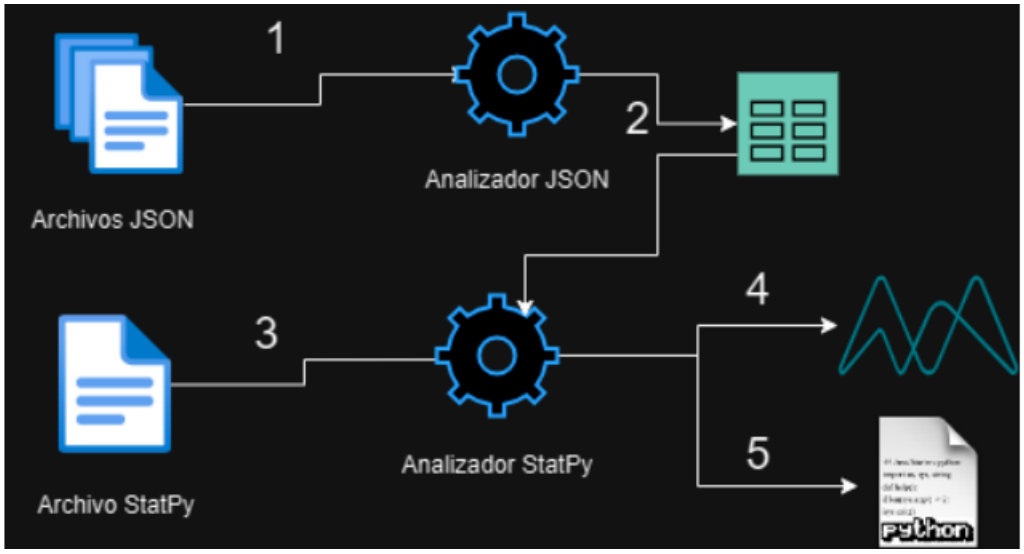
Manual técnico

El siguiente proyecto se desarrolló en el lenguaje de Java en NeatBeans.

Funcionamiento



Para el funcionamiento de este programa tenemos dos analizadores, los cuales pueden abrir dos archivos, ya se Json y sp, los cuales pasaran por un analizador para poder traducir o graficar dependiendo la situación.

Análisis Léxico – JFLEX

Se utilizo esta herramienta para generar un autómata capaz de reconocer la gramática tipo 3 que define los lexemas de entrada del lenguaje.

Análisis Sintáctico – CUP

Se utilizo esta herramienta capaz de generar un programa, capaz de evaluar un flujo de tokens y validar el orden correcto del mismo, este programa utiliza una gramática LALR, la cual es una variante de las gramáticas LR(1), con la diferencia de que la precedencia de las producciones debe ser colocada explícitamente.

Jcup (JavaCup) es una herramienta de generación de analizadores léxicos y sintácticos para el lenguaje de programación Java. Es una herramienta popular utilizada en la construcción de compiladores y analizadores sintácticos para procesar código fuente Java.

Jfreechart

Esta herramienta nos servirá para graficar ya sea las entradas correspondientes, pudiendo ser estas Json o el Statpy

Instrucciones

• Asignación: define la operación de asignación posterior a la definición.

• Case: case de la sentencias switch-case.

• Declaración: define la operación de definición de variables.

• Elif: Operación de combinar

else-if para múltiples casos.

• Ejecutar: llamadas a funciones.

• Para: Ciclo for, similar al utilizado en lenguajes como c++, con expresiones aritméticas e incrementos.

• Función: funciones con instrucciones de retorno.

• Si: sentencia if, contiene todos los casos, if simple o con sentencia else, también contempla sentencias elif.

• Operación: sentencia base, contiene los terminales, expresiones aritméticas y booleanas, contempla la ejecución de funciones y agrupación en paréntesis.

• Parámetros: parámetros de firmas de funciones y procedimientos • Imprimir: funciones que permiten definir la función de imprimir una operación.

• Procedimiento: funciones sin retorno. • Repetir: sentencia repetir hasta que una condición se cumpla.

