Figuras Geometricas

Manual Tecnico

Versión: 0001

Fecha: 20/08/2024

[Versión: 0001]

HOJA DE CONTROL

Organismo	CUNOC		
Proyecto	Figuras Geométricas		
Entregable	Manual Tecnico		
Autor	Selvyn Estuardo Ixtabalan Tistoj		
Versión/Edición	0001	**Fecha Versión**	20/08/2024
Aprobado por	Aux. Héctor	**Fecha Aprobación**	20/08/2024
		Nº Total de Páginas	13

REGISTRO DE CAMBIOS

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
0001	Versión inicial	Selvyn Estuardo Ixtabalan Tistoj	20/08/2024

CONTROL DE DISTRIBUCIÓN

Nombre y Apellidos
Selvyn Estuardo Ixtabalan Tistoj

ÍNDICE

1 DESCRIPCIÓN GENERAL SOLUCION

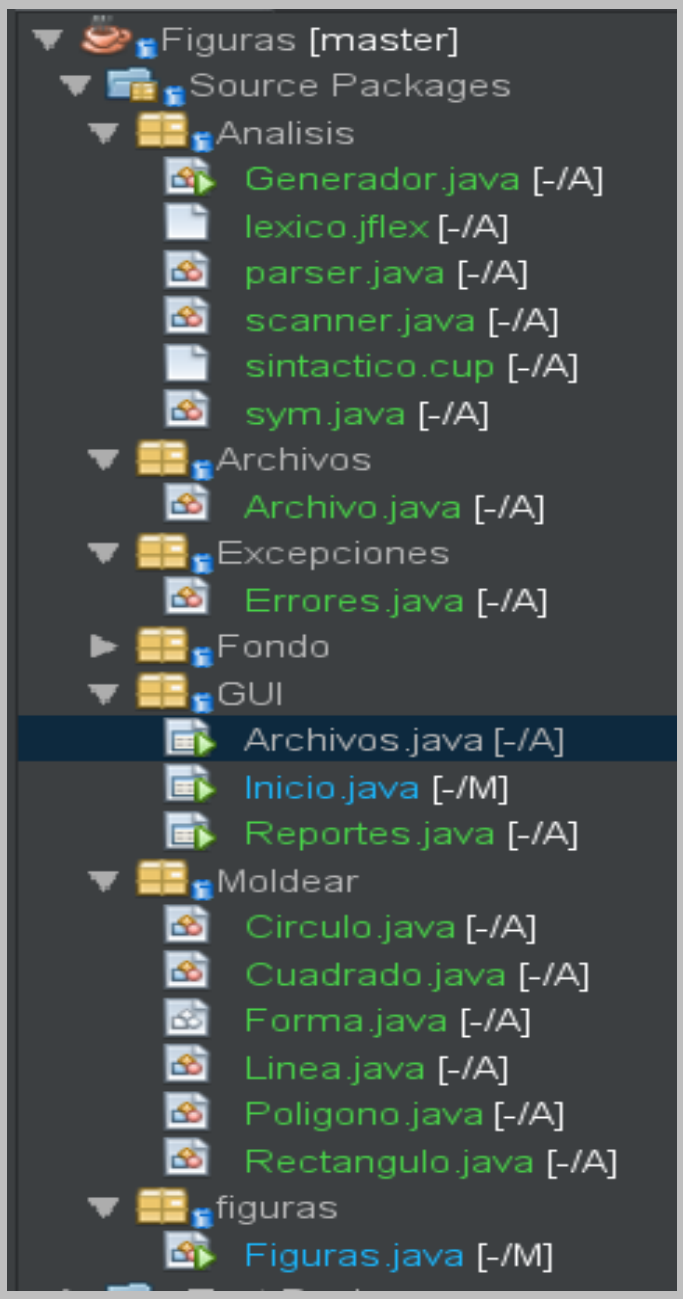
La aplicación de escritorio fue desarrollada con NetBeans, porque permite proporcionar una vista amigable al usuario y obtener información relacionada a los recursos del sistema mediante instrucciones y procedimientos en Swing. NetBeans con la ayuda de swing proporciona la comunicación correcta entre la interfaz y el código funcional de Java, mostrando detalles como el porcentaje de uso de CPU, el espacio de disco ocupado, el espacio en disco disponible, y el espacio de disco total en tiempo real, esta comunicación en tiempo real se realiza mediante eventos, con las instrucciones Event “e” enviar un mensaje de actualización y Alert desde la interfaz para que funcione como un Mensaje de Información o Error. La interfaz fue realizada con SWING, porque permite crear diseños muy llamativos y fáciles de implementar.

