MANUAL TECNICO

PRACTICA 1

LENGUAJES FORMALES Y DE PROGRAMACIÓN

William Alexander
Miranda Santos
201930967

1. PAQUETES

1.1. Archivos

Este paquete contiene clase que se encargan exclusivamente en tratar todo lo relacionado a archivos, es decir, a leer y devolver una lista de filas de un archivo de texto, o bien, exportar un archivo de texto modificado.

1.2. Clases

Contiene las clases encargadas del funcionamiento optimo de los automatas, desde el analizador encargado de separar cada palabra proveniente de un archivo de texto, como analizar cada una de ellas y clasificarlas en alguno de los tipos de tokens existentes (identificador, número, decimal, puntuación, operador, agrupación) o en un error.

1.3. Controladores

Es el encargado de almacenar todos los controladores que contienen la lógica pura del programa, es decir, métodos de ordenamiento, de analisis, tratamiento de accesos a distintas ventanas, entre otros.

1.4. Enums

Es el paquete que contiene las clases ENUM, en este caso, se utilizo para diferenciar cada tipo de token.

1.5. GUI

Contiene todos los objetos del tipo GUI, es decir, de la interfaz gráfica, el cual se relaciona directamente con las clases encargadas de realizar toda la lógica correspondiente.

1.6. Principal

Es el encargado de almacenar las clases esenciales para el funcionamiento del sistema, ya que dichas clases son las encargadas de arrancar el mismo y conectar la logica con la interfaz de usuario.

1.7. Objetos

Es el encargado de almacenar los objetos utilizados en el programa, objetos necesarios para el almacenamiento dinamico de los valores trabajados dentro del programa.

2. CLASES

2.1. Archivos

2.1.1 EscritorArchivosTexto: Es el encargado de redactar un archivo txt y guardarlo en una carpeta en especifica.

2.1.2 LectorArchivosEnTexto: Es el encargado de extraer el texto proveniente de un archivo de texto y de devolver su contenido legible.

2.2. Clases

2.2.1 Analizador: Es el encargado de realizar el análisis del texto proveniente del lector de archivos de texto, ya que debe separar las palabras y tenerlas listas para ser analizadas por el lector.

2.2.2 InformaciónTokens: Es el encargado de almacenar toda la información crucial del automata dentro del programa, ya que incluye al abecedario del mismo, y cada matriz de transiciones correspondiente a cada uno de los tokens.

2.2.3 Lector: Es el encargado de realizar los movimientos dentro del automata, al analizar un texto en especifico.

2.3. Controladores

2.3.1 ControlPrincipal: Es el encargado de realizar todas las funciones correspondientes a la lógica de la clase Principal y del programa en general.

2.4. Enums

2.4.1 TipoToken: Es el encargado de otorgar distintos tipos de Tokens.

2.5. GUI

2.5.1 BusquedaGUI: Es la ventana que corresponde al a búsqueda de los patrones.

2.5.1 InicioGUI: Es la ventana encargada del inicio de la aplicación.

2.5.1 PrincipalGUI: Es la ventana encargada de dar el acceso a todos los reportes, editar, guardar y abrir archivos de texto para ser evaluados.

2.5.1 ReportesGUI: Es la ventana encargada de dar el acceso a todos los reportes registrados.

2.5.1 TablaResultados: Es la ventana encargada de mostrar los resultados proveniente de los reportes.

2.6. Principal

2.6.1 NewMain: Es el encargado de la creación y ejecución de la clase Principal para el inicio del programa.

2.6.2 Principal: Es el encargado de dar la conexión entra clases e interfaz de usuario y controladores.

2.7. Objetos

2.7.1 Contable: Es el encargado de almacenar tanto el valor de un token, como la cantidad de los mismo que se encuentran registrados como tokens.

2.7.2 Texto: Es el encargado de almacenar tanto el valor de un texto, como la columna y fila a la cual pertenece, esto para luego ser analizado como Token o Error.

2.7.3 Token: Es el encargado de almacenar tanto el valor de un texto, como la columna y fila a la cual pertenece, proveniente del análisis realizado a los textos.

3. METODOS

3.1 EscritorArchivosTexto

3.1.1 guardarArchivoTexto(): Recibe(String texto), Es el encargado de realizar el proceso de guardar un texto en un archivo de texto.

3.2 LectorArchivosEnTexto

3.2.1 leerFichero(): Recibe(File archivo), Es el encargado de realizar el proceso de obtener el texto proveniente de un archivo de texto.

3.3 Analizador

3.3.1 evaluarTextoTotal(): Recibe(ArrayList<String> filas), Es el encargado de realizar el proceso de evaluar fila por fila, del texto proveniente del lector de archivos en texto, separándolo en textos más pequeños, obteniendo la fila y columna de cada texto.

