\3\0"+
346. "\13\3\1\334\14\3\31\0\2\3\1\0\1\3\3\0"+
347. "\4\3\1\237\23\3\31\0\2\3\1\0\1\3\3\0"+
348. "\23\3\1\335\4\3\31\0\2\3\1\0\1\3\3\0"+
349. "\1\336\27\3\31\0\2\3\1\0\1\3\3\0\13\3"+
350. "\1\167\14\3\31\0\2\3\1\0\1\3\3\0\10\3"+
351. "\1\337\17\3\31\0\2\3\1\0\1\3\3\0\1\3"+
352. "\1\204\26\3\31\0\2\3\1\0\1\3\3\0\11\3"+
353. "\1\340\16\3\31\0\2\3\1\0\1\3\3\0\4\3"+
354. "\1\334\23\3\30\0";
356. private static int [] zzUnpackTrans() {
357. int [] result = new int[11256];
358. int offset = 0;
359. offset = zzUnpackTrans(ZZ\_TRANS\_PACKED\_0, offset, result);
360. return result;
361. }
363. private static int zzUnpackTrans(String packed, int offset, int [] result) {
364. int i = 0; /\* index in packed string \*/
365. int j = offset; /\* index in unpacked array \*/
366. int l = packed.length();
367. while (i < l) {
368. int count = packed.charAt(i++);
369. int value = packed.charAt(i++);
370. value--;
371. do result[j++] = value; while (--count > 0);
372. }
373. return j;
374. }

377. /\* error codes \*/
378. private static final int ZZ\_UNKNOWN\_ERROR = 0;
379. private static final int ZZ\_NO\_MATCH = 1;
380. private static final int ZZ\_PUSHBACK\_2BIG = 2;
382. /\* error messages for the codes above \*/
383. private static final String ZZ\_ERROR\_MSG[] = {
384. "Unkown internal scanner error",
385. "Error: could not match input",
386. "Error: pushback value was too large"
387. };
389. /\*\*
390. \* ZZ\_ATTRIBUTE[aState] contains the attributes of state <code>aState</code>
391. \*/
392. private static final int [] ZZ\_ATTRIBUTE = zzUnpackAttribute();
394. private static final String ZZ\_ATTRIBUTE\_PACKED\_0 =
395. "\1\0\1\11\2\1\1\11\31\1\1\11\4\1\2\11"+
396. "\2\1\1\11\1\1\5\11\2\1\1\11\1\1\1\11"+
397. "\55\1\1\0\1\11\1\0\1\11\1\0\1\11\2\0"+
398. "\57\1\1\0\1\11\1\0\1\11\37\1\1\0\1\11"+
399. "\44\1";
401. private static int [] zzUnpackAttribute() {
402. int [] result = new int[224];
403. int offset = 0;
404. offset = zzUnpackAttribute(ZZ\_ATTRIBUTE\_PACKED\_0, offset, result);
405. return result;
406. }
408. private static int zzUnpackAttribute(String packed, int offset, int [] result) {
409. int i = 0; /\* index in packed string \*/
410. int j = offset; /\* index in unpacked array \*/
411. int l = packed.length();
412. while (i < l) {
413. int count = packed.charAt(i++);
414. int value = packed.charAt(i++);
415. do result[j++] = value; while (--count > 0);
416. }
417. return j;
418. }
420. /\*\* the input device \*/
421. private java.io.Reader zzReader;
423. /\*\* the current state of the DFA \*/
424. private int zzState;
426. /\*\* the current lexical state \*/
427. private int zzLexicalState = YYINITIAL;
429. /\*\* this buffer contains the current text to be matched and is
430. the source of the yytext() string \*/
431. private char zzBuffer[] = new char[ZZ\_BUFFERSIZE];
433. /\*\* the textposition at the last accepting state \*/
434. private int zzMarkedPos;
436. /\*\* the current text position in the buffer \*/
437. private int zzCurrentPos;
439. /\*\* startRead marks the beginning of the yytext() string in the buffer \*/
440. private int zzStartRead;
442. /\*\* endRead marks the last character in the buffer, that has been read
443. from input \*/
444. private int zzEndRead;
446. /\*\* number of newlines encountered up to the start of the matched text \*/
447. private int yyline;
449. /\*\* the number of characters up to the start of the matched text \*/
450. private int yychar;
452. /\*\*
453. \* the number of characters from the last newline up to the start of the
454. \* matched text
455. \*/
456. private int yycolumn;
458. /\*\*
459. \* zzAtBOL == true <=> the scanner is currently at the beginning of a line
460. \*/
461. private boolean zzAtBOL = true;
463. /\*\* zzAtEOF == true <=> the scanner is at the EOF \*/
464. private boolean zzAtEOF;
466. /\*\* denotes if the user-EOF-code has already been executed \*/
467. private boolean zzEOFDone;
469. /\* user code: \*/
470. public String lexeme;
471. TableData data = new TableData();