2 REQUERIMIENTOS MINIMOS

- Una distribución de Windows 10 o 11 (Windows 8.1 recomendado).
- JDK en versión compatible con NetBeans (NetBeans Linux/Windows recomendado)
- NetBeans en versión compatible con JDK (V. 17 recomendado)
- Procesador: Intel Core I5
- Memoria RAM: 4 GB
- Espacio de disco duro disponible: 8 GB

En equipos con características similares o mejores debe funcionar sin problema.

3 DICCIONARIO CLASES



4 DICCIONARIO DE METODOS

1. Main: El llamado del inicio para mostrar la primera ventana.
2. Generar: Se utiliza para poder generarlos y poder pasarlo al método de inicio.
3. Scanner: Me indicia los tipos de token que conformaran ese reconocimiento de tokens
4. Parser: Indica si toda la gramática es correcta.

5. Expresiones Regulares y Gramáticas:

GRAMATICA JFLEX:

```
package Analisis;

import java_cup.runtime.*;
import java.util.LinkedList;
import Excepciones.Errores;

%%

%{
    public LinkedList<Errores> listaErrores = new LinkedList<>();
%}

%init{
  yyline = 1;
  yycolumn = 1;
%init}

%cup
%class scanner
%public
%line
%char
%full
%ignorecase
```

```
PAR1 = "("
PAR2 = ")"
COMA = ","
SUMA = "+"
RESTA = "-"
MULTIPLICACION = "*"
DIVISION = "/"
ID = [a-zA-Z_][a-zA-Z0-9_]*
BLANCOS = [\ \rt\f\n]+
ENTERO = [0-9]+
DECIMAL=[0-9]+"."[0-9]+

// Palabras Reservadas
GRAFICAR = "graficar"
CIRCULO = "circulo"
CUADRADO = "cuadrado"
RECTANGULO = "rectangulo"
LINEA = "linea"
POLIGONO = "poligono"
ANIMAR = "animar"
OBJETO = "objeto"
ANTERIOR = "anterior"
CURVA = "curva"
LINEAL = "lineal"
```

```
AZUL = "azul"
ROJO = "rojo"
AMARILLO = "amarillo"
VERDE = "verde"
MORADO = "morado"
NARANJA = "naranja"
GRIS = "gris"
TURQUESA = "turquesa"
FUSIA = "fusia"

%%

// Palabras reservadas
<YYINITIAL> {GRAFICAR} {return new Symbol(sym.GRAFICAR, yyline, yycolumn, yytext());}
<YYINITIAL> {CIRCULO} {return new Symbol(sym.CIRCULO, yyline, yycolumn, yytext());}
<YYINITIAL> {CUADRADO} {return new Symbol(sym.CUADRADO, yyline, yycolumn, yytext());}
<YYINITIAL> {RECTANGULO} {return new Symbol(sym.RECTANGULO, yyline, yycolumn, yytext());}
<YYINITIAL> {LINEA} {return new Symbol(sym.LINEA, yyline, yycolumn, yytext());}
<YYINITIAL> {POLIGONO} {return new Symbol(sym.POLIGONO, yyline, yycolumn, yytext());}
