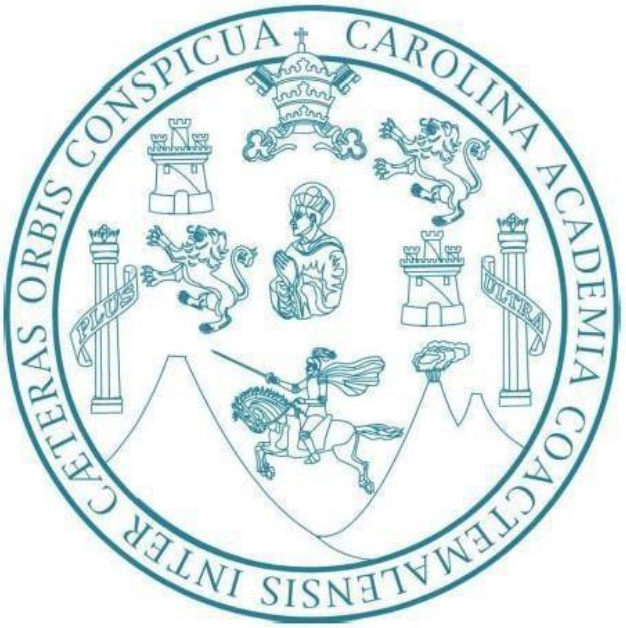
Universidad de San Carlos de Guatemala
División de Ciencias de la Ingeniería
Organización de Lenguajes y Compiladores 1



DOCUMENTACIÓN PRÁCTICA 1
ANALIZADOR LÉXICO Y SINTÁCTICO

Estudiante:
Carné: 201831260
Nombre: Michael Kristopher Marín Reyes

Sección: A

Contenido

GESTOR DE BIBLIOTECA

La siguiente documentación muestra la estructura de código en la cual se construyó el analizador léxico y sintáctico.

Paquete main

Este paquete contiene tres carpetas importantes, una es el cup que contiene el archivo de configuración para el analizador sintáctico, java que contiene todas las clases para el funcionamiento de la aplicación y flex que contiene el archivo de configuración para el analizador léxico. A continuación, se muestra la descripción de cada paquete:

Paquete main/cup

Este paquete contiene el archivo de configuración de para el analizador sintáctico.

parser.cup

Este archivo contiene las gramáticas formales y métodos el cual se utilizan para analizar los tokens recibidos del analizador léxico. Este archivo genera la clase Parser y ParserSym las cuales se describirán más adelante. A continuación, se describe cada parte del archivo cup el cual permite su funcionamiento:

Campos:

- Figura ultimaFigura: Almacena la última figura creada.
- List<Figura> listaObjetosList: Lista que almacena todas las figuras creadas.

Métodos

- public List<Figura> getListaObjetosList(): Devuelve la lista de figuras.
- public void report_error(String message, Object info): Reporta un error sintáctico y lo agrega a

Lexer.errores.

- public void report_fatal_error(String message, Object info) throws Exception: Reporta un error sintáctico fatal y lanza una excepción.
- public void graficar(): Muestra una ventana con las figuras graficadas y opciones para guardar como PNG o PDF.
- public void animar(String animacion): Ejecuta una animación en la última figura creada.

Producciones

Terminales:

- Palabras clave y operadores: PARENTESIS_ABRE, PARENTESIS_CIERRA, COMA, AZUL, ROJO, AMARILLO, VERDE, NEGRO, CELESTE, ROSADO, MORADO, LINEA, CURVA, GRAFICAR, CIRCULO, CUADRADO, RECTANGULO, LINEA_OBJ, POLIGONO, ANIMAR, OBJETO, ANTERIOR, NUMERO.
- Identificadores y números: String ID, Double SUMA, RESTA, MULTIPLICACION, DIVISION.
- No-terminales: programa, instruccion, animar, forma, color, tipoanimacion, inicial, errorSintactico, Double expresion

Precedencia de Operadores

- Suma y Resta: precedence left SUMA, RESTA
- Multiplicación y División: precedence left MULTIPLICACION, DIVISION

Reglas de Gramática

- Inicial: inicial ::= programa;
- Programa: Permite múltiples instrucciones o maneja errores.
- Instrucción: Permite graficar o animar una figura.
- Forma: Define la sintaxis para crear diferentes tipos de figuras (círculo, cuadrado, línea, rectángulo, polígono).
- Color: Define los colores disponibles.
- Animar: Permite animar la última figura creada.
- Tipo de Animación: Define los tipos de animación disponibles (curva o línea).
- Expresiones: Maneja operaciones aritméticas y paréntesis.

Paquete main/java

Este paquete contiene todas las clases necesarias que logran el funcionamiento de la aplicación. Esta carpeta contiene dos paquetes más llamados backend y frontend continuación se describe cada paquete y clase que contiene.

