Manual Tecnico – Analizador Lexico

Realizado por: Javier Alejandro Mérida Gómez – 202230368

Descripción:

¡Bienvenido al "Analizador Léxico"!
Este sistema está diseñado para analizar texto, escaneando el código fuente y generando una imagen en forma de cuadrícula, donde cada cuadro se pinta según el orden y el tipo de contenido del código fuente.

Requisitos del Sistema

- Plataforma: El analizador Léxico fue desarrollado para ejecutarse en sistemas operativos Windows y Linux.
- Entorno de Desarrollo: Se utilizó Apache NetBeans IDE 20 para el desarrollo del código y dependencias de Graphviz.
- Lenguaje de Programación: El analizador Léxico se implementó en Java (Programación Orientada a Objetos), versión 17.
- Sistema de Desarrollo: El desarrollo se llevó a cabo en un entorno Ubuntu Linux.
- Link del repositorio: https://github.com/JavierMeridaLk/PracticaUnoLenguajes

Componentes del analizador Léxico

Backen:

- Se encarga de toda la logica y procedimientos/metodos los cuales el usuario no ve, nuestro backen cuenta con 4 clases, las cuales son: Analizador, Token,Image y EstiloTabla.
- Clase “Analizador”:
    - La clase Analizador es responsable de procesar el código fuente ingresado, dividiendo el texto en tokens. Además, la clase permite la lectura de archivos de texto que contienen el código fuente y la creación de reportes sobre los tokens analizados.
    - Constructor Analizador(ReportesDialog reporte): Inicializa la clase con un objeto ReportesDialog, que se utiliza para mostrar los reportes generados durante el análisis léxico.
    - analizarCodigoFuente(String texto, int cantidadDeToken, JPanel panel, JToggleButton boton, JToggleButton botonReporte): Este método es el núcleo del analizador léxico. Toma el código fuente como una cadena de texto, lo divide en tokens, y los visualiza en una cuadrícula en un JPanel.

- También genera una tabla de reportes que detalla cada token, su posición en el código, y el cuadro de la cuadrícula que ocupa.
- Si el número de tokens especificado es mayor o igual al número de palabras en el código fuente, el análisis se completa y la cuadrícula se rellena con tokens o celdas vacías según sea necesario.
- AbrirChose(): Permite al usuario seleccionar un archivo desde el sistema de archivos, que será leído posteriormente para ser analizado.
- leer(JTextPane text): Lee el contenido de un archivo de texto seleccionado y lo muestra en un JTextPane. Este contenido es el que se analizará en el método analizarCodigoFuente.

- Clase “Token”:
  - La clase Token es una extensión de JLabel que se utiliza para representar y gestionar tokens en una interfaz gráfica.
  - Atributos de Color: La clase define varios colores en formato hexadecimal que se asocian con diferentes tipos de tokens (como identificadores, operadores, palabras reservadas, tipos de datos, etc.).
  - Constructor: El constructor de la clase añade un MouseListener que detecta clics del usuario en el token y muestra información adicional sobre el mismo cuando es clicado.
  - Método nuevoToken(String palabra): Este método toma una cadena (palabra) y la analiza para determinar su tipo (por ejemplo, si es un número, operador, palabra reservada, etc.). Luego, asigna un color específico al token basado en su tipo y lo personaliza visualmente.
  - Método mostrarInformacion(String token): Este método se encarga de crear una representación gráfica del token como un autómata finito utilizando Graphviz. Genera un archivo DOT que representa el autómata y lo convierte en una imagen PNG. Luego, muestra esta imagen en un cuadro de diálogo junto con la información de fila y columna del token en la cuadrícula.
  - generarDialog(File archivoImagen): Muestra un cuadro de diálogo con la imagen del autómata y la información del token.
  - generarCadena(String palabra): Construye la cadena que representa la estructura del autómata en formato DOT.
  - generarNombre(String extension): Genera un nombre de archivo basado en la fecha y hora actuales.

