Manual Técnico

La clase analizador léxico tiene como propósito recibir el listado de conjuntos y la tabla de transiciones para verificar, también recibe la entrada en cuestión y el nombre del autómata al cual se está haciendo referencia.

```
public AnalizadorLexico(String entrada, Tabla automata, LinkedList conjuntos, String nombreAutomata){
    this.entrada = entrada;
    this.automata = automata;
    this.conjuntos = conjuntos;
    this.nombreAutomata = nombreAutomata;
}
```

Este también cuenta con 3 métodos, uno para comprobar que el carácter actual corresponde a uno de los conjuntos, el cual vendría siendo este

```
public boolean buscarCoincidenciaConjunto(char caracter, String conjunto){
    Conjunto conjSeleccionado = null;
    for(int i = 0; i < conjuntos.size(); i++){
        conjSeleccionado = (Conjunto) conjuntos.get( index: i);
        if(conjSeleccionado.getNombre().equals( anObject: conjunto)) break;
    }

    LinkedList caracteres = conjSeleccionado.getLista();
    char caracterAux = ' ';
    for(int i = 0; i < caracteres.size(); i++){
        caracterAux = (char) caracteres.get( index: i);
        if(caracterAux == caracter) return true;
    }
    return false;
}
```

Luego la otra función tiene como propósito ir pasando carácter por carácter y comprobando si este coincide con la tabla de transiciones

```java
public Resultado pasarCaracter() throws InterruptedException{
    LinkedList transiciones;
    LinkedList tokens;
    Resultado result;



    while(comprobarFinalCadena() == false){
        transiciones = automata.getFila( fila: estado).getColumna();
        tokens = automata.getFila( fila: estado).getTokens();
        for(int i = 0; i < transiciones.size(); i++){
            String tokenActual = (String) tokens.get( index: i);
            char puntero = entrada.charAt( index: posicion);
            if(tokenActual.charAt( index: 0)!='"'){
                if(buscarCoincidenciaConjunto( caracter: puntero, conjunto: tokenActual)){
                    estado = (int) transiciones.get( index: i);
                    posicion++;
                    break;
                }
            }else{
                if(tokenActual.charAt( index: 1) == puntero){
                    estado = (int) transiciones.get( index: i);
                    posicion++;
                    break;
                }
            }
            if(i == transiciones.size() - 1){
                return new Resultado( automata: nombreAutomata,  cadena: entrada,  aprobado: "no aprobado");
            }
        }

    }
```

Luego por cada iteración debe confirmar si ya se terminó la cadena, es decir, si se está hablando del carácter # o se acabaron los caracteres de la cadena

```java
public boolean comprobarFinalCadena(){
    if(entrada.length() <= posicion){
        for(int i = 0; i < automata.getFila( fila: estado).getTokens().size(); i++){
            if(automata.getFila( fila: estado).getTokens().get( index: i).equals( obj: "#")){
                resultado = new Resultado( automata: nombreAutomata,  cadena: entrada,  aprobado: "aprobado");
                return true;
            }
        }
        resultado = new Resultado( automata: nombreAutomata,  cadena: entrada,  aprobado: "no aprobado");
        return true;
    }
    return false;
}
```

Por otro lado la clase Arbol, siendo esta la más compleja de todas la cantidad de métodos incluida en ella, ya que apartir de esta se crea la tabla de transiciones y la tabla de siguientes, anulables, primeros y últimos, por lo tanto esta cuenta con muchos métodos

```
public Arbol(String valor, int id){
    this.valor = valor;
    this.id = id;
    this.anulabilidad = false;
    this.ramaIzq = null;
    this.ramaDer = null;
    this.primeros = new LinkedList();
    this.ultimos = new LinkedList();
}
```

Se solicita un id y un valor para crear un nodo, el id es para hacer único cada uno de los nodos y poder realizar el gráfico en graphviz

Luego la clase Errores tiene puro propósito de estructura de datos, solamente son 3 variables y se utiliza para almacenarlas mediante una lista

```
public Errores(int col, int fil, String token){
    this.col = col;
    this.fil = fil;
    this.token = token;
}
```

Para simular una tabla en Java se hizo uso de dos clases, Fila y Tabla, la Tabla siendo un conjunto de Filas

```
public class Fila {
    private int encabezado;
    private LinkedList columnas;
    private LinkedList tokens = new LinkedList();
    private String token;

public Tabla(){
    filas = new LinkedList();
}
```

Para la tabla de transiciones se utilizaron varias cosas

```
public class Transiciones {
    private Tabla tablaEstados = new Tabla();
    private Tabla tablaSiguientes = new Tabla();
    private Tabla tablaTransiciones = new Tabla();
    private Tabla tablaTransicionesGrafica = new Tabla();
```