Manual Técnico

Analizador Léxico

Versión 1.0

Requisitos:

Java Development Kit 17

Al menos 1 GB RAM

Procesador con al menos 1 GHz

Conocimientos en manejo de proyectos Java Maven

Conocimientos en Java Swing

Creado por Josecarlos Emanuel Hernández Chuvac

Dependencias: graphviz-java maven 0.18.1

Introducción

Este manual es una guía para entender el funcionamiento del programa Analizador Léxico, este manual esta enfocado a futuros desarrolladores que deseen actualizar el código del programa.

En este manual encontraran información del funcionamiento de las clases y métodos mas importantes del programa.

Índice

Inicio del programa

FramePrincipal extends JFrame

Es el unico JFrame del programa es el encargado de mostrar y administrar contenido en los paneles internos, contiene dos JPanel. Uno encargado de mostrar la barra del botones “Ingresar Archivo” y “Exportar Imágen” (panelAcciones) y el segundo es el encargado de mostrar todo el contenido del programa (panelContenido).

solicitarCuadricula(): void

Metodo que genera un JDialog y solicita la dimension de la cuadricula deseada

botonIngresarArchivoActionPerformed(java.awt.event.ActionEvent evt): void

Se encarga de pedirle al usuario un archivo de texto y colocar el texto en el área del código fuente, utiliza la clase ExportImportFiles

botonExportarPngActionPerformed(java.awt.event.ActionEvent evt): void

Se encarga de pedirle al usuario un directorio para guardar la imagen y guarda la imagen de la cuadricula, utiliza la clase ExportImportFiles

botonReportesActionPerformed(java.awt.event.ActionEvent evt): void

Se encarga de pedirle al usuario un directorio para guardar la imagen y guarda la imagen de la cuadricula, utiliza la clase ExportImportFiles

AnalizadorLéxico

Es la clase encargada de identificar todos los tokens existentes en el código de entrada. Para ello hace uso de un List<Tokens> que guarda secuencialmente los tokens encontrados

AnalizadorLexico(String textoIngresado): constructor

Recibe el texto ingresado en el área del codigo fuente.

charActual(): char

Regresa el carácter actual en base al currentIndex

avanzar(): void

Avanza al siguiente indice

retroceder(int marcadorIndice): void

Retrocede el indice a uno indicado por el marcador

generarTokens(): List<Token>

Metodo encargado de analizar todo el texto y regresar los tokens hallados

-> return lista de token

identificarPalabra(): Token

Metodo encargado de identificar todo tipo de cadenas que inicien con una letra.

Puede devolver PALABRA_RESERVADA, OPERADOR_ARITMETICO_MODULO, OPERADOR_LOGICO_AND, OPERADOR_LOGICO_OR, OPERADOR_LOGICO_NOT, BOOLEANO o IDENTIFICADOR;

identificarPalabraReservadaEspecial(StringBuilder string): Token

Determina si el token es una palabra reservada que contiene un punto (Console.Writeline o Console.ReadLine)

-> return token palabra reservada especial o null

identificarFuncionSquare(StringBuilder string): Token

Identifica si es un token de la funcion Square.Color

-> return tokenSquare o null

identificarParametrosSquare(StringBuilder string): Token

Identifica los parametros de la funcion Square.Color

-> return tokenSquare o null

identificarFilaColumnaSquare(StringBuilder string, String color): Token

Identifica si la función Square.Color contiene parametros de fila y columna

-> return tokenSquare o null

identificarNumero(): Token

Identifica si es un numero entero o numero decimal

-> return Token numero entero o numero decimal

identificarSimbolo(): Token

Identifica el simbolo ingresado

puede devolver uno de los Tokens: PARENTESIS, LLAVE, CORCHETE, COMA, PUNTO o null

identificarOperadorOAsignacionCompuesta(): Token

Identifica si esun operador aritmetico o una asignacion compuesta

-> return token aritmetico o token asignacion compuesta

identificarOperadorOAsignacionSimple(): Token

Identifica si es un operador relacional o un operador de asignacion compuesto

-> return token operador relacional o asignacion simple

identificarCaracterOComentario(): Token

Identifica si es un caracter o un comentario

-> return token caracter o token comentario

identificarCadena(): Token

Identifica si es una cadena de texto encerrada por commillas

-> return tokenCadena

Token

Clase encargada de almacenar toda la informácion del token encontrado

Token(TokenType type, String value, int fila, int columna): constructor

Recibe los datos del token

Type - > enum del tipo de token identificado

Value - > Cadena correspondiente

Fila - > Posicion de la fila del token en el texto

Columna - > Posicion de la columna del token en el texto

Token(TokenType type, String codigoColor, String value, int fila, int columna): constructor

Recibe los datos del token especificando un codigo de color distinto al del token

Type - > enum del tipo de token identificado

CodigoColor - > color en formato hexadecimal

Value - > Cadena correspondiente

Fila - > Posicion de la fila del token en el texto

Columna - > Posicion de la columna del token en el texto

getReporte(): String[]

Devuelve un array con los datos del token en formato reporte

TokenType: enum

Enum que se encarga de definir todos los tipos de token soportados, guardando su codigo de color correspondiente a cada token

TokenType(String codigoColor): constructor

Recibe el color en formato hexadecimal

ExportImportFiles

Clase encargada de realizar todas las interacciones que involucren exportar e importar archivos

mostrarDialog(int mode) throws SolicitudCanceladaException : void

Muestra un file chooser que permite al usuario seleccionar el archivo o carpeta

Mode - > modo de fileChooser (Archivos o directorios)

-> throws SolicitudCanceladaException si se cancela la solicitud

recibirArchivoEntrada(): String

Lee el archivo de entrada y lo convierte a una cadena de texto

-> return string texto del archivo de entrada

createImage(JPanel panel): void

se encarga de crear la imagen en base a un Jpanel ingresado

Panel - > panel cuadricula