Paquete main/java/backend

Este paquete contiene toda la parte de la lógica que permite el funcionamiento de la aplicación, y como maneja los datos recibidos por el frontend. A continuación, se describen las clases dentro de este paquete y los paquetes que contiene.

Clase LeerArchivoTexto

La clase `LeerArchivoTexto` se encarga de abrir un archivo de texto desde una ruta específica y leer su contenido, retornándolo como una cadena de texto. Es útil para la lectura de archivos de texto plano, como archivos `.txt`.

Atributos

- `private String path`: Almacena la ruta del archivo que se va a leer.

Métodos

- **Método** `abrirArchivo`: Este método abre un archivo de texto ubicado en la ruta especificada, lo lee línea por línea, y concatena su contenido en una sola cadena de texto.
- **Parámetros**:
  `String ruta`: La ruta del archivo que se va a leer.
- **Retorno**:
  `String`: El contenido del archivo en formato de cadena de texto, con cada línea separada por un salto de línea (`\n`).
- **Manejo de Excepciones**: Se maneja una excepción `IOException` en caso de que ocurra un error al intentar leer el archivo. En caso de error, se imprime el mensaje de la excepción en la consola.

Clase Main

La clase `Main` se encarga de iniciar la aplicación al crear una instancia de la ventana principal de la interfaz gráfica y hacerla visible para el usuario.

Métodos

- **Método** `main`: Este es el método principal que Java invoca al iniciar la aplicación. Dentro de este método se crea una instancia de la clase `VentanaPrincipal`, se centra la ventana en la pantalla y se hace visible.
- **Parámetros**:

  `String[] args`: Un array de cadenas que contiene los argumentos de línea de comandos pasados al programa. En este caso, no se utilizan.
- **Funcionamiento**:
  - Se crea una instancia de `VentanaPrincipal`.
  - Se llama al método `setLocationRelativeTo(null)` para centrar la ventana en la pantalla.
  - Finalmente, se llama al método `setVisible(true)` para mostrar la ventana.

Paquete main/java/backend/figuras

Este paquete contiene las clases que dan forma a las figuras que se generan a través de ciertos comandos.

Clase Circulo

La clase Circulo representa una figura geométrica de tipo círculo y extiende la clase Figura. Proporciona métodos para definir y manipular las propiedades del círculo, así como para graficarlo.

Atributos

- private String nombre: Nombre del círculo.
- private double posicionX: Coordenada X de la posición del círculo.
- private double posicionY: Coordenada Y de la posición del círculo.
- private double radio: Radio del círculo.
- private String color: Color del círculo en formato de texto.

Constructor

Crea una nueva instancia de Circulo con los valores especificados para nombre, posición, radio y color.

- **Parámetros**:
  - String nombre: Nombre del círculo.
  - double posicionX: Coordenada X de la posición del círculo.
  - double posicionY: Coordenada Y de la posición del círculo.
  - double radio: Radio del círculo.

- String color: Color del círculo.

Métodos

- **Getters y Setters**:
    - public String getNombre(): Devuelve el nombre del círculo.
    - public void setNombre(String nombre): Establece el nombre del círculo.
    - public double getPosicionX(): Devuelve la coordenada X del círculo.
    - public void setPosicionX(double posicionX): Establece la coordenada X del círculo.
    - public double getPosicionY(): Devuelve la coordenada Y del círculo.
    - public void setPosicionY(double posicionY): Establece la coordenada Y del círculo.
    - public double getRadio(): Devuelve el radio del círculo.
    - public void setRadio(double radio): Establece el radio del círculo.
    - public String getColor(): Devuelve el color del círculo.
    - public void setColor(String color): Establece el color del círculo.
- **Método de la clase** Figura:
    - @Override public void paint(Graphics graphics): Método que imprime un mensaje en la consola indicando que se está graficando un círculo. En una implementación completa, este método debería contener el código para dibujar el círculo en el componente gráfico.

Clase Cuadrado

La clase Cuadrado representa una figura geométrica de tipo cuadrado y extiende la clase Figura. Proporciona métodos para definir y manipular las propiedades del cuadrado, así como para graficarlo.

Atributos

- private String nombre: Nombre del cuadrado.
- private double posicionX: Coordenada X de la posición del cuadrado.
- private double posicionY: Coordenada Y de la posición del cuadrado.
- private double tamanio: Tamaño del lado del cuadrado.
- private String color: Color del cuadrado en formato de texto.

Constructor

Crea una nueva instancia de Cuadrado con los valores especificados para nombre, posición, tamaño y color.

