Manual Técnico

Este manual técnico proporciona una guía detallada para implementar un proyecto de análisis y exportación de figuras geométricas utilizando JFlex y CUP para el análisis de tokens, Java Swing para la interfaz gráfica, y Apache Ant para la construcción del proyecto. El proyecto incluye la capacidad de analizar expresiones, representar figuras geométricas y exportar gráficos a formatos PNG o PDF.

Recursos utilizados:

- **JDK**: Java Development Kit (JDK) 21 o superior.
- **JFlex**: Herramienta para el análisis léxico.
- **CUP**: Constructor de analizadores de sintaxis.
- **Apache Ant**: Herramienta de construcción de proyectos.
- **Biblioteca iText**: Para la generación de archivos PDF.
- **IDE**: NetBeans

Análisis Léxico y Sintáctico

1. **JFlex**: Define las reglas léxicas en el archivo .flex para generar el analizador léxico.
2. **CUP**: Define la gramática y el manejo de errores en el archivo .cup para generar el analizador sintáctico.

Gramática utilizada:

// Definición de los símbolos terminales

terminal ERROR, CURVA, LINEA, NOMBRE, COMA, NUMERO, PARENTECIS_ABRE, PARENTECIS_CIERRA;

terminal CIRCULO, CUADRADO, POLIGONO, RECTANGULO, GRAFICAR, OBJETO, ANTERIOR, ANIMAR, PLUS, MINUS, DIVIDE, TIMES;

terminal ROSA, NARANJA, GRIS, MORADO, NEGRO, VERDE, AMARILLO, ROJO, AZUL;

```
// Definición de los símbolos no terminales
non terminal color, expresion, sentencia, sentencias, animar, graficar, tipoA, figuraG, entrada;

// Definición de la precedencia
precedence left PLUS, MINUS;
precedence left TIMES, DIVIDE;

// Símbolo de inicio
start with sentencias;

// Reglas de producción
sentencias ::= sentencia
            | sentencia sentencias;

sentencia ::= graficar
            | animar;

tipoA ::= CURVA
        | LINEA;

animar ::= ANIMAR OBJETO ANTERIOR PARENTECIS_ABRE tipoA:t COMA expresion:x COMA
expresion:y COMA expresion:o PARENTECIS_CIERRA
        {:
            // Se crea una instancia de Animacion y se guarda en la pila
            //Animacion anim = new Animacion(t, (double)x, (double)y, (double)o);
            //figuras.push(anim);
        :}
```

```
    ;


graficar ::= GRAFICAR entrada:e
        {:

          // Se crea la figura correspondiente y se guarda en la pila
          figuras.add((Animable)e);
        :}
        ;


entrada ::= figuraG:f PARENTECIS_ABRE NOMBRE:nombre COMA expresion:ex1 COMA
expresion:ex2 COMA expresion:ex3 COMA color:c PARENTECIS_CIERRA
        {:
          switch (f.toString()) {
             case "circulo":
                Circulo circulo = new Circulo(nombre.toString(), (double)ex1, (double)ex2,
(double)ex3, c.toString());