474. /\*\*
475. \* Creates a new scanner
476. \* There is also a java.io.InputStream version of this constructor.
477. \*
478. \* @param in the java.io.Reader to read input from.
479. \*/
480. Lexer(java.io.Reader in) {
481. this.zzReader = in;
482. }
484. /\*\*
485. \* Creates a new scanner.
486. \* There is also java.io.Reader version of this constructor.
487. \*
488. \* @param in the java.io.Inputstream to read input from.
489. \*/
490. Lexer(java.io.InputStream in) {
491. this(new java.io.InputStreamReader(in));
492. }
494. /\*\*
495. \* Unpacks the compressed character translation table.
496. \*
497. \* @param packed the packed character translation table
498. \* @return the unpacked character translation table
499. \*/
500. private static char [] zzUnpackCMap(String packed) {
501. char [] map = new char[0x10000];
502. int i = 0; /\* index in packed string \*/
503. int j = 0; /\* index in unpacked array \*/
504. while (i < 136) {
505. int count = packed.charAt(i++);
506. char value = packed.charAt(i++);
507. do map[j++] = value; while (--count > 0);
508. }
509. return map;
510. }

513. /\*\*
514. \* Refills the input buffer.
515. \*
516. \* @return <code>false</code>, iff there was new input.
517. \*
518. \* @exception java.io.IOException if any I/O-Error occurs
519. \*/
520. private boolean zzRefill() throws java.io.IOException {
522. /\* first: make room (if you can) \*/
523. if (zzStartRead > 0) {
524. System.arraycopy(zzBuffer, zzStartRead,
525. zzBuffer, 0,
526. zzEndRead-zzStartRead);
528. /\* translate stored positions \*/
529. zzEndRead-= zzStartRead;
530. zzCurrentPos-= zzStartRead;
531. zzMarkedPos-= zzStartRead;
532. zzStartRead = 0;
533. }
535. /\* is the buffer big enough? \*/
536. if (zzCurrentPos >= zzBuffer.length) {
537. /\* if not: blow it up \*/
538. char newBuffer[] = new char[zzCurrentPos\*2];
539. System.arraycopy(zzBuffer, 0, newBuffer, 0, zzBuffer.length);
540. zzBuffer = newBuffer;
541. }
543. /\* finally: fill the buffer with new input \*/
544. int numRead = zzReader.read(zzBuffer, zzEndRead,
545. zzBuffer.length-zzEndRead);
547. if (numRead > 0) {
548. zzEndRead+= numRead;
549. return false;
550. }
551. // unlikely but not impossible: read 0 characters, but not at end of stream
552. if (numRead == 0) {
553. int c = zzReader.read();
554. if (c == -1) {
555. return true;
556. } else {
557. zzBuffer[zzEndRead++] = (char) c;
558. return false;
559. }
560. }
562. // numRead < 0
563. return true;
564. }

567. /\*\*
568. \* Closes the input stream.
569. \*/
570. public final void yyclose() throws java.io.IOException {
571. zzAtEOF = true; /\* indicate end of file \*/
572. zzEndRead = zzStartRead; /\* invalidate buffer \*/
574. if (zzReader != null)
575. zzReader.close();
576. }

