

Universidad de San Carlos de Guatemala

Centro Universitario de Occidente División de Ciencias de la Ingeniería

Laboratorio de Lenguajes Formales y de Programación

Julio Fernando Ixcoy

Sección: A



Manual Técnico

Carlos Raúl Alberto López Peláez 202031871

Quetzaltenango, Agosto el 2023

Indice

Herramientas Usadas.....	1
Java Swing.....	1
Java.....	1
Graphviz.....	1
Estructura del proyecto.....	2
Main del proyecto.....	3
Apartado del analizador léxico.....	4
Método currentchar.....	5
Método Advance.....	5
Método newToken.....	5
Método readInteger.....	5
Método readLetter.....	5
Método cadena y cadenaComDod.....	5
Método fila.....	5
Método columna.....	5
Apartado Token.....	6
Apartado TipoTokens.....	7
Enum TipoAritmetico.....	7
Enum TipoAsignacion.....	7
Enum TipoComentario.....	8
Enum TipoComparacion.....	8
Enum TipoConstante.....	9
Enum TipoEspacio.....	9
Enum Tipoid.....	10
Enum TipoLogico.....	10
Enum TipoOtro.....	11
Enum PalabrasRes.....	11
Apartado Inicio.....	13
Método jMenuItem1ActionPerformed.....	16
Método dividirInstancias.....	16
Método colorPalabra.....	16
Generador de Imágenes.....	17
Método JlistValueChange.....	20
Método GenerarImage.....	20
Método reset.....	20
Métodos relacionados a tokens.....	20
Imagen.....	21
Reportes.....	22
Clase ButtonRender.....	23
Clase ButtonEditor.....	23
Clase NumeroLinea.....	23

Herramientas Usadas

Para la interfaz de la aplicación se uso java Swing, la lógica principal del programa esta construida en java y para generación de imágenes se uso graphviz

Java Swing

Java Swing es una biblioteca de interfaz de usuario para Java que permite a los desarrolladores crear aplicaciones de escritorio con elementos gráficos interactivos y personalizables, mejorando la experiencia del usuario en el entorno de escritorio.

