UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
CENTRO UNIVERSITARIO DE OCCIDENTE
DIVISIÓN DE CIENCIAS DE LA INGENIERÍA
LENGUAJES FORMALES DE PROGRAMACIÓN

MANUAL TÉCNICO

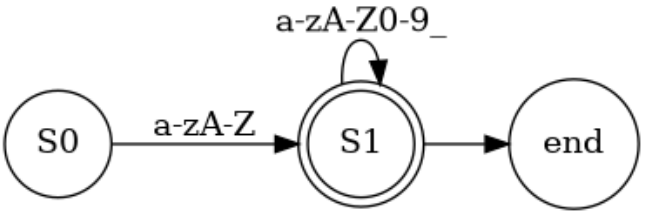
DIEGO ALESSANDRO GONZALEZ BRAVO / 202230939

HERRAMIENTAS DE DESARROLLO:
Las herramientas (IDES, lenguajes de programación) utilizadas para el desarrollo de la aplicación Manejo de tarjetas son:

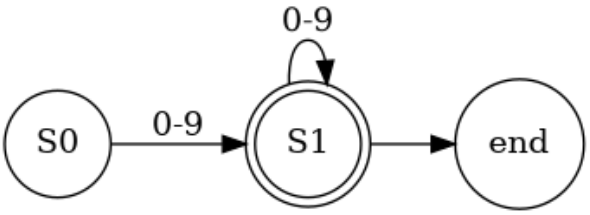
- JAVA
  Version openjdk version "21.0.4"
- Apache NetBeans
  Versión 22
- Sistema operativo de desarrollo
  Ubuntu 24.04 LTS
- Graphviz
  versión 2.43.0

CONCEPTOS APLICADOS:
AUTÓMATAS DE SUBPROGRAMAS:
Definición de Autómatas

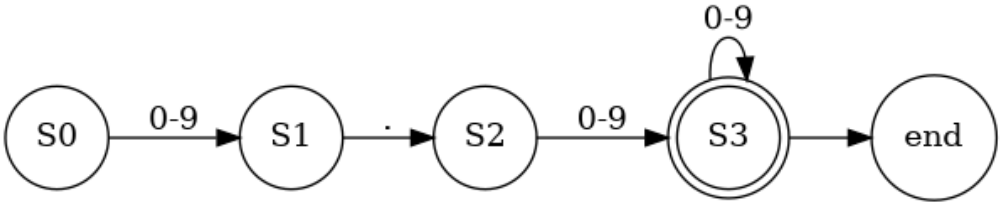
1. Autómata para Identificadores:
   - Comienza con una letra.
   - Puede contener letras, dígitos, o guiones bajos.



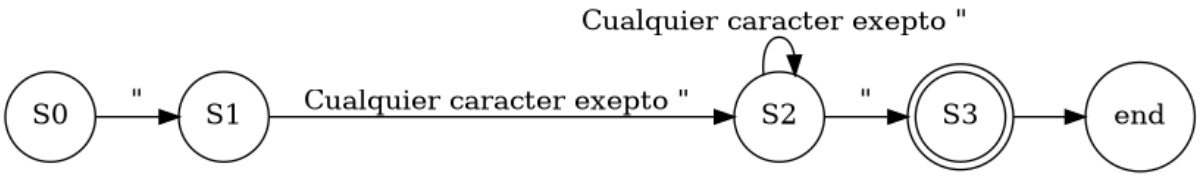
2. Autómata para Números Enteros:
   - Comienza con un dígito.
   - Puede tener más dígitos.



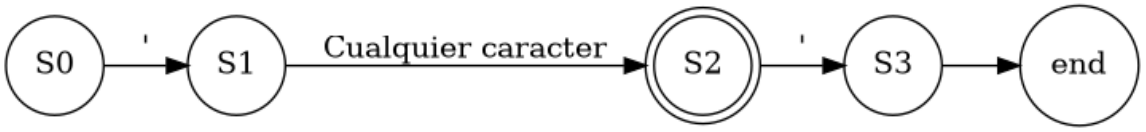
3. Autómata para Números Decimales:
   - Comienza con un dígito.
   - Debe contener un punto seguido de más dígitos.