579. /\*\*
580. \* Resets the scanner to read from a new input stream.
581. \* Does not close the old reader.
582. \*
583. \* All internal variables are reset, the old input stream
584. \* <b>cannot</b> be reused (internal buffer is discarded and lost).
585. \* Lexical state is set to <tt>ZZ\_INITIAL</tt>.
586. \*
587. \* @param reader the new input stream
588. \*/
589. public final void yyreset(java.io.Reader reader) {
590. zzReader = reader;
591. zzAtBOL = true;
592. zzAtEOF = false;
593. zzEOFDone = false;
594. zzEndRead = zzStartRead = 0;
595. zzCurrentPos = zzMarkedPos = 0;
596. yyline = yychar = yycolumn = 0;
597. zzLexicalState = YYINITIAL;
598. }

601. /\*\*
602. \* Returns the current lexical state.
603. \*/
604. public final int yystate() {
605. return zzLexicalState;
606. }

609. /\*\*
610. \* Enters a new lexical state
611. \*
612. \* @param newState the new lexical state
613. \*/
614. public final void yybegin(int newState) {
615. zzLexicalState = newState;
616. }

619. /\*\*
620. \* Returns the text matched by the current regular expression.
621. \*/
622. public final String yytext() {
623. return new String( zzBuffer, zzStartRead, zzMarkedPos-zzStartRead );
624. }

627. /\*\*
628. \* Returns the character at position <tt>pos</tt> from the
629. \* matched text.
630. \*
631. \* It is equivalent to yytext().charAt(pos), but faster
632. \*
633. \* @param pos the position of the character to fetch.
634. \* A value from 0 to yylength()-1.
635. \*
636. \* @return the character at position pos
637. \*/
638. public final char yycharat(int pos) {
639. return zzBuffer[zzStartRead+pos];
640. }

643. /\*\*
644. \* Returns the length of the matched text region.
645. \*/
646. public final int yylength() {
647. return zzMarkedPos-zzStartRead;
648. }

651. /\*\*
652. \* Reports an error that occured while scanning.
653. \*
654. \* In a wellformed scanner (no or only correct usage of
655. \* yypushback(int) and a match-all fallback rule) this method
656. \* will only be called with things that "Can't Possibly Happen".
657. \* If this method is called, something is seriously wrong
658. \* (e.g. a JFlex bug producing a faulty scanner etc.).
659. \*
660. \* Usual syntax/scanner level error handling should be done
661. \* in error fallback rules.
662. \*
663. \* @param errorCode the code of the errormessage to display
664. \*/
665. private void zzScanError(int errorCode) {
666. String message;
667. try {
668. message = ZZ\_ERROR\_MSG[errorCode];
669. }
670. catch (ArrayIndexOutOfBoundsException e) {
671. message = ZZ\_ERROR\_MSG[ZZ\_UNKNOWN\_ERROR];
672. }
674. throw new Error(message);
675. }

678. /\*\*
679. \* Pushes the specified amount of characters back into the input stream.
680. \*
681. \* They will be read again by then next call of the scanning method
682. \*
683. \* @param number the number of characters to be read again.
684. \* This number must not be greater than yylength()!
685. \*/
686. public void yypushback(int number) {
687. if ( number > yylength() )
688. zzScanError(ZZ\_PUSHBACK\_2BIG);
690. zzMarkedPos -= number;
691. }