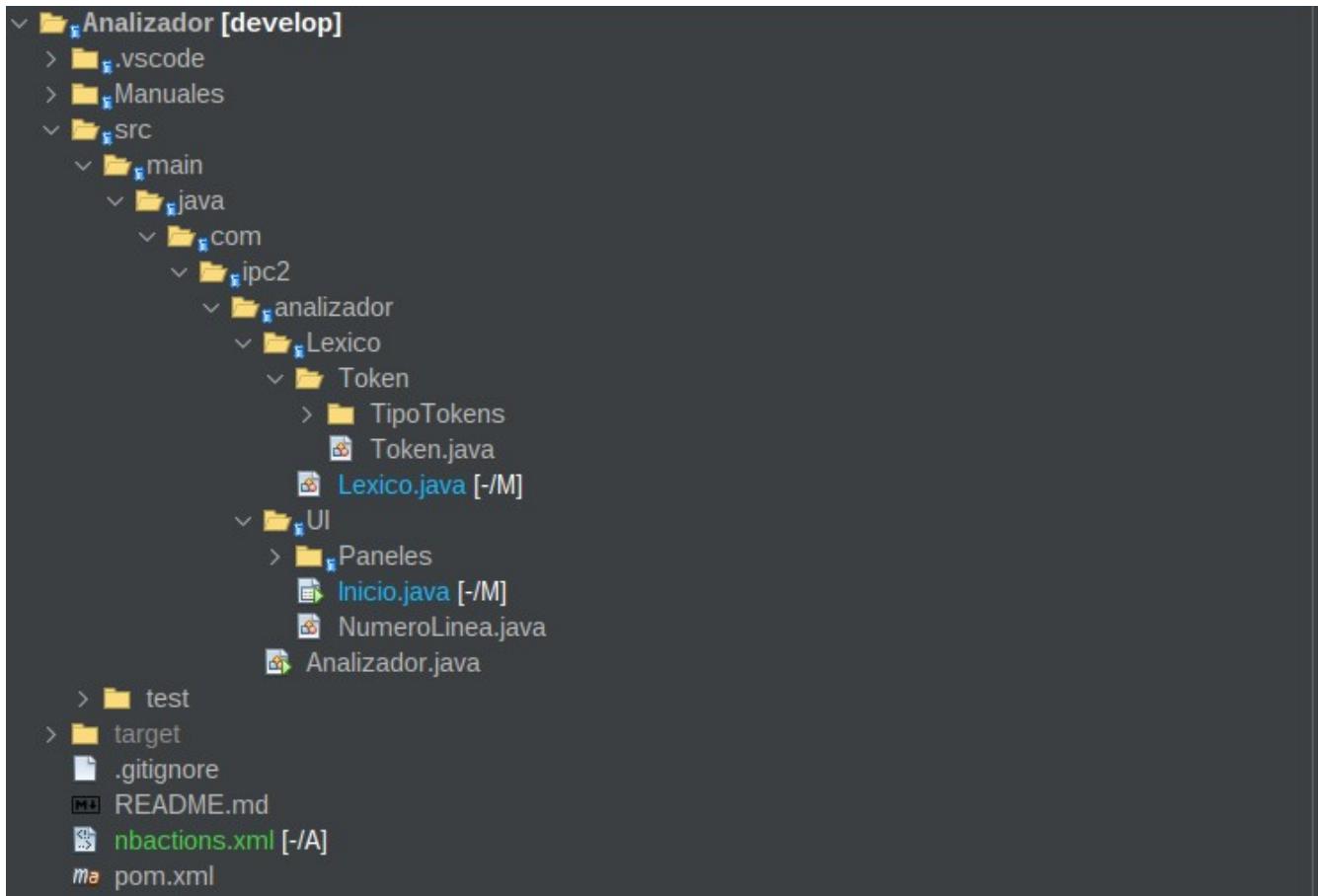
Java

Java es un lenguaje de programación orientado a objetos y altamente portátil, conocido por su seguridad, capacidad de programación multihilo y rica biblioteca estándar. Es utilizado en una amplia gama de aplicaciones, desde desarrollo de software hasta sistemas embebidos y desarrollo web.

Graphviz

Graphviz en Java es una herramienta que permite generar visualizaciones gráficas a partir del código Java, usando el lenguaje DOT para describir estructuras. Facilita la representación de datos complejos y relaciones en forma de diagramas y gráficos, ofreciendo flexibilidad y opciones de personalización en la disposición y aspecto visual de los elementos del gráfico.

Estructura del proyecto



Esta es la estructura del proyecto la parte principal o el main que es el analizador, después separado por dos carpetas, la parte encargada de la interfaz UI y la parte encargada de la lógica del analizador léxico carpeta Léxico cada una separada a su vez en otras carpetas y clases de la aplicación.

Main del proyecto

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 */

package com.ipc2.analizador;
import com.ipc2.analizador.UI.Inicio;

/**
 *
 * @autho carlos117
 */
public class Analizador {

    public static void main(String[] args) {
        new Inicio().setVisible(true);
    }
}
```

Apartado del analizador léxico

```
public class Lexico {  
    privat final String input;  
    public static int column ;  
    public static int #fila;  
    privat int position;  
    privat final ArrayList<Object> palabrasG;  
    e  
    public Lexico(String input) {  
        this.palabrasG = new ArrayList<>();  
        this.input = input;  
        this.position = 0;  
        this.column = 0;  
        this.#fila = 1;  
    }  
  
    privat char currentChar() {  
        e  
        ....}  
  
    privat void advanc () {  
        e  
        e  
        ....}  
  
    public Token nextToken() {  
        ....  
    }  
  
    privat Token readInteger() {  
        e  
        ....}  
  
    privat Token readLetter() {  
        e  
        ....}  
  
    public Token cadena(char curren ){  
        t  
        ....}  
  
    public int fila(){  
        ....}  
  
    public int column (){  
        a  
        ....}  
    }  
}
```

La clase Léxico se encarga de generar los tokens, guardar los lexemas, filas y columnas donde se encuentra cada lexema dentro del espacio de ingreso de información en inicio.

Método currentchar

Se encarga de retornar carácter por carácter del texto que sea ingresado en la clase.

Método Advance

Se encarga de de que currentchar() avance al siguiente carácter.

Método newToken

Se encarga de pasar en un ciclo while cada carácter que retorne el método currentchar() buscando si alguno de los caracteres pertenece a alguno de los tokens guardados.

Método readInteger

Se encarga de encontrar y analizar los caracteres numéricos y devolver sus tokens respectivos

Método readLetter

Se encarga de encontrar y analizar los caracteres de tipo strings y devolver sus tokens respectivos

Método cadena y cadenaComDad

Estos métodos se encargan de generar los tokens de las cadenas y sus respectivos lexemas.

Método fila

Devuelve la fila donde se encuentran los diferentes caracteres.

Método columna

Devuelve la columna donde se encuentran los diferentes caracteres.

Apartado Token