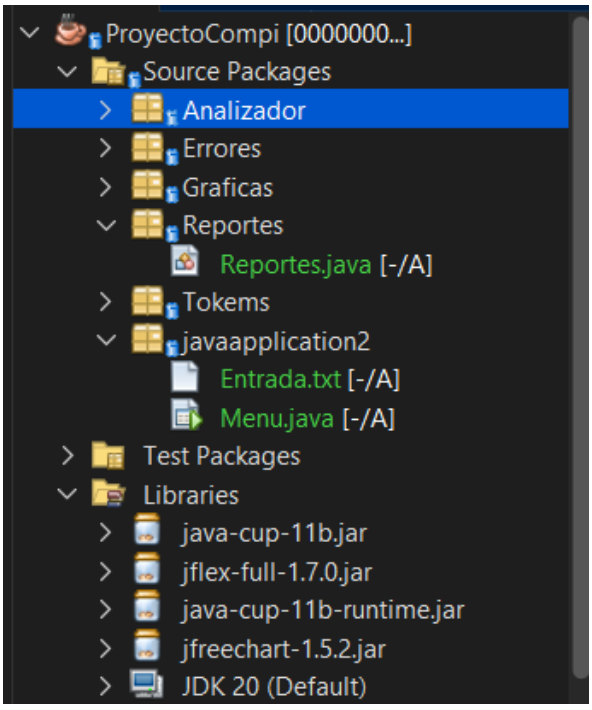
• Retorno: retorna una operación.

• Switch: sentencia contenedora de múltiples casos. • Mientras: ciclo mientras-hacer

Paquetes

Para el siguiente proyecto se hizo uso de diversos paquetes, los cuales tenemos

- Analizador: En este paquete se encuentran los analizadores léxicos y sintácticos, los cuales nos ayudaran a definir nuestros tokens y nuestras expresiones regulares, respectivamente. Esto con el objetivo de poder realizar las traducciones y las lectura del lenguaje Statpy y Json.
- Errores: En esta clase nosotros podemos obtener el tipo de error, la descripción, la fila y la columna en que estos se encuentran,
- Grafica: En esta clase podremos realizar las graficas con los valores obtenidos en nuestros analizadores, los cuales serán almacenados en arraylist, esto con el objetivo de poder y almacenándolos en orden, para posteriormente ir imprimiéndolos y mostrarlos en nuestra interfaz.
- Tokens. En esta clase nosotros podemos obtener los tokens, la descripción, la fila y la columna en que estos se encuentran.



Podemos ver que hay varios paquetes los cuales nos ayudaran para poder identificar de mejor manera cada archivo, como demás de poder trabajar de manera más ordenada.

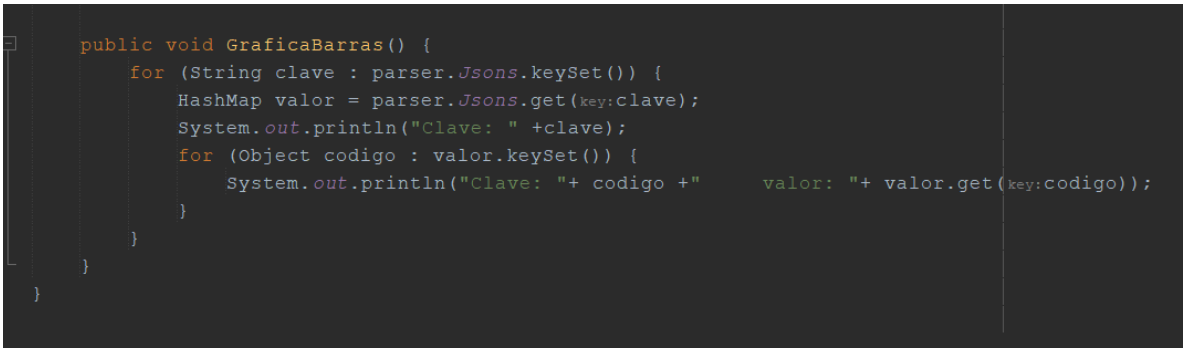
```java
public class Reportes {

    public static void ReportesLexema() {
        // Especifica la ruta y el nombre del archivo que deseas abrir
        String rutaArchivoHTML = "TablaLexicos.html"; // Cambia esta ruta según tu archivo HTML

        try {
            // Verifica si el sistema admite la clase Desktop
            if (Desktop.isDesktopSupported()) {
                Desktop desktop = Desktop.getDesktop();

                // Comprueba si el archivo HTML existe
                File archivoHTML = new File(pathname: rutaArchivoHTML);
                if (archivoHTML.exists()) {
                    // Abre el archivo HTML en el navegador web predeterminado
                    desktop.open(file: archivoHTML);
                } else {
                    JOptionPane.showMessageDialog(parentComponent: null, message: "El archivo no se ha creado", title: "Alerta", messageType: JOpt
                }
            } else {
                System.out.println(x: "El sistema no admite Desktop.");
            }
        } catch (IOException e) {
            // Manejo de excepciones en caso de error al abrir el archivo
            e.printStackTrace();
        }
```