694. /\*\*
695. \* Resumes scanning until the next regular expression is matched,
696. \* the end of input is encountered or an I/O-Error occurs.
697. \*
698. \* @return the next token
699. \* @exception java.io.IOException if any I/O-Error occurs
700. \*/
701. public Tokens yylex() throws java.io.IOException {
702. int zzInput;
703. int zzAction;
705. // cached fields:
706. int zzCurrentPosL;
707. int zzMarkedPosL;
708. int zzEndReadL = zzEndRead;
709. char [] zzBufferL = zzBuffer;
710. char [] zzCMapL = ZZ\_CMAP;
712. int [] zzTransL = ZZ\_TRANS;
713. int [] zzRowMapL = ZZ\_ROWMAP;
714. int [] zzAttrL = ZZ\_ATTRIBUTE;
716. while (true) {
717. zzMarkedPosL = zzMarkedPos;
719. boolean zzR = false;
720. for (zzCurrentPosL = zzStartRead; zzCurrentPosL < zzMarkedPosL;
721. zzCurrentPosL++) {
722. switch (zzBufferL[zzCurrentPosL]) {
723. case '\u000B':
724. case '\u000C':
725. case '\u0085':
726. case '\u2028':
727. case '\u2029':
728. yyline++;
729. yycolumn = 0;
730. zzR = false;
731. break;
732. case '\r':
733. yyline++;
734. yycolumn = 0;
735. zzR = true;
736. break;
737. case '\n':
738. if (zzR)
739. zzR = false;
740. else {
741. yyline++;
742. yycolumn = 0;
743. }
744. break;
745. default:
746. zzR = false;
747. yycolumn++;
748. }
749. }
751. if (zzR) {
752. // peek one character ahead if it is \n (if we have counted one line too much)
753. boolean zzPeek;
754. if (zzMarkedPosL < zzEndReadL)
755. zzPeek = zzBufferL[zzMarkedPosL] == '\n';
756. else if (zzAtEOF)
757. zzPeek = false;
758. else {
759. boolean eof = zzRefill();
760. zzEndReadL = zzEndRead;
761. zzMarkedPosL = zzMarkedPos;
762. zzBufferL = zzBuffer;
763. if (eof)
764. zzPeek = false;
765. else
766. zzPeek = zzBufferL[zzMarkedPosL] == '\n';
767. }
768. if (zzPeek) yyline--;
769. }
770. zzAction = -1;
772. zzCurrentPosL = zzCurrentPos = zzStartRead = zzMarkedPosL;
774. zzState = ZZ\_LEXSTATE[zzLexicalState];