```
public class Token {  
    public Object type;  
    public String value;  
  
    public Token(Object type, String value) {  
        this.type = type;  
        this.value = value;  
    }  
}
```

Es una clase que contiene el valor del token y el lexema, al ser instanciada se guardan ambos valores.

Apartado TipoTokens

En este apartado todos los tokens están separados por clases enum, cada clase enum tiene sus tokens determinados estos tokens son los que se utilizan en la clase token y también en el apartado léxico

Aquí esta cada uno de los enums generados para los tokens

Enum TipoAritmetico

```
public enum TipoAritmetic {
    SUMA,    O
    RES,
    MULTI,
    DIVISIO
N,    EXP,
    MODULO
}
```

Enum TipoAsignacion

```
public enum TipoAsignacio {
    ASIG    n
}
```

Enum TipoComentario

```
public enum TipoComentario {
    COMENTARIO
}
```

Enum TipoComparacion

```
public enum TipoComparacion {
    OPRMYN,
    I, OPRMN,
    I, OPRMY,
    Q, OPRMN,
    Q, IGUAL,
    DIFERENT
}
```

Enum TipoConstante

```
public enum TipoConstante {  
    INTEGE e  
    R, DECIMA  
    L, CADEN  
    A, BOOLEA  
    N
```

Enum TipoEspacio

```
public enum TipoEspacio {  
  
    SPACE,  
    SALTO,  
    EOF  
}
```

Enum Tipold

```
public enum TipoIdentificado {  
    ID  
}
```

Enum TipoLogico

```
public enum TipoLogico {  
    AND,  
    OR,  
    NOT  
}
```

Enum TipoOtro

```
public enum TipoOtro {  
    COROP  
    N, CORCL  
    S, COMNOR  
    M, COMSIM  
    P, PAROP  
    N, PARCL  
    S, PUNTCOM  
    A, PUNTD0  
    B, COMA,  
    ERROR  
}
```

Enum PalabrasRes

```
public enum TipoPalabraRe {  
    AS,      S  
    ASSERT,  
    BREAK,  
    CLASS,  
    CONTINUE,  
    DEF,  
    DEL,  
    ELIF,  
    ELSE,  
    EXCEPT,  
    FALSE,  
    FINALL ,  
    FROM,  
    GLOBAL,  
    IMPORT,  
    IN,  
    IS,  
    LAMBDA,  
    NONE,  
    NONLOCAL,  
    PASS,  
    RAISE,  
    RETURN,  
    TRUE,  
    TRY,  
    WITH,  
    YIELD,  
    WHILE,  
    FOR,  
    IF  
}
```

Apartado Inicio

```
public class Inicio extends javax.swing.JFrame {
    /*           s      g.
     *   * Creates new form Inici
o   */
    public Inicio() {
        initComponents();
        Bumerolinea = new NumeroLinea(jTextPane
jScrollPane .setRowHeaderView(NumeroLine
Bumerolinea2 = new NumeroLinea(jTextArea
jScrollPane .setRowHeaderView(NumeroLinea
dapTokens = new HashMap<String, Integer>());
frame = this; p
reporte = new Reportes();
generador = new Generador();
int panelMaxWidth = 600;
int panelMaxHeight = 700;
jPanel1.setMaximumSize(new Dimension
(panelMaxWidth, panelMaxHeight));
    }

    /*
     * This method is called from within the constructor to initialize the f
orm.
     * WARNING: Do NOT modify this code. The content of this method is alway
s
     * regenerated by the Form Edito
r. */
    @SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        e           s
....}// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt
) { e           d
    ...
    private void jMenuItem2MouseClicked(java.awt.event.MouseEvent evt
) { e           d
        jMenuItem1.setVisible(false);
        getContentPane().removeAll();
        getContentPane().add(generado
        revalidate(); r);
        repaint();
    } t

