

Universidad De San Carlos de Guatemala  
Facultad de Ingeniería  
Escuela de Ciencias y Sistemas  
Organización de lenguajes y compiladores 1

## **MANUAL TÉCNICO**

Gerson Aaron Quinia Folgar  
Carnet: 201904157

## Herramientas utilizadas para el desarrollo

### Java

El lenguaje de programación Java es un lenguaje de desarrollo orientada a objetos, fue diseñado para que no se dependa en muchas implementaciones, el lenguaje permite a los desarrolladores ejecutar sus aplicaciones en cualquier dispositivo si necesidad de recompilar el código y por lo tanto es considerado multiplataforma.

### Graphviz

Graphviz es un conjunto de herramientas de software para el diseño de diagramas definido en el lenguaje descriptivo DOT y se utilizó para poder representar los árboles, autómatas y tablas generadas por el programa.

## Librerías utilizadas

### JFlex

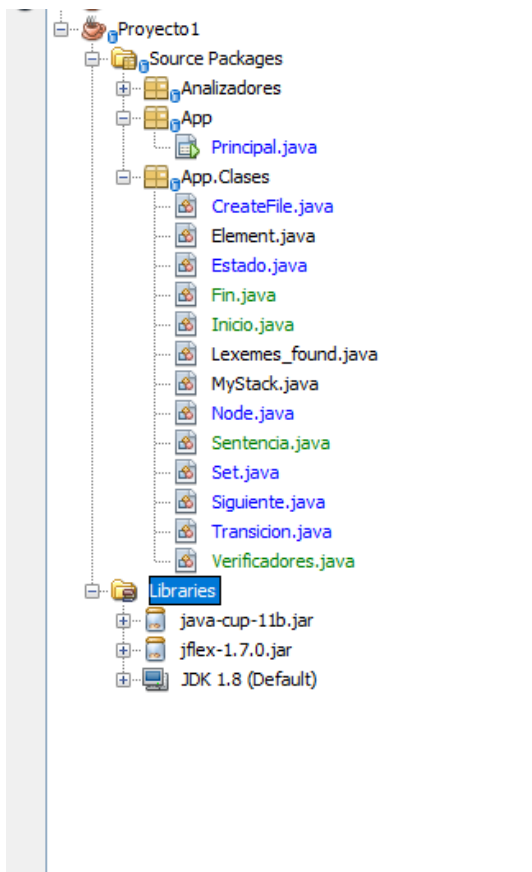
JFlex es un generador de un analizador léxico parecido a LEX, el cual toma como entrada una cadena de caracteres, y lo convierte en una secuencia de tokens.

### Cup

Cup es un generador de analizadores sintácticos LALR en Java el cual recibe de entrada un archivo con la estructura de la gramática y su salida es un parser escrito en Java listo para usarse.

## Estructura general

El programa regexive utiliza el paradigma orientado a objetos, se divide en varias clases que controlan cada parte de la aplicación, la clase Principal contiene el método main y es donde se inicia el programa. Para un mejor orden se almacenaron las clases en tres distintos paquetes según su uso.



### Analizadores

Este paquete contiene todas las clases generadas por las librerías de Cup y Jflex, así como los archivos donde se definieron las gramáticas y las instrucciones para la creación del analizador léxico.

### App

Contiene la clase principal donde se inicia el programa, en la clase principal se encuentra toda la parte del desarrollo fronted del programa.

### App.Clases

Contiene todas las clases utilizadas para la generación de archivos, imágenes, estados,

tablas de transiciones, tablas de siguientes, recorrido de árboles, creación de árboles y validación de cadenas con las expresiones regulares que se ingresan.

## Clase Node.java

La clase nodo es una de las más importantes ya que a través de ella se genera el árbol utilizando la recursividad, también tiene un método para recorrer el árbol desde

abajo hacia arriba pasando como parámetro la raíz del árbol que es el nodo inicial, por medio de esta función se aplican las leyes del método del árbol para crear la tabla de siguientes y al mismo tiempo que se recorre se va formando el AFND, para posteriormente ser graficado. Otro método importante es el que se utiliza para obtener los hijos de cada nodo y posteriormente graficarlos, este método es de tipo String, el resultado de este se envía a un método de la clase CreateFile para poder graficar el método del árbol.