777. zzForAction: {
778. while (true) {
780. if (zzCurrentPosL < zzEndReadL)
781. zzInput = zzBufferL[zzCurrentPosL++];
782. else if (zzAtEOF) {
783. zzInput = YYEOF;
784. break zzForAction;
785. }
786. else {
787. // store back cached positions
788. zzCurrentPos = zzCurrentPosL;
789. zzMarkedPos = zzMarkedPosL;
790. boolean eof = zzRefill();
791. // get translated positions and possibly new buffer
792. zzCurrentPosL = zzCurrentPos;
793. zzMarkedPosL = zzMarkedPos;
794. zzBufferL = zzBuffer;
795. zzEndReadL = zzEndRead;
796. if (eof) {
797. zzInput = YYEOF;
798. break zzForAction;
799. }
800. else {
801. zzInput = zzBufferL[zzCurrentPosL++];
802. }
803. }
804. int zzNext = zzTransL[ zzRowMapL[zzState] + zzCMapL[zzInput] ];
805. if (zzNext == -1) break zzForAction;
806. zzState = zzNext;
808. int zzAttributes = zzAttrL[zzState];
809. if ( (zzAttributes & 1) == 1 ) {
810. zzAction = zzState;
811. zzMarkedPosL = zzCurrentPosL;
812. if ( (zzAttributes & 8) == 8 ) break zzForAction;
813. }
815. }
816. }
818. // store back cached position
819. zzMarkedPos = zzMarkedPosL;
821. switch (zzAction < 0 ? zzAction : ZZ\_ACTION[zzAction]) {
822. case 16:
823. { data.linea=yyline; lexeme=yytext(); return Porcentaje;
824. }
825. case 29: break;
826. case 21:
827. { data.linea=yyline; lexeme=yytext(); return Interogacion;
828. }
829. case 30: break;
830. case 15:
831. { data.linea=yyline; lexeme=yytext(); return DosPuntos;
832. }
833. case 31: break;
834. case 20:
835. { data.linea=yyline; lexeme=yytext(); return Dinero;
836. }
837. case 32: break;
838. case 12:
839. { data.linea=yyline; lexeme=yytext(); return Terminador;
840. }
841. case 33: break;
842. case 25:
843. { data.linea=yyline; lexeme = yytext(); return Op\_Incremento;
844. }
845. case 34: break;
846. case 17:
847. { data.linea=yyline; lexeme=yytext(); return BarraInvertida;
848. }
849. case 35: break;
850. case 8:
851. { data.linea=yyline; lexeme=yytext(); return Delimitador;
852. }
853. case 36: break;
854. case 7:
855. { data.linea=yyline; lexeme = yytext(); return Op\_Relacional;
856. }
857. case 37: break;
858. case 2:
859. { data.linea=yyline; lexeme=yytext(); return Identificador;
860. }
861. case 38: break;
862. case 1:
863. { data.linea=yyline; lexeme=yytext(); return ERROR;
864. }
865. case 39: break;
866. case 22:
867. { data.linea=yyline; lexeme=yytext(); return Op\_Logico;
868. }
869. case 40: break;
870. case 3:
871. { data.linea=yyline; lexeme=yytext(); return Numero;
872. }
873. case 41: break;
874. case 9:
875. { data.linea=yyline; lexeme=yytext(); return Comilla;
876. }
877. case 42: break;
878. case 28:
879. { data.linea=yyline; lexeme=yytext(); return Caracter;
880. }
881. case 43: break;
882. case 13:
883. { data.linea=yyline; lexeme=yytext(); return And;
884. }
885. case 44: break;
886. case 26:
887. { data.linea=yyline; lexeme=yytext(); return Cadena;
888. }
889. case 45: break;
890. case 23:
891. { data.linea=yyline; lexeme = yytext(); return Op\_Atribucion;
892. }
893. case 46: break;
894. case 11:
895. { data.linea=yyline; lexeme=yytext(); return Hastag;
896. }
897. case 47: break;
898. case 10:
899. { data.linea=yyline; lexeme=yytext(); return ComillaSimple;
900. }
901. case 48: break;
902. case 5:
903. { /\*Ignore\*/
904. }
905. case 49: break;
906. case 14:
907. { data.linea=yyline; lexeme=yytext(); return Igual;
908. }
909. case 50: break;
910. case 24:
911. { data.linea=yyline; lexeme=yytext(); return Reservadas;
912. }
913. case 51: break;
914. case 18:
915. { data.linea=yyline; lexeme=yytext(); return Virgulilla;
916. }
917. case 52: break;
918. case 4:
919. { data.linea=yyline; lexeme=yytext(); return Punto;
920. }
921. case 53: break;
922. case 27:
923. { data.linea=yyline; lexeme=yytext(); return Libreria;
924. }
925. case 54: break;
926. case 19:
927. { data.linea=yyline; lexeme=yytext(); return Arroba;
928. }
929. case 55: break;
930. case 6:
931. { data.linea=yyline; lexeme=yytext(); return Op\_Aritmeticos;
932. }
933. case 56: break;
934. default:
935. if (zzInput == YYEOF && zzStartRead == zzCurrentPos) {
936. zzAtEOF = true;
937. return null;
938. }
939. else {
940. zzScanError(ZZ\_NO\_MATCH);
941. }
942. }
943. }
944. }

947. }

## LineNumber.java

1. /\*
2. \* ARCHIVO QUE MODIFICA EL DISEÑO DEL JSCROOL PARA AGREGARLE NUMERACION
3. \*/
4. package code;
6. /\* Librerias \*/
7. import java.awt.\*;
8. import java.beans.\*;
9. import java.util.HashMap;
10. import javax.swing.\*;
11. import javax.swing.border.\*;
12. import javax.swing.event.\*;
13. import javax.swing.text.\*;
15. /\* Clase e Implementaciones \*/
16. public class LineNumber extends JPanel
17. implements CaretListener, DocumentListener, PropertyChangeListener {
19. /\* Variables publicas \*/
20. public final static float LEFT = 0.0f;
21. public final static float CENTER = 0.5f;
22. public final static float RIGHT = 1.0f;
23. private Color color1 = new Color(7,86,100);
24. private final static Border OUTER = new MatteBorder(0, 0, 0, 2,Color.BLACK);
25. private final static int HEIGHT = Integer.MAX\_VALUE - 1000000;
26. private JTextComponent component;
27. private boolean updateFont;
28. private int borderGap;
29. private Color currentLineForeground;
30. private float digitAlignment;
31. private int minimumDisplayDigits;
32. private int lastDigits;
33. private int lastHeight;
34. private int lastLine;
35. private HashMap<String, FontMetrics> fonts;