    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt
) { e           d
        generador.reset();
        jMenuItem1.setVisible(true);
        getContentPane().removeAll();
        getContentPane().add(jPanel1);
        revalidate();
        repaint();
    } t
}
```

```

    privat void jMenuItem1ActionPerforme (java.awt.event.ActionEvent evt
) { e d
    try{

        JFileChooser fileChooser = new JFileChooser ();
        int result = fileChooser.showOpenDialog (frame);
        g
        if (result == JFileChooser .APPROVE_OPTIO ) {
            File file = fileChooser.getSelectedFile ();
            System.out.println ("Archivo: " + file.getName ());
            n e
            BufferedReader reader = new BufferedReader (new
FileReader(file)); r
r

            StringBuilde stringBuilde = new StringBuilde ();
            r r r
            String line = null;
            while ((line = reader.readLine()) != null) {
                stringBuilde.append(line)
                stringBuilde.append("\n");
                r
            }
            reader.close();

            tex = stringBuilde.toString();
            r
            System.out.println (tex
jTextPanel.setText(tex
t t);
        }
    }
    catch(Exception e){
        System.out.println ("Errorrrrrrr : " + e);
    }
}

privat void jMenuItem3MouseClicke (java.awt.event.MouseEvent evt
) { e d
    jMenuItem1.setVisible(false);
    getContentPane ().removeAll();
    //report.removeAll
    report.insertarInf (infoTable);
    getContentPane ().add(report);
    infoTable.clear();
    revalidate();
    repaint();
} t

public static HashMa todoToken (){
    return mapTokens;
}

public static List<List<Object>> infoT (){
    return infoTable;
}

```

```

public void dividirInstacias(){

    for (List<Object> list : tablaToken)
        for(Object value : list){
            if(value instanceof TipoIdentificador )
                for (int j &gt; 0; j < 2; j++) {
                    Object value2 = list.get(j);
                    if (!newLis .contains(value2)) {
                        newList .add(value2);
                        System.out.printl ("tabla: " + newList );
                        if (j == 1)n{
                            List<Object> valore  = Arrays.asList(list.
                                get(0),list.get(1));
                            newListofList .add(valores);
                            System.out.printl ("tabla lsitadelistas :
                                n           "
                                s);
                        }
                    }
                }
            }
        }
    }

}

}

}

}

}

.... System.out printl ("");
t. System.out printl ("Valor de lista Final =  + newListofList );
t. System.out printl ("");
t.   n

public void colorPalabra () {
    StyledDocumen doc = jTextField1.getStyledDocumen ();
    t
    String text1 = jTextField1.getText ();
    t

    SwingUtilitie.invokeLater(() -> {
        // Limpia los atributos de estilo del document
        doc.setCharacterAttribute (0, text1.length(), jTextField1.getStyle(
        "default$), true);

        for (List<Object> tabla : newListofLists)
            {
                ....
            }

            System.out.printl ("Valor de token en colores :  + tokenValue +
    " llave : " + tabla.get(0) ); "
            System.out.printl ();
            n

        }
    );
}
}

```

Esta clase cuenta con el primer Jframe que se ejecuta en la clase main de la aplicación , esta clase de inicio genera el Jframe junto a los jtextpane y el JButton se encarga básicamente de generar la interfaz, los eventos de presionar botones o menús ejecutan de forma directa los cambios, es decir el menú de cargar archivo se ejecuta directamente dentro del método del evento de presionar .

Método jMenuItem1ActionPerformed

Este método es el que detecta cuando es presionado el menú de carga de archivo, dentro de este esta toda la lógica para buscar y agregar la carga del archivo.

Método dividirInstancias

El método dividir instancias encargado de reorganizar el orden con que se colorean las palabras dependiendo el tipo de token que se crea.

Método colorPalabra

El método permite el coloreado de palabras en el área de ingreso texto, el coloreado de palabras esta definido por su token.

Toda la información recolectada en el jtextpane es mandada a los paneles de generación de imágenes y también al apartado de reportes.

Generador de Imágenes

```
/*
 * @autho carlos117
 */
public class Generador extend javax.swing JPanel {
    s     g.