- Clase “Imagen”:
  - La clase Imagen es responsable de manejar la creación de una cuadrícula y la exportación de su contenido como una imagen en formato PNG

- Método crearCuadricula(JPanel panel): Solicita al usuario el número de filas y columnas para crear una cuadrícula dentro de un Jpanel. Si el usuario ingresa valores válidos, el método configura el JPanel con un GridLayout basado en las dimensiones proporcionadas y devuelve un arreglo con los valores de filas y columnas. Si el usuario ingresa datos inválidos o cancela la operación, se muestra un mensaje de error y el método devuelve un arreglo indicando que ocurrió un error.
- Método exportarImagen(JPanel ImgPanel): Captura la imagen del contenido del JPanel proporcionado (ImgPanel) y la guarda como un archivo PNG en una carpeta llamada "Imagenes". El método solicita al usuario un nombre para la imagen y guarda el archivo en la carpeta especificada. Si la exportación es exitosa, se muestra un mensaje de confirmación. En caso de error, se muestra un mensaje de error.

- Clase “EstiloTabla”:
  - La clase EstiloTabla extiende la clase DefaultTableCellRenderer y se utiliza para personalizar la apariencia de las celdas en una tabla (JTable). Esta clase redefine el método getTableCellRendererComponent, que es responsable de cómo se renderizan las celdas.
  - JTextArea para las celdas: En lugar de usar un componente estándar para las celdas de la tabla, se utiliza un JTextArea. Esto permite que el texto dentro de cada celda se ajuste automáticamente al ancho de la celda.
  - Personalización del fondo y el texto: El fondo y el color del texto cambian dependiendo de si la celda está seleccionada o no.
  - Fuente y borde: La fuente de texto en las celdas se mantiene consistente con la fuente de la tabla, y se aplica un borde vacío alrededor del JTextArea para proporcionar un pequeño margen interno.

Fronted:

- Se encarga de toda la parte visual, todo lo que el usuario veo y tiene interacciones, en nuestra aplicación contamos con un Fromulario Frame (framePrincipal) y un JDialog (ReportesDialog).
- FramePrincipal:
  - Es la ventana principal del programa, encargada de gestionar la interfaz gráfica de usuario (GUI) para realizar el análisis léxico, mostrar los resultados y permitir la exportación de imágenes.
  - Contiene varios JToggleButton para iniciar, analizar, cargar archivos, exportar imágenes y ver reportes.
  - Utiliza un JPanel (ImgPanel) para mostrar visualmente los resultados del análisis.

- JTextPane (panelTexto y TextPanelLineas) para ingresar y mostrar el código fuente con la numeración de líneas.
- limpiar(): Restablece todos los componentes de la ventana, limpiando el contenido y desactivando botones.
- iniciarTablero(): Prepara el tablero para un nuevo análisis, creando una cuadrícula de acuerdo a las especificaciones y habilitando la edición en el panel de texto.
- botonExportarActionPerformed(): Exporta una imagen del contenido visualizado en ImgPanel.
- jToggleButton3ActionPerformed(): Ejecuta el análisis léxico del código fuente ingresado y visualiza los resultados.
- jToggleButton4ActionPerformed(): Permite cargar un archivo de texto para ser analizado.
- DocumentListener en panelTexto: Este listener actualiza el conteo de líneas conforme se edita el texto en el JTextPane.
- CaretListener en panelTexto: Muestra la posición actual del cursor (fila y columna) en las etiquetas LabelFila y LabelColumna.

- JDialog dialogReportes:
  - Diseñado para mostrar un diálogo de reportes en una aplicación. Esta clase extiende JDialog, lo que significa que representa una ventana secundaria dentro de la interfaz gráfica.
  - Método agregarTabla: Este método permite añadir una tabla (JTable) al panel jPanel1. La tabla se coloca dentro de un JScrollPane, que proporciona barras de desplazamiento automáticamente si la tabla excede el tamaño visible. El panel se reorganiza con un BorderLayout, lo que asegura que el JScrollPane ocupe todo el espacio disponible. Finalmente, se actualiza el panel (revalidate() y repaint()) y se ajusta el tamaño de la ventana nuevamente (pack()).