38. public LineNumber(JTextComponent component) {
39. this(component, 3);
40. }

43. public LineNumber(JTextComponent component, int minimumDisplayDigits) {
44. this.component = component;
46. setFont(component.getFont());
48. setBorderGap(5);
49. setCurrentLineForeground(color1);
50. setDigitAlignment(RIGHT);
51. setMinimumDisplayDigits(minimumDisplayDigits);
53. component.getDocument().addDocumentListener(this);
54. component.addCaretListener(this);
55. component.addPropertyChangeListener("font", this);
56. }

59. public boolean getUpdateFont() {
60. return updateFont;
61. }

64. public void setUpdateFont(boolean updateFont) {
65. this.updateFont = updateFont;
66. }

69. public int getBorderGap() {
70. return borderGap;
71. }

74. public void setBorderGap(int borderGap) {
75. this.borderGap = borderGap;
76. Border inner = new EmptyBorder(0, borderGap, 0, borderGap);
77. setBorder(new CompoundBorder(OUTER, inner));
78. lastDigits = 0;
79. setPreferredWidth();
80. }

83. public Color getCurrentLineForeground() {
84. return currentLineForeground == null ? getForeground() : currentLineForeground;
85. }

88. public void setCurrentLineForeground(Color currentLineForeground) {
89. this.currentLineForeground = currentLineForeground;
90. }

93. public float getDigitAlignment() {
94. return digitAlignment;
95. }
96. public void setDigitAlignment(float digitAlignment) {
97. this.digitAlignment
98. = digitAlignment > 1.0f ? 1.0f : digitAlignment < 0.0f ? -1.0f : digitAlignment;
99. }

102. public int getMinimumDisplayDigits() {
103. return minimumDisplayDigits;
104. }

107. public void setMinimumDisplayDigits(int minimumDisplayDigits) {
108. this.minimumDisplayDigits = minimumDisplayDigits;
109. setPreferredWidth();
110. }

