MANUAL TÉCNICO

Descripción General del Programa:

Este es un programa desarrollado en el lenguaje de programación de Java. Este nos permite el poder dibujar una variedad de figuras geométricas a través de comandos interactivos con los que podremos personalizar los atributos de cada una de las figuras que queramos dibujar.

Este se basa en el uso de las herramientas de jflex y cup para la detección de dichos comandos y poder ejecutar las acciones que el usuario requiera.

Se tendrá una interfaz amigable que le permita al usuario el poder manejarse comodamente a través del programa y así poder tener una experiencia agradable durante la ejecución del programa.

Descripción General de las Herramientas:

jFlex: Es un generador de analizadores léxicos (también conocido como lexer o scanner). Toma como entrada un archivo de especificación en el que defines patrones regulares para reconocer tokens (como palabras clave, identificadores, operadores, etc.). A partir de esta especificación, JFlex genera un código Java que puede leer una secuencia de caracteres de entrada y dividirla en estos tokens.

Cup: Es un generador de analizadores sintácticos (o parser) que trabaja con gramáticas definidas en una forma similar a BNF (Backus-Naur Form). CUP toma como entrada una gramática que describe la estructura del lenguaje y produce un analizador sintáctico en Java. Este parser puede entonces procesar la secuencia de tokens generada por JFlex para construir una representación estructurada (como un árbol sintáctico) del código de entrada.

Java: Java es un lenguaje de programación de propósito general, orientado a objetos, y diseñado para ser independiente de la plataforma. Java es conocido por su simplicidad, seguridad, robustez, y por tener una rica API que facilita el desarrollo de aplicaciones desde pequeñas hasta sistemas grandes y complejos. Es ampliamente utilizado en el desarrollo de aplicaciones empresariales, móviles (particularmente en Android), y sistemas distribuidos.

Descripción General de los Métodos Más Importantes Utilizados:

Graficos:

- **repintar():** Forza una actualización visual del componente jPanelFiguras
- **graficarCirculo():** Grafica un circulo tomando en cuenta el radio y la posicion.
- **graficarCuadrado(), graficarRectangulo():** Grafican cuadriláteros que se basan en el tamaño de los lados y su posición.
- **graficarLinea():** Grafica una línea de un punto A a un punto B.
- **graficarPoligono():** Dibuja un polígono regular de n lados.

Lectura:

- **AgregarOperador, AgregarColor, AgregarErrores, AgregarMovimiento:** Agrega tales datos a unas listas específicas para su uso y manejo.

Reportes:

- **AgregarDatos:** Este método llena cuatro tablas diferentes con datos provenientes de cuatro listas distintas.

ReporteError:

- **AgregarErrores:** Con este método se llena una tabla que da información sobre errores.

Gramáticas Utilizadas:

jFlex:

```
//      OPERADORES LOGICOS

Suma =      "+"

Resta =     "-"

Multi =     "*"

Divisn =    "/"
```

Con estas producciones se detectan las operaciones aritméticas en el programa

```
//       COLORES DISPONIBLES

amarillo =  "amarillo"

azul =      "azul"

celeste =   "celeste"

morado =    "morado"

naranja =   "naranja"

rojo =      "rojo"

rosado =    "rosado"

turquesa =  "turquesa"

verde =     "verde"
```

Estas producciones detectan los colores disponibles, que serán tratadas como palabras reservadas.

```
//      FIGURAS

circulo =   "circulo"

cuadrado =  "cuadrado"

rectangulo = "rectangulo"

poligono =  "poligono"
```

```
linea =     "linea"

//      MOVIMIENTO DE LAS FIGURAS

Mov_linea = "linea"

Mov_curva = "curva"
```

Así mismo, las figuras disponibles y los tipos de movimiento se tratarán igual, como palabras reservadas en el programa.

```
//      INSTRUCCIONES

graficar =  "graficar"

animar =    "animar"

objeto = "objeto"

anterior = "anterior"
```

Las instrucciones también serán tomadas como palabras reservadas.

```
//      UTILIDADES

Digito = [0-9]

Numeros = {Digito}+

Flotantes = {Numeros}"."{Numeros}

Par_Opn =   "("

Par_Cls =   ")"

Coma = ","

nombre = [a-zA-Z0-9_][a-zA-Z0-9_]*

Espacio = \s*(\r\n|\r|\n)
```

Por ultimo tenemos las diferentes utilidades en el programa, como la deteccion de números enteros, números flotantes, paréntesis, comas, identificadores y espacios en blanco. Estos nos sirven para poder utilizar el programa de manera comoda y segura.

