MANUAL TÉCNICO

PRACTICA 1 LENGUAJES FORMALES DE PROGRAMACIÓN

INTRODUCCIÓN

La finalidad de todo manual técnico es la de proporcionar al lector las pautas de configuración y la lógica con la que se ha desarrollado una aplicación, la cual se sabe que es propia de cada programador; por lo que se considera necesario ser documentada.

Aclarando que este manual no pretende ser un curso de aprendizaje de cada una de las herramientas empleadas para el desarrollo del sitio, sino documentar su aplicación en el desarrollo del sitio. Para un mayor detalle acerca de cada una de las herramientas utilizadas, y su forma de operación y aplicación, se recomienda consultar los manuales respectivos de cada una de ellos.

```java
public class Practica1_LF {

    public static void main(String[] args) {
        FrameAnalizadorLexico frameAnalizadorLexico = new FrameAnalizadorLexico();
        frameAnalizadorLexico.setLocationRelativeTo(null);
        frameAnalizadorLexico.setVisible(true);
    }
}
```

El método main(), aparte que es el método donde inicializa la aplicación, lo que se ve al inicio del programa son unas ventanas de solicitud de algunos datos para el tamaño de la imagen.
Este método empieza instanciando un objeto de tipo FrameAnalizadorLexico, esl cual se establece que cuando se abra esta ventana pueda abrirse desde el centro de la pantalla y que sea visible.

```java
public FrameAnalizadorLexico() {
    initComponents();
    this.controlador = new Controlador(this);
    areaTextoNumeroLinea.setEditable(false);
    this.setTitle("Analizador Lexico");
    SolicitarFilasColumnas solicitarFilasColumnas = new SolicitarFilasColumnas(this, true, controlador);
    solicitarFilasColumnas.setLocationRelativeTo(null);
    solicitarFilasColumnas.setVisible(true);
}
```

Método constructor de FrameAnalizadorLexico, este constructor en su bloque de código tiene como instrucciones inicializar los componentes de la clase, instancia un nuevo objeto de tipo constructor, dicho objeto se le pasa como parámetro un FrameAnalizadorLexico, en la siguiente linea tenemos una instrucción que va a permitir que un área de texto no pueda ser editable, osea que, no se pueda escribir en esta área durante la ejecución del programa, la siguiente linea, establece el titulo a la ventana.
Luego en la siguiente linea se instancia un objeto de tipo SolicitarFilasColumnas, no es mas que una JdialogForm, esta ventana de dialogo es la que nos va a permitir solicitar las dimensiones de la imagen, de igual manera se establece que la ventana se abra en el centro de la pantalla y que sea visible.

CLASE Controlador.

Metodo constructor, este metodo solo recibe como parametro un objeto de tipo FrameAnalizadorLexico para poder tener el control de la parte grafica de la aplicación.

```java
public void imprimirBotonesImagen() {
    frameAnalizadorLexico.getPanelImagen().removeAll();
    frameAnalizadorLexico.getPanelImagen().setLayout(new GridLayout(frameAnalizadorLexico.getTamañoFilas(), frameAnalizado
    imagen = new Pixel[frameAnalizadorLexico.getTamañoFilas()][frameAnalizadorLexico.getTamañoColumnas()];//estabelciendo
    for (int i = 0; i < frameAnalizadorLexico.getTamañoFilas(); i++) {
        for (int j = 0; j < frameAnalizadorLexico.getTamañoColumnas(); j++) {
            imagen[i][j] = new Pixel();
            frameAnalizadorLexico.getPanelImagen().add(imagen[i][j]);
        }
    }
}
```

Método imprimirBotonesImagen(), este método permite mostrar los pixeles del tablero de la imagen a generar.

```
private void reseteandoImagen() {
    for (int i = 0; i < frameAnalizadorLexico.getTamañoFilas(); i++) {
        for (int j = 0; j < frameAnalizadorLexico.getTamañoColumnas(); j++) {
            imagen[i][j].setBackground(Color.WHITE);
        }
    }
    frameAnalizadorLexico.getPanelImagen().validate();
    frameAnalizadorLexico.getPanelImagen().repaint();
}
```