113. private void setPreferredWidth() {
114. Element root = component.getDocument().getDefaultRootElement();
115. int lines = root.getElementCount();
116. int digits = Math.max(String.valueOf(lines).length(), minimumDisplayDigits);
118. if (lastDigits != digits) {
119. lastDigits = digits;
120. FontMetrics fontMetrics = getFontMetrics(getFont());
121. int width = fontMetrics.charWidth('0') \* digits;
122. Insets insets = getInsets();
123. int preferredWidth = insets.left + insets.right + width;
125. Dimension d = getPreferredSize();
126. d.setSize(preferredWidth, HEIGHT);
127. setPreferredSize(d);
128. setSize(d);
129. }
130. }
132. @Override
133. public void paintComponent(Graphics g) {
134. super.paintComponent(g);
136. FontMetrics fontMetrics = component.getFontMetrics(component.getFont());
137. Insets insets = getInsets();
138. int availableWidth = getSize().width - insets.left - insets.right;
140. Rectangle clip = g.getClipBounds();
141. int rowStartOffset = component.viewToModel(new Point(0, clip.y));
142. int endOffset = component.viewToModel(new Point(0, clip.y + clip.height));
144. while (rowStartOffset <= endOffset) {
145. try {
146. if (isCurrentLine(rowStartOffset)) {
147. g.setColor(getCurrentLineForeground());
148. } else {
149. g.setColor(getForeground());
150. }
152. String lineNumber = getTextLineNumber(rowStartOffset);
153. int stringWidth = fontMetrics.stringWidth(lineNumber);
154. int x = getOffsetX(availableWidth, stringWidth) + insets.left;
155. int y = getOffsetY(rowStartOffset, fontMetrics);
156. g.drawString(lineNumber, x, y);
157. rowStartOffset = Utilities.getRowEnd(component, rowStartOffset) + 1;
158. } catch (Exception e) {
159. break;
160. }
161. }
162. }
164. private boolean isCurrentLine(int rowStartOffset) {
165. int caretPosition = component.getCaretPosition();
166. Element root = component.getDocument().getDefaultRootElement();
168. if (root.getElementIndex(rowStartOffset) == root.getElementIndex(caretPosition)) {
169. return true;
170. } else {
171. return false;
172. }
173. }
175. protected String getTextLineNumber(int rowStartOffset) {
176. Element root = component.getDocument().getDefaultRootElement();
177. int index = root.getElementIndex(rowStartOffset);
178. Element line = root.getElement(index);
180. if (line.getStartOffset() == rowStartOffset) {
181. return String.valueOf(index + 1);
182. } else {
183. return "";
184. }
185. }
187. /\*
188. \* Determine the X offset to properly align the line number when drawn
189. \*/
190. private int getOffsetX(int availableWidth, int stringWidth) {
191. return (int) ((availableWidth - stringWidth) \* digitAlignment);
192. }
194. /\*
195. \* Determine the Y offset for the current row
196. \*/
197. private int getOffsetY(int rowStartOffset, FontMetrics fontMetrics)
198. throws BadLocationException {
199. // Get the bounding rectangle of the row
201. Rectangle r = component.modelToView(rowStartOffset);
202. int lineHeight = fontMetrics.getHeight();
203. int y = r.y + r.height;
204. int descent = 0;
206. // The text needs to be positioned above the bottom of the bounding
207. // rectangle based on the descent of the font(s) contained on the row.
208. if (r.height == lineHeight) // default font is being used
209. {
210. descent = fontMetrics.getDescent();
211. } else // We need to check all the attributes for font changes
212. {
213. if (fonts == null) {
214. fonts = new HashMap<String, FontMetrics>();
215. }
217. Element root = component.getDocument().getDefaultRootElement();
218. int index = root.getElementIndex(rowStartOffset);
219. Element line = root.getElement(index);
221. for (int i = 0; i < line.getElementCount(); i++) {
222. Element child = line.getElement(i);
223. AttributeSet as = child.getAttributes();
224. String fontFamily = (String) as.getAttribute(StyleConstants.FontFamily);
225. Integer fontSize = (Integer) as.getAttribute(StyleConstants.FontSize);
226. String key = fontFamily + fontSize;
228. FontMetrics fm = fonts.get(key);
230. if (fm == null) {
231. Font font = new Font(fontFamily, Font.PLAIN, fontSize);
232. fm = component.getFontMetrics(font);
233. fonts.put(key, fm);
234. }
236. descent = Math.max(descent, fm.getDescent());
237. }
238. }
240. return y - descent;
241. }
243. //
244. // Implement CaretListener interface
245. //
246. @Override
247. public void caretUpdate(CaretEvent e) {
248. // Get the line the caret is positioned on
250. int caretPosition = component.getCaretPosition();
251. Element root = component.getDocument().getDefaultRootElement();
252. int currentLine = root.getElementIndex(caretPosition);
254. // Need to repaint so the correct line number can be highlighted
255. if (lastLine != currentLine) {
256. repaint();
257. lastLine = currentLine;
258. }
259. }
261. //
262. // Implement DocumentListener interface
263. //
264. @Override
265. public void changedUpdate(DocumentEvent e) {
266. documentChanged();
267. }
269. @Override
270. public void insertUpdate(DocumentEvent e) {
271. documentChanged();
272. }
274. @Override
275. public void removeUpdate(DocumentEvent e) {
276. documentChanged();
277. }
279. /\*
280. \* A document change may affect the number of displayed lines of text.
281. \* Therefore the lines numbers will also change.
282. \*/
283. private void documentChanged() {
284. // View of the component has not been updated at the time
285. // the DocumentEvent is fired
287. SwingUtilities.invokeLater(new Runnable() {
288. @Override
289. public void run() {
290. try {
291. int endPos = component.getDocument().getLength();
292. Rectangle rect = component.modelToView(endPos);
294. if (rect != null && rect.y != lastHeight) {
295. setPreferredWidth();
296. repaint();
297. lastHeight = rect.y;
298. }
299. } catch (BadLocationException ex) {
300. /\* nothing to do \*/ }
301. }
302. });
303. }
305. @Override
306. public void propertyChange(PropertyChangeEvent evt) {
307. if (evt.getNewValue() instanceof Font) {
308. if (updateFont) {
309. Font newFont = (Font) evt.getNewValue();
310. setFont(newFont);
311. lastDigits = 0;
312. setPreferredWidth();
313. } else {
314. repaint();
315. }
316. }
317. }
318. }