3.3.2 evaluarFilaTexto(): Recibe(String texto, int fila), Es el encargado de realizar el proceso de evaluar una fila, del texto proveniente del lector de archivos en texto, separándolo en textos más pequeños, obteniendo la fila y columna de cada texto.

3.3.3 separar(): Recibe(String texto, int fila), Es el encargado de realizar el proceso de separar en textos más pequeños, el texto proveniente de una fila.

3.3.4 verificar(): Recibe(ArrayList<Texto> textosSeparados), Es el encargado de verificar cada uno de los textos previamente separados, para clasificarlos entre tokens o error.

3.3.5 verificarTipoToken(): Recibe(String texto, ArrayList<Token> tokens), Es el encargado de comparar el texto con algún tipo de token valido, para luego evaluar si dicho texto es un token valido.

3.3.6 evaluarIdentificador(): Recibe(String texto, ArrayList<Token> tokens), Es el encargado de verificar si el token ingresado corresponde a un tipo Identificador.

3.3.7 evaluarNumero(): Recibe(String texto, ArrayList<Token> tokens), Es el encargado de verificar si el token ingresado corresponde a un tipo Numero.

3.3.8 evaluarDecimal(): Recibe(int caracteres, String texto, ArrayList<Token> tokens), Es el encargado de verificar si el token ingresado corresponde a un tipo Decimal.

3.3.9 evaluarPuntuacion(): Recibe(String texto, ArrayList<Token> tokens), Es el encargado de verificar si el token ingresado corresponde a un tipo Puntuacion.

3.3.10 evaluarOperador(): Recibe(String texto, ArrayList<Token> tokens), Es el encargado de verificar si el token ingresado corresponde a un tipo Operador.

3.3.11 evaluarAgrupacion(): Recibe(String texto, ArrayList<Token> tokens), Es el encargado de verificar si el token ingresado corresponde a un tipo Agrupación.

3.3.12 evaluarError(): Recibe(String texto, ArrayList<Token> tokens), Es el encargado de verificar si el token ingresado corresponde a un Error.

3.3.13 evaluarCHAR(): Recibe(char charEvaluado, String[] datos), Es el encargado de verificar si el carácter ingresado, se encuentra dentro del conjunto de caracteres.

3.3.14 redimensionarTexto(): Recibe(String texto, int caracteres), Es el encargado de redimensionar un texto, es decir, de escribir un nuevo texto debido a los restos o sobrantes de uno anterior.

3.3.14 darSeguimiento(): Recibe(int caracteres, String texto, ArrayList<Token> tokens), Es el encargado de reiniciar la búsqueda de la evaluación de un texto sobrante.

3.4 Lector

3.4.1 iniciarLector(): Recibe(String texto, TipoToken tipo, ArrayList<Token> tokens), Es el encargado de realizar el proceso de evaluar carácter por carácter, según el tipo de token al que posiblemente corresponda.

3.4.2 verificarTokenValido():Recibe(char actual), Es el encargado de realizar el proceso de evaluar un carácter seleccionado, para revisar si corresponde a un movimiento valido dentro de nuestro autómata, siguiendo las transiciones previamente definidas.

3.4.3 obtenerTipoCaracter():Recibe(char actual), Es el encargado de devolver un valor representativo en las transiciones, si este forma parte de algún alfabeto en específico, según sea el tipo de token que se esta evaluando.

3.4.4 definirEstadosYFuncion():Recibe(char actual), Es el encargado de definirle un valor tanto a la función de transición, como a los estados de aceptación, todo esto dependiendo de el tipo de token que se está evaluando.

3.4.5 evaluarCHAR():Recibe(char actual, String [] datos), Es el encargado de confirmar si el carácter de entrada pertenece dentro del arreglo de datos.

3.4.6 verificarEstadosAceptacion():Recibe(), Es el encargado de verificar que el ultimo estado registrado, corresponda a un estado de aceptación, de lo contrario, el token sería un error.

3.4.7 reiniciarLector():Recibe(TipoToken tipo), Es el encargado de reiniciar los valores del lector, para volver a evaluar un nuevo texto.

3.4.8 añadirMovimientoRegistro():Recibe(String estado, char valor), Es el encargado de agregar el registro de un movimiento de estados a una lista.

3.4.9 añadirPalabraRegistro():Recibe(String token), Es el encargado de crear y agregar el nombre de un token a un registro de un movimiento de estados a una lista.

3.4.10 guardarMovimientosTotales():Recibe(), Es el encargado guardar todos los registros de un token evaluado, dentro de una lista con todos los movimientos efectuados de todos los tokens.