- **Parámetros**:
    - String nombre: Nombre del cuadrado.
    - double posicionX: Coordenada X de la posición del cuadrado.
    - double posicionY: Coordenada Y de la posición del cuadrado.
    - double tamanio: Tamaño del lado del cuadrado.
    - String color: Color del cuadrado.

Métodos

- **Getters y Setters**:
    - public String getNombre(): Devuelve el nombre del cuadrado.
    - public void setNombre(String nombre): Establece el nombre del cuadrado.
    - public double getPosicionX(): Devuelve la coordenada X del cuadrado.
    - public void setPosicionX(double posicionX): Establece la coordenada X del cuadrado.
    - public double getPosicionY(): Devuelve la coordenada Y del cuadrado.
    - public void setPosicionY(double posicionY): Establece la coordenada Y del cuadrado.
    - public double getTamanio(): Devuelve el tamaño del lado del cuadrado.
    - public void setTamanio(double tamañoLado): Establece el tamaño del lado del cuadrado. Nota: El parámetro tiene un nombre diferente en el setter (tamañoLado en lugar de tamanio).
    - public String getColor(): Devuelve el color del cuadrado.
    - public void setColor(String color): Establece el color del cuadrado.
- **Método de la clase** Figura:
    - @Override public void paint(Graphics graphics): Método que imprime un mensaje en la consola indicando que se está dibujando un cuadrado. En una implementación completa, este método debería contener el código para dibujar el cuadrado en el componente gráfico.

Clase Figura

La clase abstracta Figura sirve como una clase base para todas las figuras geométricas. Define los atributos comunes y métodos abstractos que deben ser implementados por las clases concretas que extienden Figura.

Atributos

- private String nombre: Nombre de la figura. Este atributo puede ser usado para identificar la figura.
- private String color: Color de la figura en formato de texto.

Métodos

- **Métodos Abstractos**:
    - public abstract void paint(Graphics graphics): Método abstracto que debe ser implementado por las subclases para definir cómo se debe dibujar la figura. Recibe un objeto Graphics que se usa para realizar el dibujo en el componente gráfico.
- **Getters**:
    - public String getColor(): Devuelve el color de la figura.
    - public String getNombre(): Devuelve el nombre de la figura.

Clase Linea

La clase Linea representa una figura geométrica de tipo línea y extiende la clase Figura. Proporciona métodos para definir y manipular las propiedades de una línea, así como para graficarla.

Atributos

- private String nombre: Nombre de la línea.
- private double posicionInicialX: Coordenada X del punto inicial de la línea.
- private double posicionInicialY: Coordenada Y del punto inicial de la línea.
- private double posicionFinalX: Coordenada X del punto final de la línea.
- private double posicionFinalY: Coordenada Y del punto final de la línea.
- private String color: Color de la línea en formato de texto.

Constructor

Crea una nueva instancia de Linea con los valores especificados para nombre, posición inicial, posición final y color.

- **Parámetros**:
    - String nombre: Nombre de la línea.
    - double posicionInicialX: Coordenada X del punto inicial de la línea.
    - double posicionInicialY: Coordenada Y del punto inicial de la línea.
    - double posicionFinalX: Coordenada X del punto final de la línea.
    - double posicionFinalY: Coordenada Y del punto final de la línea.
    - String color: Color de la línea.

Métodos

- **Getters y Setters**:
    - public String getNombre(): Devuelve el nombre de la línea.
    - public void setNombre(String nombre): Establece el nombre de la línea.
    - public double getPosicionInicialX(): Devuelve la coordenada X del punto inicial de la línea.
    - public void setPosicionInicialX(double posicionInicialX): Establece la coordenada X del punto inicial de la línea.
    - public double getPosicionInicialY(): Devuelve la coordenada Y del punto inicial de la línea.
    - public void setPosicionInicialY(double posicionInicialY): Establece la coordenada Y del punto inicial de la línea.
    - public double getPosicionFinalX(): Devuelve la coordenada X del punto final de la línea.
    - public void setPosicionFinalX(double posicionFinalX): Establece la coordenada X del punto final de la línea.
    - public double getPosicionFinalY(): Devuelve la coordenada Y del punto final de la línea.

    - public void setPosicionFinalY(double posicionFinalY): Establece la coordenada Y del punto final de la línea.
    - public String getColor(): Devuelve el color de la línea.
    - public void setColor(String color): Establece el color de la línea.
- **Método de la clase** Figura:
    - @Override public void paint(Graphics graphics): Método que imprime un mensaje en la consola indicando que se está dibujando una línea. En una implementación completa, este método debería contener el código para dibujar la línea en el componente gráfico.

Clase Poligono

La clase Poligono representa una figura geométrica de tipo polígono y extiende la clase Figura. Ofrece métodos para definir y manipular las propiedades de un polígono, así como para graficarlo.