                RESULT = circulo;
                break;
             case "cuadrado":
                Cuadrado cuadrado = new Cuadrado(nombre.toString(), (double)ex1, (double)ex2,
(double)ex3, c.toString());
                RESULT = cuadrado;
                break;
              default:
                errores.add(new Error(f.toString(), 0, 0, "figura invalida: " + f.toString(),
"sintactico"));
```

```
        }
    :}
    | figuraG:f PARENTECIS_ABRE NOMBRE:nombre COMA expresion:ex1 COMA
expresion:ex2 COMA expresion:ex3 COMA expresion:ex4 COMA color:c PARENTECIS_CIERRA
    {:

      switch (f.toString()) {
        case "rectangulo":
          Rectangulo rectangulo = new Rectangulo(nombre.toString(), (double)ex1,
(double)ex2, (double)ex3, (double)ex4, c.toString());
          RESULT = rectangulo;
          break;
        case "linea":
          Linea linea = new Linea(nombre.toString(), (double)ex1, (double)ex2, (double)ex3,
(double)ex4, c.toString());
          RESULT = linea;
          break;
        default:
          errores.add(new Error(f.toString(), 0, 0, "figura invalida: " + f.toString(),
"sintactico"));
      }
    :}
    | figuraG:f PARENTECIS_ABRE NOMBRE:nombre COMA expresion:ex1 COMA
expresion:ex2 COMA expresion:ex3 COMA expresion:ex4 COMA expresion:ex5 COMA color:c
PARENTECIS_CIERRA
    {:
    if(!f.toString().equals("poligono") ){
        errores.add(new Error(f.toString(), 0, 0, "figura invalida: " + f.toString(), "sintactico"));
```

```
        }
          Poligono poligono = new Poligono(nombre.toString(), (double)ex1, (double)ex2,
(double)ex3, (double)ex4, (double)ex5, c.toString());
          RESULT = poligono;
        :}
        ;

figuraG ::= CIRCULO {: RESULT = "circulo"; :}
   | RECTANGULO {: RESULT = "rectangulo"; :}
   | LINEA {: RESULT = "linea"; :}
   | CUADRADO {: RESULT = "cuadrado"; :}
   | POLIGONO {: RESULT = "poligono"; :};

color ::= MORADO {: RESULT = "Morado"; :}
       | ROSA {: RESULT = "Rosa"; :}
       | NARANJA {: RESULT = "Naranja"; :}
       | GRIS {: RESULT = "Gris"; :}
       | NEGRO {: RESULT = "Negro"; :}
       | VERDE {: RESULT = "Verde"; :}
       | AMARILLO {: RESULT = "Amarillo"; :}
       | ROJO {: RESULT = "Rojo"; :}
       | AZUL {: RESULT = "Azul"; :};

expresion ::= expresion:e1 PLUS expresion:e2 {: RESULT = (Double) e1 + (Double) e2; :}
          | expresion:e1 MINUS expresion:e2 {: RESULT = (Double) e1 - (Double) e2; :}
          | expresion:e1 TIMES expresion:e2 {: RESULT = (Double) e1 * (Double) e2; :}
          | expresion:e1 DIVIDE expresion:e2 {: RESULT = (Double) e1 / (Double) e2; :}
```