Método resetandoImagen(), este método establece el color blanco a cada pixel de nuestra imagen, por medio de dos ciclos for recorre todo el tablero, casilla por casilla.

```
public void abrirArchivo() {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);

    FileNameExtensionFilter filtro = new FileNameExtensionFilter("Archivos de texto", ".txt");
    int resultado = fileChooser.showOpenDialog(frameAnalizadorLexico);
    File archivoSeleccionado = fileChooser.getSelectedFile();
    obtenerContenidoArchvivo(archivoSeleccionado.getAbsolutePath());
}
```

Método abrirArchivo(), este método va a permitir ubicar un archivo de texto utilizando la herramienta FileChooser, esta herramienta permite navegar entre los archivo de una manera mas amigable visualmente hablando.

```
private void obtenerContenidoArchvivo(String rutaArchivo) {
    try {
        FileReader archivoALeer = new FileReader(rutaArchivo);
        BufferedReader bufferedReader = new BufferedReader(archivoALeer);
        StringBuilder contenido = new StringBuilder();//Variable de tipo StringBuilder que
        StringBuilder numeroLineaS = new StringBuilder();
        String linea;
        int numeroLineaI = 0;
        while ((linea = bufferedReader.readLine()) != null) {
            numeroLineaI++;
            numeroLineaS.append(numeroLineaI).append("\n");
            contenido.append(linea).append("\n");
        }
        frameAnalizadorLexico.getAreaTextoCodigo().setText(contenido.toString());
        frameAnalizadorLexico.getAreaTextoNumeroLinea().setText(numeroLineaS.toString());
    } catch (IOException e) {
        e.printStackTrace();
        frameAnalizadorLexico.getAreaTextoCodigo().setText("Error al abrir el archivo");
    }
    frameAnalizadorLexico.getAreaTextoCodigo().validate();
    frameAnalizadorLexico.getAreaTextoCodigo().repaint();
}
```

Método obtenerContenidoArchivo(), este método va a permitir obtener la información del archivo ubicado en el path que recibe como parámetro.

```
public void generarImagen() throws ExceptionsAnalizadorLexico {
    if (frameAnalizadorLexico.getAreaTextoCodigo() == null || frameAnalizadorLexico.getAreaTex
        throw new ExceptionsAnalizadorLexico("Ingrese codigo para poder analizarlo");
    }
      System.out.print(frameAnalizadorLexico.getAreaTextoCodigo().getText());
    controladorColumnas = 0;
    controladorFilas = 0;
    reseteandoImagen();
    obtenerPalabras();
}
```

Método generarImaget(), este método va a permitir llamar a una función que permite verificar cada palabra ingresada y poder pintar los pixeles para la imagen.

```
public void obtenerPalabras() {
    String texto = frameAnalizadorLexico.getAreaTextoCodigo().getText();//obteniendo
    String palabras[] = texto.split("\\s+");
    Operadores identificarOperadores = new Operadores(this);
    for (String palabra : palabras) {
        if (!palabra.isEmpty()) {
            if (!palabraIdentificada) {
                identificarOperadores.identificadorOperadorRelacionalComparacion(pal
            }
            if (!palabraIdentificada) {
                identificarOperadores.identificadorOperadoresAsignacion(palabra);
            }
            if (!palabraIdentificada) {
                identificarOperadores.identificadorOperadorAritmetico(palabra);
            }
            if (!palabraIdentificada) {
                identificarOperadores.identificadorPalabrasReservadas(palabra);
            }
            if (!palabraIdentificada) {
                identificarOperadores.identificadorTiposDatos(palabra);
            }
        }
        palabraIdentificada = false;
    }
}
```