Atributos

- private String nombre: Nombre del polígono.
- private double posicionX: Coordenada X del centro del polígono.
- private double posicionY: Coordenada Y del centro del polígono.
- private int cantidadLados: Número de lados del polígono.
- private double ancho: Ancho del polígono.
- private double alto: Alto del polígono.
- private String color: Color del polígono en formato de texto.

Constructor

Crea una nueva instancia de Poligono con los valores especificados para nombre, posición, cantidad de lados, tamaño y color.

- **Parámetros**:
    - String nombre: Nombre del polígono.
    - double posicionX: Coordenada X del centro del polígono.
    - double posicionY: Coordenada Y del centro del polígono.
    - int cantidadLados: Número de lados del polígono.
    - double ancho: Ancho del polígono.
    - double alto: Alto del polígono.
    - String color: Color del polígono.

Métodos

- **Getters y Setters**:
    - public String getNombre(): Devuelve el nombre del polígono.
    - public void setNombre(String nombre): Establece el nombre del polígono.

    - public double getPosicionX(): Devuelve la coordenada X del centro del polígono.
    - public void setPosicionX(double posicionX): Establece la coordenada X del centro del polígono.
    - public double getPosicionY(): Devuelve la coordenada Y del centro del polígono.
    - public void setPosicionY(double posicionY): Establece la coordenada Y del centro del polígono.
    - public int getCantidadLados(): Devuelve el número de lados del polígono.
    - public void setCantidadLados(int cantidadLados): Establece el número de lados del polígono.
    - public double getAncho(): Devuelve el ancho del polígono.
    - public void setAncho(double ancho): Establece el ancho del polígono.
    - public double getAlto(): Devuelve el alto del polígono.
    - public void setAlto(double alto): Establece el alto del polígono.
    - public String getColor(): Devuelve el color del polígono.
    - public void setColor(String color): Establece el color del polígono.
- **Método de la clase** Figura:
    - @Override public void paint(Graphics graphics): Método que imprime un mensaje en la consola indicando que se está dibujando un polígono. En una implementación completa, este método debería contener el código para dibujar el polígono en el componente gráfico.

Clase Rectangulo

La clase Rectangulo representa una figura geométrica de tipo rectángulo y extiende la clase Figura. Ofrece métodos para definir y manipular las propiedades de un rectángulo, así como para graficarlo.

Atributos

- private String nombre: Nombre del rectángulo.
- private double posicionX: Coordenada X del rectángulo.
- private double posicionY: Coordenada Y del rectángulo.
- private double ancho: Ancho del rectángulo.
- private double alto: Alto del rectángulo.
- private String color: Color del rectángulo en formato de texto.

Constructor

Crea una nueva instancia de **Rectangulo** con los valores especificados para nombre, posición, tamaño y color.

- **Parámetros**:
    - String nombre: Nombre del rectángulo.

- double posicionX: Coordenada X del rectángulo.
- double posicionY: Coordenada Y del rectángulo.
- double ancho: Ancho del rectángulo.
- double alto: Alto del rectángulo.
- String color: Color del rectángulo.

Métodos

- **Getters y Setters**:
    - public String getNombre(): Devuelve el nombre del rectángulo.
    - public void setNombre(String nombre): Establece el nombre del rectángulo.
    - public double getPosicionX(): Devuelve la coordenada X del rectángulo.
    - public void setPosicionX(double posicionX): Establece la coordenada X del rectángulo.
    - public double getPosicionY(): Devuelve la coordenada Y del rectángulo.
    - public void setPosicionY(double posicionY): Establece la coordenada Y del rectángulo.
    - public double getAncho(): Devuelve el ancho del rectángulo.
    - public void setAncho(double ancho): Establece el ancho del rectángulo.
    - public double getAlto(): Devuelve el alto del rectángulo.
    - public void setAlto(double alto): Establece el alto del rectángulo.
    - public String getColor(): Devuelve el color del rectángulo.
    - public void setColor(String color): Establece el color del rectángulo.
- **Método de la clase** Figura:
    - @Override public void paint(Graphics graphics): Método que imprime un mensaje en la consola indicando que se está dibujando un rectángulo. En una implementación completa, este método debería contener el código para dibujar el rectángulo en el componente gráfico.

Paquete main/java/backend/lexico

Este paquete tiene dos clases que se encargan de apoyar la función del analizador léxico.

Clase OperadorAritmetico

La clase OperadorAritmetico representa un operador aritmético en un análisis léxico. Incluye información sobre el tipo de operador, su símbolo, y la posición en el código fuente donde aparece, así como la cantidad de veces que ocurre.

Atributos

- private String operador: El nombre del operador aritmético (e.g., "suma", "resta").
- private String simbolo: El símbolo que representa el operador (e.g., "+", "-").
- private int linea: El número de línea en el código fuente donde aparece el operador.