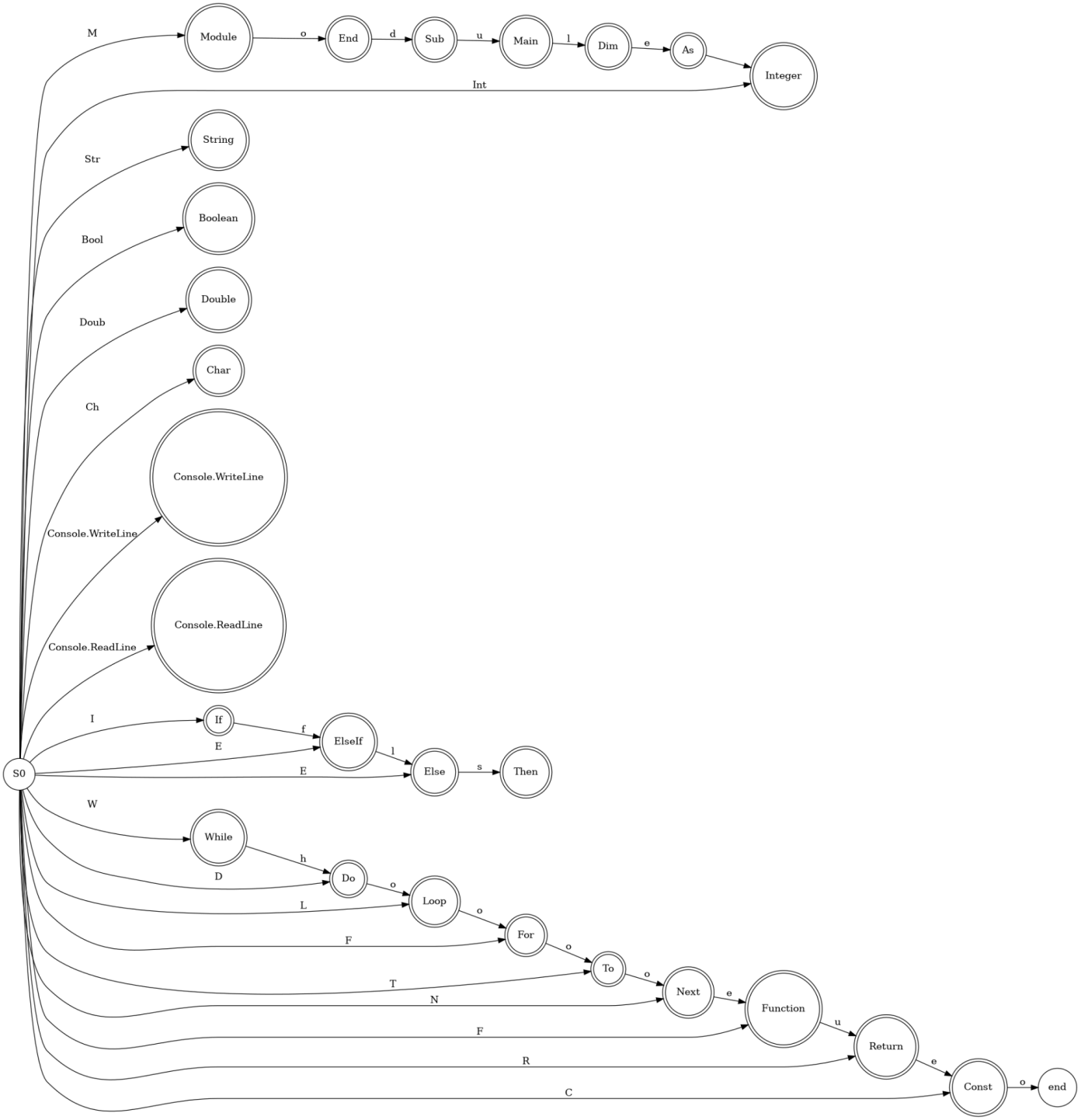
4. Autómata para Cadenas de Texto:
    - Comienza y termina con comillas dobles (").
    - Puede contener cualquier carácter excepto las comillas dobles no escapadas.



5. Autómata para Caracteres:
    - Comienza y termina con comillas simples (').
    - Puede contener un único carácter dentro de las comillas.



6. Autómata para Palabras Reservadas:
    - Compara el token completo con una lista predefinida de palabras reservadas.



CREAR TOKENS Y PONER COLORES:

1. Lectura del Texto
    - Obtención del Texto:
        - Se lee el contenido del JTextArea en forma de una cadena de texto completa. Esto se realiza para procesar todo el texto ingresado por el usuario.