<YYINITIAL> {ANIMAR} {return new Symbol(sym.ANIMAR, yyline, yycolumn, yytext());}
<YYINITIAL> {CURVA} {return new Symbol(sym.CURVA, yyline, yycolumn, yytext());}
<YYINITIAL> {LINEAL} {return new Symbol(sym.LINEAL, yyline, yycolumn, yytext());}
<YYINITIAL> {OBJETO} {return new Symbol(sym.OBJETO, yyline, yycolumn, yytext());}
<YYINITIAL> {ANTERIOR} {return new Symbol(sym.ANTERIOR, yyline, yycolumn, yytext());}
```

```
// Colores
<YYINITIAL> {AZUL} {return new Symbol(sym.COLOR, yytext());}
<YYINITIAL> {ROJO} {return new Symbol(sym.COLOR, yytext());}
<YYINITIAL> {AMARILLO} {return new Symbol(sym.COLOR, yytext());}
<YYINITIAL> {VERDE} {return new Symbol(sym.COLOR, yytext());}
<YYINITIAL> {MORADO} {return new Symbol(sym.COLOR, yytext());}
<YYINITIAL> {NARANJA} {return new Symbol(sym.COLOR, yytext());}
<YYINITIAL> {GRIS} {return new Symbol(sym.COLOR, yytext());}
<YYINITIAL> {TURQUESA} {return new Symbol(sym.COLOR, yytext());}
<YYINITIAL> {FUSIA} {return new Symbol(sym.COLOR, yytext());}

// Identificadores y Números
<YYINITIAL> {ID} {return new Symbol(sym.ID, yyline, yycolumn, yytext());}
<YYINITIAL> {DECIMAL} {return new Symbol(sym.DECIMAL, yyline, yycolumn, yytext());}
<YYINITIAL> {ENTERO} {return new Symbol(sym.ENTERO, yyline, yycolumn, yytext());}

// Operadores y Delimitadores
<YYINITIAL> {PAR1} {return new Symbol(sym.PAR1, yyline, yycolumn, yytext());}
<YYINITIAL> {PAR2} {return new Symbol(sym.PAR2, yyline, yycolumn, yytext());}
<YYINITIAL> {COMA} {return new Symbol(sym.COMA, yyline, yycolumn, yytext());}
<YYINITIAL> {SUMA} {return new Symbol(sym.SUMA, yyline, yycolumn, yytext());}
<YYINITIAL> {RESTA} {return new Symbol(sym.RESTA, yyline, yycolumn, yytext());}
<YYINITIAL> {MULTIPLICACION} {return new Symbol(sym.MULTIPLICACION, yyline, yycolumn, yytext());}
<YYINITIAL> {DIVISION} {return new Symbol(sym.DIVISION, yyline, yycolumn, yytext());}

// Manejo de espacios y saltos de línea
<YYINITIAL> {BLANCOS} {/* Ignorar espacios en blanco */}
```

```
// Manejo de caracteres no válidos
<YYINITIAL> . {
    listaErrores.add(new Errores("LEXICO", "El caracter " + yytext() + " no pertenece al lenguaje", yyline, yyco
}

. { System.out.println("Token no reconocido: " + yytext()); }
```