- private int columna: El número de columna en el código fuente donde aparece el operador.
- private int ocurrencia: El número de veces que el operador aparece en el código fuente.

Constructor
Crea una nueva instancia de OperadorAritmetico con los valores especificados para el operador, símbolo, línea, columna y ocurrencia.

- **Parámetros**:
    - String operador: El nombre del operador aritmético.
    - String simbolo: El símbolo que representa el operador.
    - int linea: La línea en el código fuente donde aparece el operador.
    - int columna: La columna en el código fuente donde aparece el operador.
    - int ocurrencia: La cantidad de veces que el operador aparece en el código fuente.

Métodos
- **Getters y Setters**:
    - public String getOperador(): Devuelve el nombre del operador aritmético.
    - public void setOperador(String operador): Establece el nombre del operador aritmético.
    - public String getSimbolo(): Devuelve el símbolo del operador.
    - public void setSimbolo(String simbolo): Establece el símbolo del operador.
    - public int getLinea(): Devuelve el número de línea donde aparece el operador.
    - public void setLinea(int linea): Establece el número de línea donde aparece el operador.
    - public int getColumna(): Devuelve el número de columna donde aparece el operador.
    - public void setColumna(int columna): Establece el número de columna donde aparece el operador.
    - public int getOcurrencia(): Devuelve la cantidad de veces que el operador aparece en el código fuente.
    - public void setOcurrencia(int ocurrencia): Establece la cantidad de veces que el operador aparece en el código fuente.

Clase Token

La clase Token representa un token identificado durante el proceso de análisis léxico. Un token es una unidad de información que tiene un tipo, una posición en el código fuente y un lexema asociado.

Atributos
- private String tipo: Representa el tipo de token, como una palabra reservada, un identificador, un operador, etc.
- private int linea: La línea en el código fuente donde se encontró el token.

- private int columna: La columna en el código fuente donde se encontró el token.
- private String lexema: El lexema es la representación textual del token, es decir, la cadena de caracteres en el código fuente que corresponde al token.

Constructor
Crea una nueva instancia de Token con los valores especificados para el tipo, lexema, línea y columna.

- **Parámetros**:
    - String tipo: El tipo del token.
    - String lexema: El lexema del token.
    - int linea: La línea en el código fuente donde se encontró el token.
    - int columna: La columna en el código fuente donde se encontró el token.

Métodos
- **Getters y Setters**:
    - public String getTipo(): Devuelve el tipo del token.
    - public void setTipo(String tipo): Establece el tipo del token.
    - public int getLinea(): Devuelve el número de línea donde se encontró el token.
    - public void setLinea(int linea): Establece el número de línea donde se encontró el token.
    - public int getColumna(): Devuelve el número de columna donde se encontró el token.
    - public void setColumna(int columna): Establece el número de columna donde se encontró el token.
    - public String getLexema(): Devuelve el lexema del token.
    - public void setLexema(String lexema): Establece el lexema del token.
- **Método** toString:
    - @Override public String toString(): Devuelve una representación en cadena del token, que incluye su tipo, línea, columna y lexema.

Paquete /main/java/backend/movimientos

Este paquete contiene la forma en que las figuras se pueden mover, estas son en curva y línea.

Clase CurvaMovimiento

La clase CirculoMovimiento extiende la clase Thread y se encarga de manejar el movimiento de un círculo en un área bidimensional. El círculo se moverá dentro de un marco determinado y rebotará cuando alcance los límites especificados.

Atributos

- private int posicionX: Coordenada X actual del círculo.
- private int posicionY: Coordenada Y actual del círculo.
- private int movimientoX: Desplazamiento en la dirección X en cada paso de movimiento.
- private int movimientoY: Desplazamiento en la dirección Y en cada paso de movimiento.
- private int distanciaMaximaX: Límite máximo en la dirección X dentro del cual el círculo puede moverse.
- private int distanciaMaximaY: Límite máximo en la dirección Y dentro del cual el círculo puede moverse.

Constructor

Constructor por defecto que inicializa un objeto CirculoMovimiento.