## TableData.java

1. /\*
2. ARCHIVO CONTENEDOR DE LA UBICACION DEL LEXEMA Y EL TOKEN ENCONTRADO
3. \*/
5. /\* Paquete \*/
6. package code;
8. /\* Clase \*/
9. public class TableData {
11. /\* Variables \*/
12. public static int linea;
13. public static String lexema;
14. public static Tokens token;
16. /\* Metodo que guarda los valores \*/
17. public static void setInfoCode(int li, String le, Tokens to){
18. linea = li;
19. lexema = le;
20. token = to;
21. }
22. }

## TableRender.java

1. /\*
2. ARCHIVO QUE MODIFICA EL DISEÑO DE LA TABLA PARA PONER COLORES
3. \*/
5. /\* Paquete \*/
6. package code;
8. /\* Librerias \*/
9. import java.awt.Color;
10. import java.awt.Component;
11. import javax.swing.JLabel;
12. import javax.swing.JTable;
13. import javax.swing.table.DefaultTableCellRenderer;
15. /\* Clase \*/
16. public class TableRender extends DefaultTableCellRenderer{
18. /\* Metodo con el que se modifica el diseño del jTable \*/
19. @Override
20. public Component getTableCellRendererComponent(JTable jtable, Object o, boolean bln, boolean bln1, int i, int il) {
21. /\* Se crea una objeto tipo etiqueta en la ubicacion de la celda de la tabla con la informacion obtenida \*/
22. JLabel label = (JLabel) super.getTableCellRendererComponent(jtable, o, bln, bln1, i, il);
23. /\* Se analiza el valor del jLabel \*/
24. if(o instanceof String ){
25. /\* Variable que guarda el valor \*/
26. String Dato = (String) o;
27. /\* Condicion que si es verdadera marca en rojo la celda y si no la deja de blanco \*/
28. if(Dato.equals("NO DEFINIDO")){
29. /\* Cambia el color del fondo y de la letras \*/
30. label.setBackground(Color.RED);
31. label.setForeground(Color.WHITE);
32. } else {
33. label.setBackground(Color.WHITE);
34. label.setForeground(Color.BLACK);
35. }
36. }
37. /\* Retorna el valor del jLabel \*/
38. return label;
39. }
41. }

## Tokens.java

1. /\* ARCHIVO QUE ALMACENA LOS TIPOS DE TOKENS QUE SE PUEDEN ENCONTRAR \*/
2. /\* Paquete \*/
3. package code;
5. /\* Clase \*/
6. public enum Tokens {
7. Libreria,
8. Reservadas,
9. Identificador,
10. Cadena,
11. Numero,
12. Caracter,
13. Delimitador,
14. Hastag,
15. Terminador,
16. Coma,
17. And,
18. Igual,
19. Punto,
20. DosPuntos,
21. Porcentaje,
22. BarraInvertida,
23. Virgulilla,
24. Arroba,
25. Dinero,
26. Interogacion,
27. ComillaSimple,
28. Comilla,
29. Op\_Aritmeticos,
30. Op\_Relacional,
31. Op\_Logico,
32. Op\_Atribucion,
33. Op\_Incremento,
34. ERROR,
35. }