GRAMATICA CUP

```
package Analisis;

import java_cup.runtime.*;
import java.util.LinkedList;
import java.util.ArrayList;
import java.awt.Color;
import Moldear.*;
import Excepciones.Errores;

parser code
{:
    public static ArrayList<Forma> formas = new ArrayList<>();

    public static LinkedList<Errores> listaErrores = new LinkedList<>();

   public void syntax_error (Symbol s){
    listaErrores.add(new Errores("SINTACTICO RECUPERABLE", "No se esperaba el componente " + s.value, s.left, s.r

   }

   public void unrecovered_syntax_error (Symbol s){
    listaErrores.add(new Errores("SINTACTICO No RECUPERABLE", "No se esperaba el componente " + s.value, s.left,
   }

:}
```

```
action code
{:
    private Color getColor(String colorName) {
    switch (colorName.toLowerCase()) {
        case "fusia": return new Color(255, 0, 255);  // Fucsia
        case "turquesa": return new Color(64, 224, 208);  // Turquesa
        case "gris": return new Color(128, 128, 128);  // Gris
        case "naranja": return new Color(255, 165, 0);  // Naranja
        case "morado": return new Color(128, 0, 128);  // Morado
        case "verde": return new Color(0, 255, 0);  // Verde
        case "amarillo": return new Color(255, 255, 0);  // Amarillo
        case "rojo": return new Color(255, 0, 0);  // Rojo
        case "azul": return new Color(0, 0, 255);  // Azul
        default: return Color.BLACK;  // Color por defecto
    }
}
:}

//Terminales
terminal GRAFICAR, CIRCULO, CUADRADO, RECTANGULO, LINEA, POLIGONO, ANIMAR, OBJETO, ANTERIOR, OBJETO_ANTERIOR;
terminal String COLOR, ANIMAR_TIPO_LINEAL, ANIMAR_TIPO_CURVA, LINEAL, CURVA;
terminal SUMA, RESTA, MULTIPLICACION, DIVISION, PAR1, PAR2, COMA;
terminal String ID;
terminal Integer ENTERO;
terminal Double DECIMAL;
```

```
//NO Terminal
non terminal ArrayList<Forma> program;
non terminal Forma shape;
non terminal statement, animar;
non terminal Double expresion;

precedence left SUMA, RESTA;
precedence left MULTIPLICACION, DIVISION;
precedence left PAR1, PAR2;


program ::= statement program
          | statement
          {:
            System.out.println("Añadiendo forma al programa");
           RESULT = parser.formas;
           :}
          ;

statement ::= shape:f PAR2
            {: parser.formas.add(f); :}
            | animar PAR2
            ;
```

```
shape ::= GRAFICAR CIRCULO PAR1 ID:id COMA expresion:x COMA expresion:y COMA  expresion:r COMA COLOR:c PAR2
        {:
         System.out.println("Producción Círculo detectada con ID: " + id + " y color: " + c);
         RESULT = new Circulo(id, (int)Math.round(x), (int)Math.round(y), (int)Math.round(r), getColor(c));
        :}
        | GRAFICAR CUADRADO PAR1 ID:id COMA expresion:x COMA expresion:y COMA expresion:l COMA COLOR:c PAR2
        {: RESULT = new Cuadrado(id, (int)Math.round(x), (int)Math.round(y), (int)Math.round(l), getColor(c));
        :}
        | GRAFICAR RECTANGULO PAR1 ID:id COMA expresion:x COMA expresion:y COMA expresion:w COMA expresion:h CO
        {: RESULT = new Rectangulo(id, (int)Math.round(x), (int)Math.round(y), (int)Math.round(w), (int)Math.ro
        :}
        | GRAFICAR LINEA PAR1 ID:id COMA expresion:x1 COMA expresion:y1 COMA expresion:x2 COMA expresion:y2 COM
        {: RESULT = new Linea(id, (int)Math.round(x1), (int)Math.round(y1), (int)Math.round(x2), (int)Math.roun
        :}
        | GRAFICAR POLIGONO PAR1 ID:id COMA expresion:x COMA expresion:y COMA expresion:s COMA expresion:w COMA
        {:
          int[] puntosX = { (int)Math.round(x), (int)Math.round(x + s), (int)Math.round(x + s), (int)Math.round
          int[] puntosY = { (int)Math.round(y), (int)Math.round(y), (int)Math.round(y + h), (int)Math.round(y +

          RESULT = new Poligono(id, puntosX, puntosY, 4, (int)Math.round(x), (int)Math.round(y), getColor(c));
        :}
;

//ANIIMACION
```

```
expresion ::= ENTERO:e {: RESULT = e.doubleValue(); :}
            | DECIMAL:d {: RESULT = d; :}
            | expresion:e1 SUMA expresion:e2 {: RESULT = e1 + e2; :}
            | expresion:e1 RESTA expresion:e2 {: RESULT = e1 - e2; :}
            | expresion:e1 MULTIPLICACION expresion:e2 {: RESULT = e1 * e2; :}
            | expresion:e1 DIVISION expresion:e2 {: RESULT = e1 / e2; :}
            | PAR1 expresion:e PAR2 {: RESULT = e; :}
;
```