| PARENTECIS_ABRE expresion:e PARENTECIS_CIERRA {: RESULT = (Double) e; :}

| NUMERO:f {: RESULT = Double.valueOf(f.toString()); :};

Expresiones para el analisis Léxico:

L = [a-zA-Z]

D = [0-9]

G = [_-]

PARENTECIS_ABRE = [\(]

PARENTECIS_CIERRA = [\)]

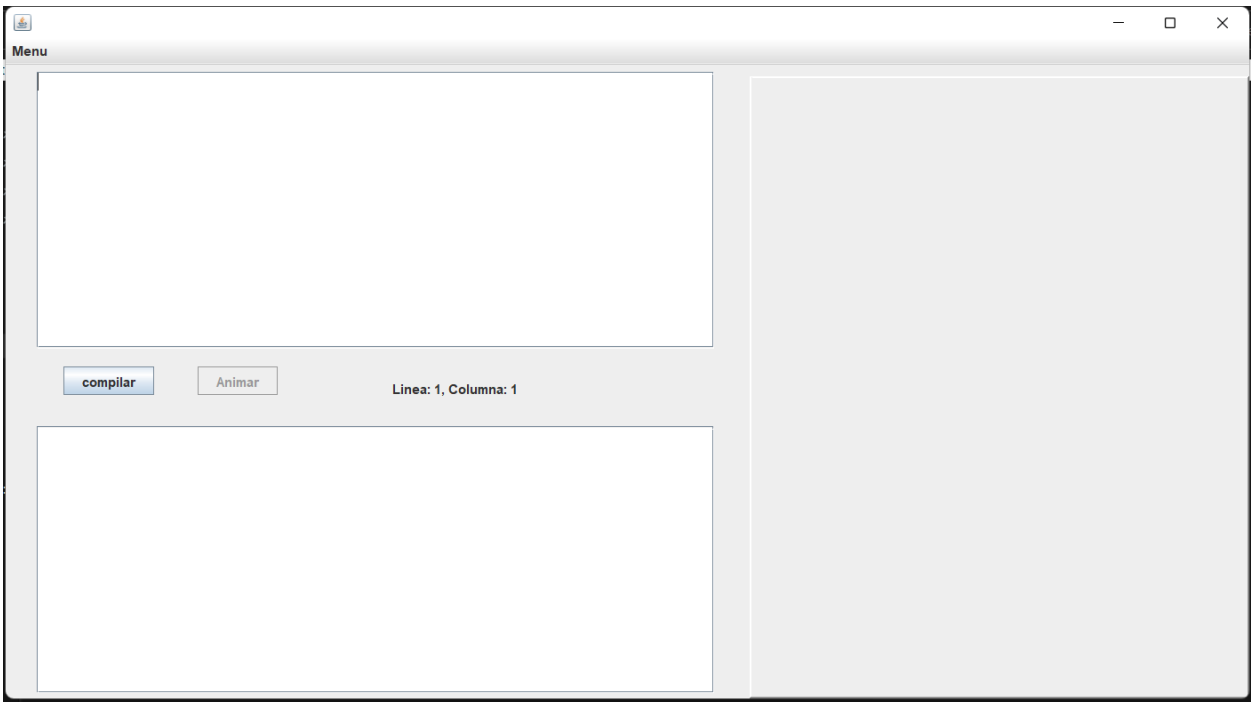
COMA = [,]

NUMERO = {D}+ ("." {D}+)?

NOMBRE = {L}({L}|{D}|{G})*

También operadores matemáticos y palabras propias del lenguaje como algunos colores y otras necesarias para la sintaxis.

Visual del proyecto:



Requisitos Mínimos para el Uso del Proyecto

1. Requisitos de Hardware

1. **Procesador**:
    - Procesador de al menos 1 GHz.
    - Procesador compatible con Java.
2. **Memoria RAM**:
    - Mínimo de 2 GB de RAM.
3. **Espacio en Disco**:
    - Espacio libre de al menos 100 MB para el proyecto y sus dependencias.
    - Espacio adicional para los archivos exportados (PNG, PDF).
4. **Resolución de Pantalla**:
    - Resolución mínima de 1024x768 píxeles para una visualización adecuada de la interfaz gráfica.

2. Requisitos de Software

1. **Sistema Operativo**:
    - Compatible con Windows, macOS o Linux.
    - Asegúrate de que el sistema operativo tenga soporte para la versión de Java que utilizas.
2. **Java Development Kit (JDK)**:
    - JDK 8 o superior.
    - Puedes descargarlo desde el sitio oficial de Oracle o usar una distribución OpenJDK.
3. **Entorno de Ejecución de Java (JRE)**:
    - JRE 8 o superior instalado si solo se ejecutará el programa (no es necesario para el desarrollo).
4. **Bibliotecas Necesarias**:
    - **JFlex**: Para el análisis léxico.

- **CUP**: Para el análisis sintáctico.
- **iText**: Biblioteca para la generación de PDFs. Debe estar incluida en el directorio /lib.

5. **Apache Ant**:
   - Versión 1.10 o superior para la construcción del proyecto.
   - Puedes descargarlo desde el sitio oficial de Apache Ant.
6. **IDE (Opcional)**:
   - **NetBeans** o **IntelliJ IDEA** para desarrollo, aunque no es estrictamente necesario si se utiliza Ant para la construcción del proyecto.