Manual técnico

PROYECTO_2_LENGUAJES

CENTRO UNIVERSITARIO DE OCCIDENTE

CUNOC

Desarrollador: Anthony Gutiérrez

03/11/2025

1. Información General

Nombre del Programa: Practica_1_lenguajes
Versión: 1.0
Descripción:
Esta es una aplicación para el análisis léxico de texto, puede reconocer patrones de texto y errores mediante la creación de tokens de distintos tipos definidos por un archivo Json que tiene las instrucciones para reconocer todos los patrones que se evalúan en el programa, puede buscar cadenas de texto introducidas por el usuario y generar reportes de la cantidad de tokens validos o errores que se obtengan tras analizar el texo.

El programa además analiza texto de forma sintáctica, revisando que las declaraciones escritas estén dentro de las reglas de la gramática que se utilizan en este programa

2. Requisitos del Sistema

Hardware

- **Procesador:** Intel Core i3 o superior
- **Memoria RAM:** 4GB o más
- **Almacenamiento disponible:** -- KB o más
- **Tarjeta gráfica:** Compatible con Java

Software

- **Sistema Operativo:** Windows, macOS o Linux
- **Java Development Kit (JDK):** Versión 21 o superior

3. Instalación

1. **Descargar el programa:**
   Asegúrate de que los archivos del programa estén disponibles en tu computadora.
2. **Instalar Java:**
   Descarga e instala la última versión del Java Development Kit (JDK) desde la página oficial de Oracle o desde el gestor de paquetes de tu sistema operativo.
3. **Ejecutar el programa:**
   - Entra a la terminar de tu ordenador y ejecuta el siguiente comando dentro de la carpeta target del archivo
     Practica_1_lenguajes-1.0-SNAPSHOT-jar-with-dependencies

2. Clases Principales

- **Practica_1_lenguajes.java**

Esta clase es la que lleva el método main, es imprescindible para el programa.

- **FileHandler**

Esta clase es la encargada de cargar y guardar los archivos de texto hacia el programa.

1. *public void buscarArchivo()*
2. *private String getFileContent()*
3. *public void sobreescribirContenido(String nuevoConetnido)*
4. *public void guardarComo(texto)*

- **TextPainter**

Esta clase es la encargada de crear los estilos para el JTextPane y también la encargada de "colorear" el pane, ya sean las letras o el fondo.

- **WordFinder**

Clase encargada comparar carácter por carácter lo que el usuario esta buscando con el teto en el pane.

- **ConfigLoader**

Esta clase es la encargada de leer el JSON que se encuentra en src/main/resources y lo carga en la clase **Config.java** la cual es la encargada de encapsular todos los datos del json y así poder ser utilizados en el analizador léxico.

- **Token**

Esta clase solo se encarga de encapsular todos los datos del token, como el tipo (**TokenTipo.java; es un enum**) o la clase **Posicion.java** que es la encargada de generar la posición del lexema dentro del pane

- **ReportesExporter**

Esta clase se encarga de guardar los reportes en un archivo CSV y lo guarda en el dispositivo.

- **lexer.flex**

Este archivo flex es el encargado de crear un archivo Lexer.java que actuara como el analizador léxico de este programa, es decir, como un automata

1. private void guardarToken(Token token)
   Este método se encarga de guardar un token ya formado, en un ArrayList<Token> que se inicia como variable global de la clase
2. public List<Token> getTokens()
   Metodo que obtene el listado de tokens evaluados con las reglas del lexer

- **SyntacticAnalizer**

Esta clase es la encargada de evaluar las entradas de texto, pues si estas son lexicamente correctas se pueden analizar sintácticamente, se basa en las reglas de una tabla LL(1), matrices y pasar de estado a estado mediante varios métodos

1. inicializarTabla();
2. set();
3. parseStatement();

   Lee declaraciones hasta que sea correcta o incorrecta
4. parseAll();

   Lee todas las declaraciones, cada una las separa con parseStatement();

- **LexicalHighlighter**

Esta clase es la que se encarga de llamara al anilizador léxico (Lexer) o el analizador sintactico (SyntacticAnalyzer)