Métodos

- **Getters y Setters**:
    - public int getPosicionX(): Devuelve la coordenada X actual del círculo.
    - public void setPosicionX(int posicionX): Establece la coordenada X del círculo.
    - public int getPosicionY(): Devuelve la coordenada Y actual del círculo.
    - public void setPosicionY(int posicionY): Establece la coordenada Y del círculo.
    - public int getMovimientoX(): Devuelve el desplazamiento en la dirección X.
    - public void setMovimientoX(int movimientoX): Establece el desplazamiento en la dirección X.
    - public int getMovimientoY(): Devuelve el desplazamiento en la dirección Y.
    - public void setMovimientoY(int movimientoY): Establece el desplazamiento en la dirección Y.
    - public int getDistanciaMaximaX(): Devuelve el límite máximo en la dirección X.
    - public void setDistanciaMaximaX(int distanciaMaximaX): Establece el límite máximo en la dirección X.
    - public int getDistanciaMaximaY(): Devuelve el límite máximo en la dirección Y.
    - public void setDistanciaMaximaY(int distanciaMaximaY): Establece el límite máximo en la dirección Y.
- **Método** mover: Actualiza las coordenadas posicionX y posicionY del círculo según los valores de movimientoX y movimientoY.
- **Método** run: Método principal de la hebra que maneja el movimiento continuo del círculo. El círculo rebota cuando alcanza los límites especificados por distanciaMaximaX y distanciaMaximaY. El método usa un bucle infinito para continuar el movimiento y hace una pausa de 10 milisegundos entre cada actualización de la posición.
- **Manejo de Excepciones**: Si ocurre una interrupción durante el Thread.sleep(10), se captura y se imprime el stack trace.

Paquete /main/java/backend/reportes

Este paquete contiene las clases que son objetos cuando se detecta un error léxico o sintáctico.

Clase ErrorReporte

La clase ErrorReporte se utiliza para representar los errores que pueden ocurrir durante el análisis léxico o sintáctico. Esta clase encapsula la información relevante del error, como el lexema que lo causó, la ubicación en el código (línea y columna), el tipo de error, y una descripción detallada del mismo.

Atributos

- private String lexema: El lexema que causó el error.
- private int linea: La línea en el código donde ocurrió el error.
- private int columna: La columna en la línea donde ocurrió el error.
- private String tipo: El tipo de error (léxico o sintáctico).
- private String descripcion: Descripción detallada del error.

Constructor

Inicializa un objeto ErrorReporte con todos los detalles necesarios para describir un error.

- **Parámetros**:
    - String lexema: El lexema asociado con el error.
    - int linea: La línea donde se encontró el error.
    - int columna: La columna donde se encontró el error.
    - String tipo: El tipo de error (léxico o sintáctico).
    - String descripcion: Una descripción detallada del error.

Métodos

- **Getters y Setters**:
    - public String getLexema(): Devuelve el lexema asociado con el error.
    - public void setLexema(String lexema): Establece el lexema asociado con el error.
    - public int getLinea(): Devuelve la línea donde ocurrió el error.
    - public void setLinea(int linea): Establece la línea donde ocurrió el error.
    - public int getColumna(): Devuelve la columna donde ocurrió el error.
    - public void setColumna(int columna): Establece la columna donde ocurrió el error.
    - public String getTipo(): Devuelve el tipo de error.
    - public void setTipo(String tipo): Establece el tipo de error.
    - public String getDescripcion(): Devuelve la descripción del error.
    - public void setDescripcion(String descripcion): Establece la descripción del error.

- **Método** toString: Devuelve una representación en formato de cadena de texto del objeto ErrorReporte, mostrando todos sus atributos.

Clase ErrorSintacticoReporte

Paquete main/java/backend/sintactico

Las clases generadas por el archivo parser.cup se almacenan en este paquete.

Clase Parser

Contiene los métodos generados por el archivo parser.cup, donde cada método sirve para recibir los tokens enviados por la clase Lexer, además de compararlos en las gramáticas y verificar si son correctos.

Clase ParserSym

Esta clase también es generada por el archivo parser.cup.

Definición de Constantes de Terminales

- Cada constante pública (public static final int) representa un terminal específico en la gramática de tu lenguaje. Estos valores son utilizados por el parser para identificar los diferentes tipos de tokens.
- Ejemplo: public static final int LINEA = 13; define el terminal LINEA con el valor 13.

Array de Nombres de Terminales

- public static final String[] terminalNames es un array que mapea los valores numéricos de los terminales a sus nombres en texto. Esto es útil para depuración y generación de mensajes informativos.
- Ejemplo: "LINEA" se corresponde con el valor 13 en el array de terminales.

Paquete main/java/frontend

Este paquete contiene todas las clases que conforman la interfaz gráfica que el usuario ve, este paquete contiene más paquetes.

Clase EditorPanel

EditorPanel extiende JPanel y está diseñado para ser un editor de texto básico con soporte para la ejecución de análisis léxico y sintáctico, así como para la limpieza del área de edición. Este panel

también muestra la columna actual del cursor y permite al usuario limpiar todo el contenido del editor con un botón o una tecla específica.

Atributos

- JTextPane areaEditor: Área de texto donde el usuario puede escribir. Está configurada con un fondo gris claro.
- JScrollPane scrollEditor: Proporciona desplazamiento al área de texto.
- JButton limpiarBoton: Botón que limpia el contenido del editor.
- JButton ejecutarBoton: Botón que ejecuta el análisis léxico y sintáctico.
- JLabel mostrarColumnaLabel: Etiqueta que muestra la posición actual de la columna del cursor en el editor.