    /*
     * Creates new form Generador
     */
    privat final JPanel jPanel ;
    privat JLabel label;
    privat JPopupMenu popupMenu;
    privat final HashMa <Object, Object> mapaToltal;
    e     p
    public Generador() {
        initComponent ();
        jPanel = new JPanel();
        label = new JLabel();
        popupMenu = new JPopupMenu();
        mapaToltal = new HashMa <>();
        setLayout(new BorderLayou ());
        add(jScrollPan l, BorderLayou .WEST);
        add(jPanel, BordetLayou .CENTER);
        t
    }

    /*
     * This method is called from within the constructor to initialize the f
     orm.
     * WARNING: Do NOT modify this code. The content of this method is alway
     s
     * regenerated by the Form Edito
     r.
     */
    @SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">
    privat void initComponent () {
        e           s
    ....}// </editor-fold>

    privat void jList1ValueChange (javax.swing.event.ListSelectionEven evt
) { e           d
        if (!evt.getValueIsAdjustin ()) {
            int selectedInde = jList1.getSelectedIndex();
            if (selectedIndex != -1) {
                String selectedElemen = jList1.getModel().getElementA
(selectedIndex); t
                System.out.printl ("Selected:  + selectedElemen
                if ("Identif cadore ".equals(select dElement))
                    sjPanel .removeAll();  {
                        jPanel .repain ();
                        Identifitado (Inicio.todoToken());
                        r/showPopupMenu
                        System.out.printl (
"Aqui en el generador esta el hash map  + mapaToltal
"                         mapaToltal.clear();   l);
                }
            ...
        }
    }
}
```

```

        }

        public void GenerarImage (Object Token){
            String prueba = Token.toString();
            System.out.printl ("valos prueba en metodo repetir   +prueba");
            int cantidadletr ="prueba.length();
            characharprueba;

            MutableNode[] nodesnew = new MutableNode[cantidadletra];

            for (int i = 0; i < cantidadletra; ++) {
                charprueba = prueba.charAt(i);

                if(i == cantidadletr -1){
                    nodesnew[i] = Factor .mutNod (String.valueOf (charprueb      add(
Shape.DOUBLE_CIRCL );           y           e           f           a)).add(
E );
                else{
                    nodesnew[i] = Factor .mutNod (String.valueOf (charprueb      add(Shape
.CIRCLE);           y           e           f           a)).add(
                }
            }

            MutableNode previousNod  = null;
            // Para llevar ue registro del nodo anterio
            r
            MutableGrap gnew = mutGraph("diagrama").setDirected(true)
            h .graphAttrs().add(Rank.dir(Rank.RankDi .LEFT_TO_RIGH ));
            r           T
            for (int i = 0; i < cantidadletra; ++) {
                MutableNode currentNod = nodesnew
                gnew.add(currentNod [i];
                e);
                if (previousNod != null && !previousNod .equals(currentNod)) {
                    previousNod .addLin (currentNod
                }   e           k           e);

                previousNod = currentNod
            }           e;

            // Renderizar el gráfico a una image
            ByteArrayOutputStrea outputStrea = new ByteArrayOutputStrea ();
            try {
                Graphviz.fromGraph(gne    render(Format.PNG).toOutputStrea
(outputStrea
w).           m
m);
            } catch (IOException ex) {
                }
            }

            // Convertir la imagen en un BufferedImag
            BufferedImag image = null;
            try {
                image = ImageI .read(new ByteArrayInputStream (outputStrea .
toByteArray()));   0           m           m
            } catch (IOException e) {
                }
            }

            label = new JLabel(new ImageIcon(image));
            //setLayout(new FlowLayout(FlowLayout.LEFT, 0,
jPanel.setLayout(new BoxLayout(jPanel1, BoxLayout.Y_AXIS));
jPanel.add(label);
jPanel.revalidate();
1
}

```