Para la clase de reporets contenida en el paquete de reportes, esta su única función es abrir los reportes ya realizados, de manera contraria si estas aun no estas creadas, esta mandara una alerta para indicar que esta aun no ha se han creado.



```java
    public void GraficaBarras() {
        for (String clave : parser.Jsons.keySet()) {
            HashMap valor = parser.Jsons.get(key:clave);
            System.out.println("Clave: " +clave);
            for (Object codigo : valor.keySet()) {
                System.out.println("Clave: "+ codigo +"     valor: "+ valor.get(key:codigo));
            }
        }
    }
}
```

En graficas hay dos contructrores los cuales comienzan con “mostrar” esta su inica función es que estas muestrren las dos grafica, que ya han sido credas, ensto para que poder observarlas al darle al botón de garficar.

De igual manera hay una función la cual esta recibe parámetros para que estas sean mandados a grafiar, obtener los datos que se quieren y estas se van ya empezaran a crear sus graficas.

```java
public void Reportes() {
    String nombreArchivo = "TablaTokems.html";
    String ejecutar = Entrada.getText();

    try {
        //  sintactico con el parser
        parser parser = new parser();
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName: nombreArchivo));

        // Escribir el encabezado HTML
        writer.write(str:"<!DOCTYPE html>");
        writer.newLine();
        writer.write(str:"<html>");
        writer.newLine();
        writer.write(str:"<head>");
        writer.newLine();
        writer.write(str:"<title>Tabla HTML generada desde Java</title>");
        writer.newLine();
        writer.write(str:"</head>");
        writer.newLine();
        writer.write(str:"<body>");
        writer.newLine();

        // Crear la tabla HTML
        writer.write(str:"<table style=\"collapse; margin: 25px 0; font-size: 1em; font-family: sans-serif; min-width: 450px;
        writer.newLine();

        // Encabezado de la tabla
        writer.write(str:"<tr style=\"background-color: #980081; color: #ffffff; text-align: middle;\" >");
```

```java
    writer.newLine();
    writer.write(str:"<th>Lexema</th>");
    writer.newLine();
    writer.write(str:"<th>Token</th>");
    writer.newLine();
    writer.write(str:"<th>Linea</th>");
    writer.newLine();
    writer.write(str:"<th>columna</th>");
    writer.newLine();
    writer.write(str:"</tr>");
    writer.newLine();

    //System.out.println(scanner.lexemas.size());
    for (int i = 0; i < scanner.lexemas.size(); i++) {
        writer.write(str:"<tr style=\"background-color: rgb(223, 223, 223);\" >");
        writer.newLine();
        writer.write("<td>" + scanner.lexemas.get(index: i).getTipo() + "</td>");
        writer.newLine();
        writer.write("<td>" + scanner.lexemas.get(index: i).getDescripcion() + "</td>");
        writer.newLine();
        writer.write("<td>" + scanner.lexemas.get(index: i).getLinea() + "</td>");
        writer.newLine();
        writer.write("<td>" + scanner.lexemas.get(index: i).getColumna() + "</td>");
        writer.newLine();
        writer.write(str:"</tr>");
        writer.newLine();
    }
```

para generar las tablas se utrilizo una Liberia la cual se llama BufferedWriter, la cual facilita la creación de la tabla html para posteriormente abrirla. Esta en el contenido de la tabla se genera a partir de un arraylist el cual estas se iran guardando las traducciones de en el cup, realizando traducciones o también para los errores léxicos