Cup

```
start with instruccion;

instruccion ::= graficar instruccion

                | animar instruccion
```

Las producciones del cup, inician con la posible detección de una instrucción que sería la de animar o graficar.

```
color ::= COLOR:color {:

    Color col = Color.black;

    switch((String)color){

        case "amarillo":

            col =  Color.YELLOW;

            break;

        case "azul":

            col =  Color.BLUE;

            break;

        case "celeste":

            col = Color.CYAN;

            break;

        case "morado":

            col =  new Color(124,51,255);

            break;

        case "naranja":

            col =  Color.ORANGE;

            break;

        case "rojo":

            col =  Color.RED;
```

```
            break;

        case "rosado":

            col =  Color.PINK;

            break;

        case "turquesa":

            col =  new Color(88,215,234);

            break;

        case "verde":

            col =  Color.GREEN;

            break;

    }

    graficos.lectura.AgregarColor((String)color);

    RESULT = col;

:};
```

Estas producciones detectan el color que se usará en la figura actual, y devuelven como resultado el color para el uso de la librería awt, y nos agrega el color utilizado a la lista para recopilar datos.

```
entero ::= FLOTANTES:numero {: RESULT =
(int)Math.round((double)numero); :};

entero ::= NUMEROS:numero  {: RESULT = numero; :};

entero ::= entero:num1 SUMA entero:num2 {:

                                        String cadena = ""; cadena
+=(int)num1; cadena += "+"; cadena += (int)num2;


graficos.lectura.AgregarOperador(cadena, "suma", cur_token.left,
cur_token.right);

                                        RESULT = (int)num1 + (int)num2;
:}
```

```
        |  entero:num1 RESTA entero:num2 {:

                                        String cadena = ""; cadena
+=(int)num1; cadena += "-"; cadena += (int)num2;


graficos.lectura.AgregarOperador(cadena, "resta", cur_token.left,
cur_token.right);

                                        RESULT = (int)num1 - (int)num2;
:};

entero ::= entero:num1 MULTI entero:num2 {:

                                        String cadena = ""; cadena
+=(int)num1; cadena += "*"; cadena += (int)num2;


graficos.lectura.AgregarOperador(cadena, "multiplicacion",
cur_token.left, cur_token.right);

                                        RESULT = (int)num1 *
(int)num2;:}

        | entero:num1 DIVISN entero:num2 {:

                                        String cadena = ""; cadena
+=(int)num1; cadena += "/"; cadena += (int)num2;


graficos.lectura.AgregarOperador(cadena, "division", cur_token.left,
cur_token.right);

                                        RESULT = (int)num1 /
(int)num2;:};

entero ::= PAROPN entero:num PARCLS {:RESULT = (int)num; :};
```

La detección de números engloba también las posibles operaciones que tengan que realizar, teniendo una precedencia ordenada para su correcto uso. También envía los datos de las operaciones utilizadas a la lista indicada para la recolección de datos.

```
graficar ::= GRAFICAR CIRCULO PAROPN ID:id COMA entero:posx COMA
entero:posy COMA entero:radio COMA color:col PARCLS

            {: graficos.graficarCirculo((int)posx, (int)posy,
(int)radio, (Color)col);
```

```
                graficos.lectura.AgregarFigura("Circulo");:}

          | GRAFICAR CUADRADO PAROPN ID:id COMA entero:posx COMA
entero:posy COMA entero:tam COMA color:col PARCLS

            {: graficos.graficarCuadrado((int)posx, (int)posy,
(int)tam, (Color)col);

                graficos.lectura.AgregarFigura("Cuadrado");:}

          | GRAFICAR RECTANGULO PAROPN ID:id COMA entero:posx COMA
entero:posy COMA entero:ancho  COMA entero:alto COMA color:col PARCLS

            {: graficos.graficarRectangulo((int)posx, (int)posy,
(int)ancho, (int)alto, (Color)col );

                graficos.lectura.AgregarFigura("Rectangulo");:}

          | GRAFICAR LINEA PAROPN ID:id COMA entero:x1 COMA entero:y1
COMA entero:x2 COMA entero:y2 COMA color:col PARCLS

            {: graficos.graficarLinea((int)x1, (int)y1, (int)x2,
(int)y2, (Color)col);

                graficos.lectura.AgregarFigura("Linea");:}

          | GRAFICAR POLIGONO PAROPN ID:id COMA entero:posx COMA
entero:posy COMA entero:cant COMA entero:ancho COMA entero:alto COMA
color:col PARCLS

            {: graficos.graficarPoligono((int) posx, (int)posy,
(int)cant, (int)ancho, (int)alto, (Color)col);

                graficos.lectura.AgregarFigura("Poligono");:};
```

La instrucción de graficar puede detectar todos los tipos de figuras y envía los datos correspondientes a los métodos graficadores para su aplicación. También recaba datos para el uso en los reportes.

```
animar ::= ANIMAR OBJETO ANTERIOR PAROPN LINEA COMA entero COMA entero
COMA entero PARCLS {: graficos.lectura.AgregarMovimiento("Lineal"); :}

        |  ANIMAR OBJETO ANTERIOR PAROPN CURVA COMA entero COMA entero
COMA entero PARCLS {: graficos.lectura.AgregarMovimiento("Curva"); :};
```

Por último, la función animar recaba los datos necesarios para el tipo de animación y para poder recopilar los datos de los reportes.

```
public void syntax_error(Symbol cur_token) {

        System.out.println("Simbolo con error:" +
symbl_name_from_id(cur_token.sym));

        System.out.println("Linea " + cur_token.left);

        System.out.println("Columna " + cur_token.right);



graficos.lectura.AgregarErrores(symbl_name_from_id(cur_token.sym),
cur_token.left, cur_token.right);



    }
```

A pesar de no ser una producción, veo necesario el identificar el manejo de errores dentro del archivo cup. Esta función detecta posibles errores en la entrada y con funciones propias de cup, podemos recopilar sus datos y enviarlo a los reportes.