MANUAL TÉCNICO

1. Información del sistema:

- • OS: Ubuntu 22.04.2 LTS
- • OS Type: 64-bit
- • CPU: Intel core i5 2.50 GHz
- • GPU: Mesa Intel HD Graphics 4400
- • Versión de java: 17
- • RAM: 8 GB
- • IDE: IntelliJ idea
- • Control de versiones: git 2.34.1
- • Github: https://github.com/Hatsune02/GeoDraw.git
-

2. Descripción del proyecto:

Consiste en la implementación de JFlex y CUP para la creación de analizadores léxicos y sintácticos. Esto con el fin de poder realizar un programa que nos ayude a graficar toda clase de polígonos y círculos de diferentes dimensiones, posiciones y colores, además de poder agregarles una animación ya sea lineal o curva para cada figura. Todo esto podrá ser exportado

3. Instalación

3.1. Instalación de Java

Descargue e instale el JDK si no lo tiene instalado.

Verifique la instalación ejecutando el comando java -version en la terminal o símbolo del sistema.

3.2. Instalación de la Aplicación

Navegue hasta la ubicación del archivo .jar de la aplicación.

Ejecute la aplicación usando el comando:

```
java -jar NombreDeLaAplicacion.jar
```

La aplicación se iniciará y la interfaz gráfica será visible.

4. Detalles Técnicos

Arquitectura del sistema

Esta sección describe la arquitectura general de la aplicación, incluyendo los componentes principales y cómo interactúan entre sí.

- **Frontend**: Interfaz gráfica construida con Java Swing para la interacción del usuario.
- **Backend**: Lógica de la aplicación, que incluye el procesamiento de instrucciones y la generación de gráficos.
- **Parser y Lexer**: Manejo de la gramática del lenguaje formal, análisis léxico y sintáctico utilizando JFlex y CUP.

Descripción General del Uso de JFlex y CUP en el Programa

Para desarrollar la funcionalidad de la aplicación, utilicé JFlex y CUP, herramientas que permiten analizar y procesar el lenguaje formal definido para graficar figuras geométricas y realizar animaciones.

JFlex

Definición: JFlex es un generador de analizadores léxicos para Java. Un analizador léxico, o lexer, es un componente que lee el código de entrada como una secuencia de caracteres y lo divide en unidades significativas llamadas tokens. Estos tokens son luego procesados por el parser para entender la estructura del lenguaje.

Uso en el Programa: En la aplicación, JFlex se utilizó para construir un lexer que identifica las palabras clave, operadores aritméticos, colores, y otros elementos sintácticos dentro de las instrucciones ingresadas por el usuario. Este lexer es responsable de escanear el código fuente introducido y generar tokens que representan las distintas partes del lenguaje, como comandos de graficación (`graficar`), operadores como `+`, `-`, y nombres de colores (`rojo`, `azul`).

CUP

Definición: CUP (Constructor of Useful Parsers) es una herramienta para generar parsers basados en Java, similar a herramientas como Yacc, pero diseñada específicamente para el lenguaje Java. Un parser toma los tokens generados por el lexer y los organiza según una serie de reglas gramaticales para comprender la estructura del código y su semántica.

Uso en el Programa: En la aplicación, CUP se empleó para crear un parser que analiza los tokens generados por JFlex, validando la sintaxis de las instrucciones y construyendo una estructura lógica que se utiliza para generar gráficos y animaciones. El parser maneja la interpretación de expresiones aritméticas, verifica la correcta estructuración de los comandos de graficación y animación, y asegura que el código siga las reglas definidas por la gramática.

LEXER

/* ___Operators___ */

Plus = [+]
Minus = [-]
Times = [*]
Division = [\/]
LParen = [\(]
RParen = [\)]

/* ___Reserved words___ */

/* actions */
graficar = (graficar)
animar = (animar)
objeto = (objeto)
anterior = (anterior)

/* colors */
blue = (azul)
red = (rojo)
yellow = (amarillo)
green = (verde)
sky = (celeste)
cyan = (cyan)
black = (negro)
pink = (rosado)
purple = (morado)

/* animation */
line = (linea)
curve = (curva)

/* objects */
circle = (circulo)
square = (cuadrado)
rectangle = (rectangulo)
polygon = (poligono)

/* Others */
Identifier = [_a-zA-Z0-9]+
Digit = [0-9]+
Decimal = Digit\.Digit
Comma = [,]

PARSER

Producciones terminales = {COMMA, GRAFICAR, ANIMAR, OBJETO, ANTERIOR, LINE, CURVE, CIRCLE, SQUARE, RECTANGLE, POLYGON, PLUS, MINUS, TIMES, DIVISION, LPAREN, RPAREN, BLUE, RED, YELLOW, GREEN, SKY, CYAN, BLACK, PINK, PURPLE, ID, DIGIT**}**

Producciones no terminales = {s, instructions, instruction, graph, animate, circle_square_param, animate_param, rectangle_line_param, polygon_param, color, expr}

Producciones / Gramatica

```
s ::= instructions
      ;

instructions ::= instruction
        | instructions instruction

instruction ::= graph
        | graph animate

graph ::=     GRAFICAR CIRCLE LPAREN circle_square_param RPAREN
        | GRAFICAR SQUARE LPAREN circle_square_param RPAREN
        | GRAFICAR RECTANGLE LPAREN rectangle_line_param RPAREN
        | GRAFICAR LINE LPAREN rectangle_line_param RPAREN
        | GRAFICAR POLYGON LPAREN polygon_param RPAREN
        ;

animate ::= ANIMAR OBJETO ANTERIOR LPAREN animate_param RPAREN

circle_square_param ::= ID COMMA expr COMMA expr COMMA expr COMMA color

rectangle_line_param ::= ID COMMA expr COMMA expr COMMA expr COMMA expr
COMMA color

polygon_param ::=   ID COMMA expr COMMA expr COMMA expr COMMA expr
COMMA expr COMMA color

animate_param ::=     LINE COMMA expr COMMA expr COMMA expr
                | CURVE COMMA expr COMMA expr COMMA expr
```

```
color ::=    BLUE
          |RED
          |YELLOW
          |GREEN
          |SKY
          |CYAN
          |BLACK
          |PINK
          |PURPLE

expr ::=  expr PLUS expr
       | expr MINUS expr
       | expr TIMES expr
       | expr DIVISION expr
       | LPAREN expr RPAREN
          | MINUS expr
       | DIGIT
```

Flujo General

1. **Entrada del Usuario**: El usuario introduce las instrucciones en un cuadro de texto.
2. **Análisis Léxico (JFlex)**: JFlex procesa el texto de entrada, identificando tokens como `graficar`, operadores aritméticos, y nombres de colores.
3. **Análisis Sintáctico (CUP)**: CUP toma estos tokens y los organiza según las reglas gramaticales, determinando si la estructura es válida y construyendo un árbol sintáctico que representa las acciones a realizar.
4. **Ejecución**: Con base en la estructura generada, la aplicación grafica las figuras geométricas y prepara las animaciones según lo indicado por las instrucciones del usuario.

Este flujo permite a la aplicación interpretar y ejecutar de manera eficiente un lenguaje formal específico, facilitando la creación y animación de gráficos de forma interactiva.