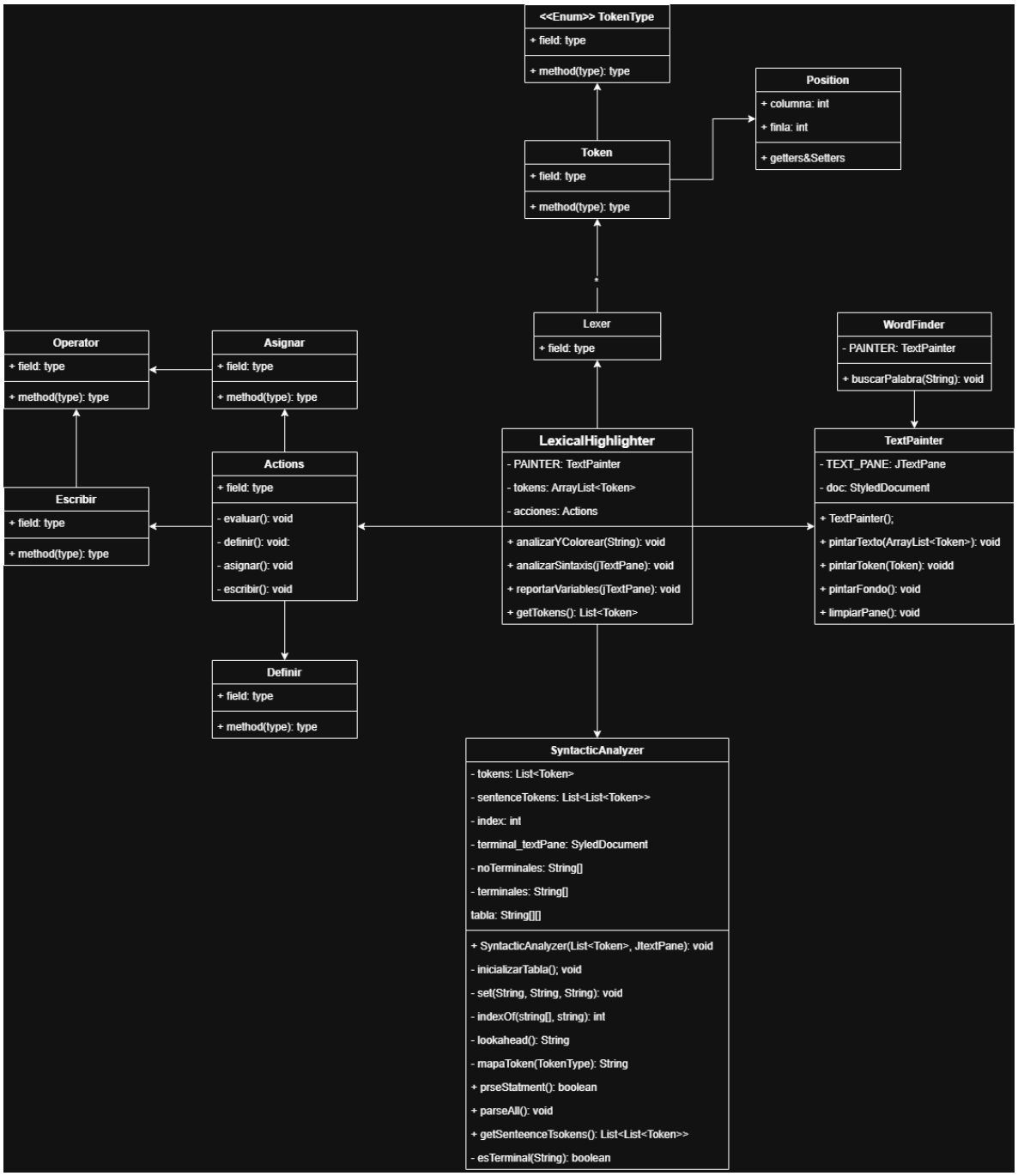
- **Actions**

Esta clase es la encargada de evaluar las declaraciones ya separadas por el analizador léxico y determinar que acción se quiere tomar entre: definir, asignar o escribir y hacer los arreglos necesarios para que la clase correspondiente solo obtenga los tokens que son necesarios para realizar la acción.

1. Evaluar()
2. escribir()
3. definir()
4. asignar()
5. comprobarExistencia():
   Comprueba la existencia de una variable

- **Asignador**
- **Escribir**
- **Operator**
- **Frontend**

Las calses encargadas de mostrar visualmente lo que sucede al usuario todo dentro de un jDesktopPane que contiene a los internalFrame para que la navegación sea mas amigable con el usuario


<<Enum>> TokenType
+ field: type
+ method(type): type
Position
+ columna: int
+ finla: int
+ getters&Setters
Token
+ field: type
+ method(type): type
*
Lexer
+ field: type
WordFinder
- PAINTER: TextPainter
+ buscarPalabra(String): void
Operator
+ field: type
+ method(type): type
Asignar
+ field: type
+ method(type): type
Actions
+ field: type
- evaluar(): void
- definir(): void:
- asignar(): void
- escribir(): void
LexicalHighlighter
- PAINTER: TextPainter
- tokens: ArrayList<Token>
- acciones: Actions
+ analizarYColorear(String): void
+ analizarSintaxis(jTextPane): void
+ reportarVariables(jTextPane): void
+ getTokens(): List<Token>
TextPainter
- TEXT_PANE: JTextPane
- doc: StyledDocument
+ TextPainter();
+ pintarTexto(ArrayList<Token>): void
+ pintarToken(Token): voidd
+ pintarFondo(): void
+ limpiarPane(): void
Escribir
+ field: type
+ method(type): type
Definir
+ field: type
+ method(type): type
SyntacticAnalyzer
- tokens: List<Token>
- sentenceTokens: List<List<Token>>
- index: int
- terminal_textPane: SyledDocument
- noTerminales: String[]
- terminales: String[]
tabla: String[][]
+ SyntacticAnalyzer(List<Token>, JtextPane): void
- inicializarTabla(); void
- set(String, String, String): void
- indexOf(string[], string): int
- lookahead(): String
- mapaToken(TokenType): String
+ prseStatment(): boolean
+ parseAll(): void
+ getSenteenceTsokens(): List<List<Token>>
- esTerminal(String): boolean