```
public void reset(){

    ...
}

public void PalabrasRes(HashMap mapTokens){
    p
    ...
}

public void Arimetico(HashMap mapTokens){
    p
    ...
}

public void Asignacion(HashMap mapTokens){
    p
    ...
}

public void Comentario(HashMap mapTokens){
    p
    ...
}

public void Comparacion(HashMap mapTokens){
    p
    ...
}

public void Constante(HashMap mapTokens){
    p
    ...
}

public void Logico(HashMap mapTokens){
    p
    ...
}

public void Identificado (HashMap mapTokens){
    r
    p
    ...
}

public void Otro(HashMap mapTokens){
    p
    ...
}

// Variables declaration - do not modify
private javax.swing.JList<String> jList1;
private javax.swing.JScrollPane jScrollPane1;
/* End of variables declaration */
}
```

Método JlistValueChange

Este método analiza los cambios hechos en la lista, es decir al momento de seleccionar algún apartado de la lista, este ejecuta una condición que se cumple y genera imágenes de los token que se hayan presionado en la lista

Método GenerarImage

Este método se encarga de recibir un valor char con el cual va generando cada nodo de la palabra en el grafico que termina siendo añadido en un label que se añade a el panel de GenerarGraficos.

Método reset

Se encarga simplemente de limpiar el panel y otras variables dentro de la clase;

Métodos relacionados a tokens

Estos métodos solo se encargan de revisar que tokens pertenecen a cada apartado del listado y se encargar de llamar al método generarImage para que se generen todas las imagenes.

Imagen

```
public class Imagen extends javax.swing.JFrame {
    public static JLabel label;
    /*
     * Creates new form Image
     */
    public Imagen() {
        initComponents();
        label = new JLabel();
    }

    /*
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        ...
    }
    ...

    public void GenerarImage (Object Token){
        String prueba = Token.toString();
        System.out.println ("valos prueba en metodo repetir   "+prueba);
        int cantidadletr = "prueba.length();
        char charprueba;

        MutableNode[] nodesnew = new MutableNode[cantidadletra];

        for (int i = 0; i < cantidadletra; ++){
            charprueba = prueba.charAt(i);

            if(i == cantidadletr -1){
                nodesnew[i] = Factor .mutNod (String.valueOf (charprueb      add(
Shape.DOUBLE_CIRCL );           y      e      f      a));
            } else{
                nodesnew[i] = Factor .mutNod (String.valueOf (charprueb      add(Shape
.CIRCLE);           y      e      f      a));
            }
        }

        MutableNode previousNod = null;
        // Para llevar ue registro del nodo anterio
        for (int i = 0; i < cantidadletra; ++){
            MutableNode currentNode = nodesnew
            gnew.add(currentNod      [i];
                      e);
            if (previousNode != null && !previousNod .equals(currentNode)) {
                previousNod .addLin (currentNod
                      e      k      e);
            }
            previousNode = currentNod
                      e;
        }

        // Renderizar el gráfico a una image
        ByteArrayOutputStream outputStrea = new ByteArrayOutputStream ();
        try {
            Graphviz.fromGraph(gne      render(Format.PNG).toOutputStrea
(outputStrea
                      w).
        } catch (IOException ex) {
        }

        // Convertir la imagen en un BufferedImage
        BufferedImage image = null;
        try {
            image = ImageI .read(new ByteArrayInputStream (outputStrea .
toByteArray()));
        } catch (IOException e) {
        }

        label = new JLabel(new ImageIcon(image));
        jPanel .setLayout(new BoxLayout(jPanel, BoxLayout.Y_AXIS));
        jPanel .add(label);
        jPanel .revalidate();
    }
}
```

La clase imagen solo es un Jframe que también genera imágenes de los token, es decir que se abre una nueva ventana y genera la imagen pedida por la clase reportes.

Reportes

```
public class Reportes extends javax.swing.JPanel {
    /* 
     * Creates new form Reporte
     */
    public Reportes() {
        initComponents();
    }

    /*
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        ...
    }

    public class ButtonRender extends JButton implements TableCellRenderer {
        ...
        public ButtonRender () {
            setOpaque(true);
        }

        @Override
        public Component getTableCellRendererComponent (JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) {
            setText ("Generar"); // Texto en el botón
            if (isSelected)
                setBackground (table.getSelectionBackground ());
            else {
                setBackground (table.getBackground ());
            }
            return this;
        }
    }

    public class ButtonEdito extends DefaultCellEditor {
        ...
        protected JButton button;
        private int clickedRow;
        private int clickedColumn ;
        ...
        public ButtonEdito () {
            super(new JTextField());
            button = new JButton ();
            button.setOpaque(true);

            // Acción al hacer clic en el botón
            button.addActionListener (new ActionListener () {
                ...
                @Override
                public void actionPerformed (ActionEvent e) {
                    ...
                    Object lexemaValue = jTable .getValueAt(clickedRow, 2);
                    // Para la columna "Lexem"
                    Imagen imagen = new Imagen();
                    imagen.GenerarImage (lexemaValue);
                    imagen.setDefaultCloseOperation (JFrame.DISPOSE_ON_CLOSE);
                    imagen.setVisible(true);