Método obtenerPalabras(), este método permite obtener palabra por palabra del area de texto y poder mandarla a cada método para poder verificar que tipo de palabra pertenece.

```
public void pintarImagen(Color color, Token token) {
    imagen[controladorFilas][controladorColumnas].setBackground(color);
    imagen[controladorFilas][controladorColumnas].setToken(token);
}
```

Este método permite establecer un color a un pixel y tambien permite establecer un token al mismo pixel.
Después tendríamos los métodos getters y setters de la clase.

```java
//Metodo que permite identificar cada uno de los operadores aritmeticos
public void identificadorOperadorAritmetico(String texto) {
    Pattern pattern = Pattern.compile("[+\\-*/^]");//estableciendo que caracteres deben identificar como artimeticos
    Matcher matcher = pattern.matcher(texto);
    while (matcher.find()) {
        System.out.println("Operador: " + matcher.group());
        establecerColorIdentificadorAritmetico(matcher.group());
        controlador.setControladorColumnas(controlador.getControladorColumnas() + 1);
        if (controlador.getControladorColumnas() == controlador.getFrameAnalizadorLexico().getTamañoColumnas()) {
            controlador.setControladorColumnas(0);
            controlador.setControladorFilas(controlador.getControladorFilas() + 1);
        }
    }

}
```

Método que permite verificar si la palabra a analizar es de tipo operador aritmético, obtiene la palabra y hace ciertas validaciones para poder establecer el tipo de dato que es.

```java
//Metodo que permite identificar cada uno de los operadores aritmeticos
public void identificadorOperadorAritmetico(String texto) {
    Pattern pattern = Pattern.compile("[+\\-*/^]");//estableciendo que caracteres deben identificar como artimeticos
    Matcher matcher = pattern.matcher(texto);
    while (matcher.find()) {
        System.out.println("Operador: " + matcher.group());
        establecerColorIdentificadorAritmetico(matcher.group());
        controlador.setControladorColumnas(controlador.getControladorColumnas() + 1);
        if (controlador.getControladorColumnas() == controlador.getFrameAnalizadorLexico().getTamañoColumnas()) {
            controlador.setControladorColumnas(0);
            controlador.setControladorFilas(controlador.getControladorFilas() + 1);
        }
    }

}
        Token token = new Token(controlador.getControladorFilas(), controlador.getControladorColumnas(),
                operadorAritmetico, "Operador Aritmetico", COLOR_HEXADECIMAL_EXPONENTE);
        controlador.pintarImagen(color = color.decode(COLOR_HEXADECIMAL_EXPONENTE), token);
        controlador.setPalabraIdentificada(true);
    } else if (operadorAritmetico.equals("/")) {
        Token token = new Token(controlador.getControladorFilas(), controlador.getControladorColumnas(),
                operadorAritmetico, "Operador Aritmetico", COLOR_HEXADECIMAL_DIVISION);
        controlador.pintarImagen(color = color.decode(COLOR_HEXADECIMAL_DIVISION), token);
        controlador.setPalabraIdentificada(true);
    } else if (operadorAritmetico.equals("Mod")) {
        Token token = new Token(controlador.getControladorFilas(), controlador.getControladorColumnas(),
                operadorAritmetico, "Operador Aritmetico", COLOR_HEXADECIMAL_MODULO);
        controlador.pintarImagen(color = color.decode(COLOR_HEXADECIMAL_MODULO), token);
        controlador.setPalabraIdentificada(true);
    } else if (operadorAritmetico.equals("*")) {
        Token token = new Token(controlador.getControladorFilas(), controlador.getControladorColumnas(),
                operadorAritmetico, "Operador Aritmetico", COLOR_HEXADECIMAL_MULTIPLICACION);
        controlador.pintarImagen(color = color.decode(COLOR_HEXADECIMAL_MULTIPLICACION), token);
        controlador.setPalabraIdentificada(true);
    }
}
```

Método que permite verificar si la palabra a analizar es de tipo operador aritmético, obtiene la palabra y hace ciertas validaciones para poder establecer el tipo de dato que es.