Métodos Principales

- **Constructor** EditorPanel:
    - Configura el layout del panel, inicializa componentes, y añade un listener para la columna del cursor.
    - Incluye un KeyAdapter que permite limpiar todo el contenido del editor presionando la tecla F4.
- ejecutarBotonActionPerformed(java.awt.event.ActionEvent evt):
    - Verifica si el editor está vacío.
    - Si hay texto, crea un objeto Lexer y Parser para realizar el análisis léxico y sintáctico del contenido.
    - Muestra una excepción en la consola si ocurre un error.
- limpiarBotonActionPerformed(java.awt.event.ActionEvent evt): Limpia el contenido del editor y restablece los atributos de estilo del documento.
- limpiarBotonKeyPressed(java.awt.event.KeyEvent evt): Permite limpiar todo el contenido del editor cuando se presiona la tecla F4.
- LimpiarTodo(): Método auxiliar que simplemente limpia el contenido del editor.
- MostrarColumna(): Agrega un CaretListener al JTextPane para actualizar la posición de la columna en la etiqueta mostrarColumnaLabel cada vez que el cursor se mueve.
- setAreaEditor(String textoLeido): Establece el contenido del areaEditor con un texto proporcionado.

Clase NumeroLinea

Es una clase que extiende JPanel. Esta clase se utiliza para crear un panel que muestra el número de línea correspondiente en un área de texto, similar a un editor de texto. El panel de números de línea se coloca junto al componente de texto y muestra el número de línea actual resaltado en color.

Constructor
Genera el panel donde se escribirá el texto.

Métodos

- **NumeroLinea(JTextComponent component, int minimumDisplayDigits):** Permite especificar el número mínimo de dígitos para la numeración de líneas.
- **GetUpdateFont():** Devuelve si la actualización de la fuente está habilitada.
- **setUpdateFont(boolean updateFont):** Se encarga de habilitar o deshabilitar la actualización de la fuente.
- **GetBorderGap():** Devuelve el espacio entre el borde del panel y los números de línea.
- **setBorderGap(int borderGap):** Establece el espacio entre el borde del panel y los números de línea, recalculando el ancho preferido del panel.
- **GetCurrentLineForeground():** Devuelve el color de primer plano para la línea actual.
- **setCurrentLineForeground(Color currentLineForeground):** Establece el color de primer plano para la línea actual.
- **GetDigitAlignment():** Devuelve la alineación de los números de línea (izquierda, centro o derecha).
- **setDigitAlignment(float digitAlignment):** Establece la alineación de los números de línea.
- **getMinimumDisplayDigits():**Devuelve el número mínimo de dígitos para la numeración de líneas.
- **setMinimumDisplayDigits(int minimumDisplayDigits):** Establece el número mínimo de dígitos para la numeración de líneas.
- **paintComponent(Graphics g):** Método para dibujar los números de línea en el panel. Itera a través de las líneas visibles y pinta los números correspondientes.
- **isCurrentLine(int rowStartOffset):** Comprueba si la línea dada es la línea actual, en función de la posición del cursor.
- **getTextLineNumber(int rowStartOffset):** Obtiene el número de línea para una posición de inicio de línea dada.
- **getOffsetX(int availableWidth, int stringWidth):** Calcula el desplazamiento X necesario para alinear correctamente el número de línea.
- **getOffsetY(int rowStartOffset, FontMetrics fontMetrics):** Calcula el desplazamiento Y necesario para alinear correctamente el número de línea verticalmente.
- **caretUpdate(CaretEvent e):** Maneja los eventos de cambio de posición del cursor, actualizando el resaltado del número de línea actual.
- **changedUpdate(DocumentEvent e):** Manejan eventos de cambio en el documento de texto, actualizando los números de línea en respuesta a las modificaciones del contenido.
- **insertUpdate(DocumentEvent e):** Manejan eventos de cambio en el documento de texto, actualizando los números de línea en respuesta a las modificaciones del contenido.
- **removeUpdate(DocumentEvent e):** Manejan eventos de cambio en el documento de texto, actualizando los números de línea en respuesta a las modificaciones del contenido.
- **DocumentChanged():** Realiza cambios en la numeración de líneas en respuesta a los cambios en el documento.
- **propertyChange(PropertyChangeEvent evt):** Maneja los cambios en las propiedades del componente, como la fuente, para actualizar la apariencia de los números de línea.