                }
            });
        }

        @Override
        public Component getTableCellEditorComponent (JTable table, Object value, boolean isSelected, int row, int column) {
            ...
            clickedRow = row;
            clickedColumn = column;
            button.setText ("Generar"); // Texto en el botón
            return button;
        }
    }
}
```

La clase reportes es un panel que contiene un jTable que tiene 5 columnas que son rellenadas con los datos de la clase inicio, en esta tabla se genera también un botón generar que permite la generación de gráficos para los lexema de la tabla, genera una ventana que permite la visualización de la grafica del lexema.

Clase ButtonRender

Esta clase se encarga de generar el botón dentro de las celdas de las tablas.

Clase ButtonEditor

Esta clase permite la funcionalidad del botón y contiene el evento que ejecuta la generación de la de la clase imagen

Clase NumeroLinea

Esta clase esta en la estructura de proyecto es la encargada de generar la enumeración de las dos áreas de texto, es llamada dentro de la clase inicio que es la clase que contiene las dos áreas de texto.

- 1) Importaciones de paquetes y clases necesarias: La clase comienza con una serie de importaciones de paquetes y clases de Java que serán necesarios para su funcionamiento. Estas importaciones incluyen clases relacionadas con la interfaz gráfica de usuario (GUI) y manipulación de texto.
- 2) Declaración de variables y constantes:
 - LEFT, CENTER y RIGHT: Son constantes que representan la alineación de los números de línea.
 - color1: Un objeto Color que define un color específico para el fondo de la línea actual.
 - OUTER: Un borde negro que se coloca a la derecha del panel para separar los números de línea del área de texto.
 - HEIGHT: Un valor constante que se utiliza como referencia para determinar la altura máxima del panel.
- 3) Declaración de variables de instancia:
 - component: Un objeto de tipo JTextComponent que representa el componente de texto al que se agregará este componente de números de línea.
 - updateFont, borderGap, currentLineForeground, digitAlignment, minimumDisplayDigits, lastDigits, lastHeight, lastLine: Variables utilizadas para almacenar diferentes configuraciones y estados internos de la clase.
 - fonts: Un mapa que almacena objetos FontMetrics para diferentes fuentes utilizadas en el componente de texto.
- 4) Constructores:

La clase define dos constructores sobrecargados. Ambos constructores toman un JTextComponent como argumento principal. El constructor sin parámetros llama al constructor con un valor predeterminado de 3 para minimumDisplayDigits. Los constructores configuran varias propiedades iniciales y agregan los escuchadores adecuados al componente de texto.
- 5) Métodos getters y setters:

La clase proporciona varios métodos para obtener y establecer diferentes propiedades y configuraciones, como `getUpdateFont()`, `setUpdateFont(boolean updateFont)`, `getBorderGap()`, `setBorderGap(int borderGap)`, etc.

6) Métodos para cálculos y dibujo:

- `setPreferredWidth()`: Calcula y establece el ancho preferido del componente de números de línea en función del número máximo de dígitos en el número de línea más alto.
-
- `paintComponent(Graphics g)`: Se sobrescribe este método para dibujar los números de línea y resaltar la línea actual.
-
- `isCurrentLine(int rowStartOffset)`: Verifica si la línea en la que está el cursor es la línea actual.
-
- `getTextLineNumber(int rowStartOffset)`: Devuelve el número de línea correspondiente a la posición `rowStartOffset` del componente de texto.
-
- `getOffsetX(int availableWidth, int stringWidth)`: Calcula el desplazamiento horizontal para alinear correctamente el número de línea.
-
- `getOffsetY(int rowStartOffset, FontMetrics fontMetrics)`: Calcula el desplazamiento vertical para la fila actual.
-
- `caretUpdate(CaretEvent e)`: Maneja eventos de cambio de posición del cursor para resaltar la línea actual.
- `changedUpdate(DocumentEvent e)`, `insertUpdate(DocumentEvent e)`, `removeUpdate(DocumentEvent e)`: Escuchan cambios en el documento para ajustar los números de línea.
-
- `documentChanged()`: Se encarga de manejar cambios en el documento que afecten el número de líneas y el ancho del componente.
- `propertyChange(PropertyChangeEvent evt)`: Responde a cambios en las propiedades del componente de texto, como el cambio de fuente.