2. Inicialización del Procesamiento
    - Inicialización de Variables:
        - Se inicializan variables para manejar el procesamiento del texto, como fila y columna para rastrear la ubicación del texto, y tokenActual para construir los tokens mientras se recorren los caracteres.

3. Recorrido del Texto
    - Iteración a través de Caracteres:
        - Se convierte el texto en un arreglo de caracteres y se recorre cada carácter uno por uno.
        -
    - Manejo de Espacios y Saltos de Línea:
        - Cuando se encuentra un espacio o un salto de línea, se verifica el tokenActual. Si hay texto en tokenActual, se procesa ese token y se agrega a la lista de tokens.
        - Se actualizan las posiciones fila y columna para reflejar los cambios debidos a espacios y saltos de línea.

4. Construcción de Tokens
    - Construcción del Token Actual:
        - Si el carácter actual es parte de un identificador, número o cualquier otro tipo de token, se añade al tokenActual.
    - Verificación de Identificadores:
        - Si se detecta una letra al inicio de un token, se construye un identificador.
        - El identificador se verifica para asegurarse de que cumple con las reglas definidas (comienza con letra y puede contener letras, dígitos y guiones bajos).
    - Verificación de Números y Otros Tokens:
        - Se verifican números enteros y decimales en función de sus formatos específicos.
        - Se identifican cadenas de texto y caracteres según los delimitadores (" para cadenas, ' para caracteres).

5. Determinación del Tipo de Token
    - Uso de Métodos de Verificación:
        - Se usa un conjunto de métodos para identificar el tipo de token (operador, comparación, palabra reservada, etc.).
        - Se comparan los tokens con un conjunto predefinido de palabras reservadas y otros tipos de símbolos.

6. Asignación de Color y Tipo
    - Asignación de Color:
        - Una vez que se determina el tipo de token, se asigna un color correspondiente utilizando un mapa de colores para tokens.
    - Adición a la Lista de Tokens:
        - El token, junto con su tipo y color, se agrega a la lista de tokens.

7. Finalización del Procesamiento
    - Procesamiento de Token Final:
        - Al final del texto, si hay un token restante en tokenActual, se procesa y se agrega a la lista de tokens.

USO DE GRAPHVIZ:

1. Método drawGraph()
   Este método es responsable de generar un archivo de gráfico a partir de un archivo .dot y convertirlo en una imagen PNG utilizando Graphviz. Aquí está el paso a paso:

a. Seleccionar Ubicación de Guardado:

- Se crea un objeto JFileChooser para abrir un diálogo de selección de archivo.
- Se configura el JFileChooser con un título y un nombre de archivo predeterminado (token_graph.png).
- Se muestra el diálogo al usuario mediante showSaveDialog(null).
- Si el usuario selecciona una ubicación y confirma (JFileChooser.APPROVE_OPTION), se obtiene el archivo seleccionado.

b. Definir Rutas para Archivos:
    - Se obtiene la ruta del escritorio del usuario mediante System.getProperty("user.home").
    - Se define la ruta para el archivo .dot en el escritorio (rutaDot).
    - Se obtiene la ruta completa del archivo PNG seleccionado por el usuario (rutaPng).

c. Generar Archivo .dot:
    - Se llama al método dotFileMaker(getTknGraph(), rutaDot) para crear el archivo .dot con la representación del token. Este método guarda el contenido del gráfico en el archivo .dot en la ubicación del escritorio.

d. Ejecutar Graphviz:

- Se configura un ProcessBuilder para ejecutar el comando de Graphviz (dot) para convertir el archivo .dot en una imagen PNG.
- El comando se construye con los siguientes parámetros:
    - dot: El ejecutable de Graphviz.
    - -Tpng: Especifica el formato de salida como PNG.
    - -o: Indica el archivo de salida.
    - rutaPng: La ruta donde se guardará el archivo PNG.
    - rutaDot: El archivo de entrada (archivo .dot).

e. Capturar y Verificar Errores:

- Se lee la salida de error del proceso para capturar cualquier mensaje de error durante la ejecución.
- Se espera a que el proceso termine y se verifica el código de salida (exitCode).
- Si el código de salida es 0, se considera que el gráfico se generó exitosamente.
- Si el código de salida es diferente de 0, se muestra un mensaje de error con los detalles de la salida.