## Clase Verificadores.java

Esta clase contienen métodos para la validación de cadenas y el ingreso de los terminales aceptados por cada estado, algunos de sus métodos mas importantes son:

### ***agregarConjuntos()***

Este método recorre las transiciones y va agregando los caracteres aceptados a una lista dentro de cada transición.

### ***Validar ()***

Recorre todos los estados de la expresión regular encontrada al mismo tiempo que analiza cada carácter de la entrada, si al final se llega al carácter de aceptación ingresa la entrada que se está analizando al archivo JSON.

## Clase Estado.java

La clase estado guarda todos los estados que han sido encontrados con sus transiciones correspondientes, posee un método para poder verificar que estados ya han sido recorridos e ir almacenando las transiciones según el terminal correspondiente. La clase estado igualmente tiene métodos para poder graficar las tablas de siguientes y tablas de transiciones.

### ***CalcularEstados()***

Este método se encarga de encontrar todos los estados mediante la tabla de siguientes, para llevar control de que estados ya han sido recorridos, los estados de cada expresión se almacenan en una lista y se verifica que aún no estén almacenados para poder recorrerse.

### ***graficarAFD ()***

El método recibe como parámetro todos los estados de cada expresión regular y devuelve un String que contiene la gráfica del AFD en el lenguaje DOT, el resultado de este método es enviado al método de creación de imagen para generar el AFD.

## **Clase CreateFile.java**

Esta clase contiene todos los métodos para la creación de archivos y generación de imágenes con graphviz, la mayoría de los métodos reciben el String con el lenguaje dot y crean la imagen en png.

### ***CreateTree()***

Este método grafica el método del árbol y lo guarda en la carpeta especificada dentro del proyecto.

### ***CreateDigraph()***

Este método recibe como parámetro un String con el nombre del archivo y otro String con las instrucciones en lenguaje dot, se encarga de crear el archivo y almacenarlo en la carpeta especificada del proyecto con el nombre que se pasa en el parámetro.

### ***CreateJson()***

El método recibe todas las cadenas validas que se han encontrado y se encarga de crear un archivo en formato JSON con las cadenas válidas que se le pasan por parámetro.

## Requisitos del sistema

### Requerimientos de hardware

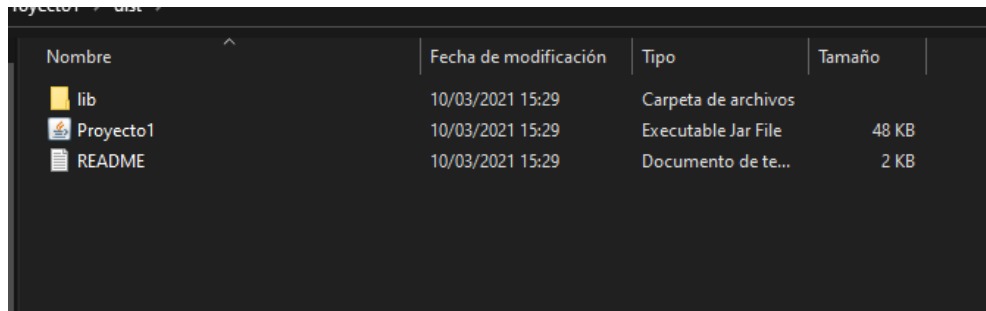
- Equipo, teclado, mouse, monitor.
- Memoria RAM 2GB.

### Requerimientos de software

- Sistema operativo Windows.
- Java.
- Graphviz 2.38.

## Ejecución del programa

Para la ejecución del programa debemos ir a la carpeta Proyecto1, en esta carpeta encontraran un archivo ejecutable llamado Proyecto1, al dar doble click sobre el archivo ejecutable el programa iniciará la ejecución.



A screenshot of a Windows File Explorer window showing the contents of a folder named 'Proyecto1'. The window has a dark theme. The address bar shows 'Proyecto1' and 'C:\'. The main area displays a table of files and folders.

Nombre	Fecha de modificación	Tipo	Tamaño
lib	10/03/2021 15:29	Carpeta de archivos	
Proyecto1	10/03/2021 15:29	Executable Jar File	48 KB
README	10/03/2021 15:29	Documento de te...	2 KB