Clase VentanaPrincipal

La clase VentanaPrincipal es una implementación de la interfaz gráfica de usuario para la aplicación "Generador de Figuras". Esta clase extiende javax.swing.JFrame y proporciona una interfaz gráfica que permite al usuario importar archivos de texto y generar diferentes tipos de reportes relacionados con figuras, colores y errores.

Atributos

- private EditorPanel editor:
    - **Descripción**: Panel de editor de texto utilizado para mostrar y editar el contenido de archivos de texto.
- **Componentes Generados**:
    - contenedorPanel: Panel principal que muestra los componentes de la interfaz.
    - menuOpciones: Barra de menú que contiene opciones para importar archivos y generar reportes.
    - abrirBoton: Botón para importar archivos de texto.
    - editorTexto: Menú para acceder al panel del editor de texto.
    - reportesBoton: Menú para acceder a diferentes tipos de reportes.
    - reportesOperadoresMatematicos: Elemento de menú para el reporte de operadores matemáticos.
    - reporteColoresUsados: Elemento de menú para el reporte de colores usados.
    - reporteObjetosUsados: Elemento de menú para el reporte de objetos usados.
    - reporteAnimacionesUsadas: Elemento de menú para el reporte de animaciones usadas.
    - reporteErrores: Elemento de menú para el reporte de errores.

Constructor

- public VentanaPrincipal()
    - **Descripción**: Inicializa la ventana principal de la aplicación, configura el título y los componentes visuales. También establece el panel de editor de texto como el panel actual.
    - **Acciones**: Configura el título de la ventana como "Generador de Figuras" y llama al método initComponents() para configurar los componentes del formulario.

Métodos

initComponents()

- **Descripción**: Método generado automáticamente para inicializar y configurar los componentes del formulario. Incluye la configuración del menú, los botones y el panel de contenido.
- **Detalles**:
    - Configura la apariencia y comportamiento de los menús y botones.
    - Configura el panel de contenido (contenedorPanel) con un fondo amarillo y un diseño de borde.

abrirBotonActionPerformed(java.awt.event.ActionEvent evt)

- **Descripción**: Acción asociada al botón "Importar archivo" del menú "Archivo". Permite al usuario seleccionar un archivo de texto y carga su contenido en el panel de editor.
- **Detalles**:
    - Utiliza JFileChooser para abrir un cuadro de diálogo de selección de archivo.
    - Filtra los archivos para permitir solo archivos .txt.
    - Si se selecciona un archivo válido, lee su contenido y lo muestra en el panel de editor.

editorTextoMouseClicked(java.awt.event.MouseEvent evt)

- **Descripción**: Acción asociada al clic en el menú "Editor". Actualiza el panel para mostrar el editor de texto.
- **Detalles**:
    - Llama al método pintarPanel() para mostrar el panel de editor de texto.

reportesOperadoresMatematicosActionPerformed(java.awt.event.ActionEvent evt)

- **Descripción**: Acción asociada al menú "Operadores matemáticos" en el menú de reportes. Muestra el reporte de operadores matemáticos.
- **Detalles**:
    - Crea una instancia de OperadoresMatematicos y la muestra en el panel de contenido.

reporteColoresUsadosActionPerformed(java.awt.event.ActionEvent evt)

- **Descripción**: Acción asociada al menú "Colores usados" en el menú de reportes. Muestra el reporte de colores usados.
- **Detalles**:
    - Crea una instancia de ColoresUsados y la muestra en el panel de contenido.

reporteObjetosUsadosActionPerformed(java.awt.event.ActionEvent evt)

- **Descripción**: Acción asociada al menú "Objetos usados" en el menú de reportes. Muestra el reporte de objetos usados.
- **Detalles**:
    - Crea una instancia de ObjetosUsados y la muestra en el panel de contenido.

reporteAnimacionesUsadasActionPerformed(java.awt.event.ActionEvent evt)

- **Descripción**: Acción asociada al menú "Animaciones usadas" en el menú de reportes. Muestra el reporte de animaciones usadas.
- **Detalles**:
    - Crea una instancia de AnimacionesUsadas y la muestra en el panel de contenido.

reporteErroresActionPerformed(java.awt.event.ActionEvent evt)

- **Descripción**: Acción asociada al menú "Errores" en el menú de reportes. Muestra el reporte de errores.
- **Detalles**:
    - Crea una instancia de ReporteErrores y la muestra en el panel de contenido.

pintarPanel(Component panel)

- **Descripción**: Actualiza el panel de contenido (contenedorPanel) para mostrar el componente proporcionado.

- **Detalles**:
    - Elimina todos los componentes actuales del panel de contenido.
    - Añade el nuevo componente y actualiza el panel para reflejar los cambios.

Paquete main/java/frontend/graficas

Esta clase contiene las gráficas que se generarán haciendo uso de las clases Figura.

Clase GraficaCirculo

La clase GraficaCirculo extiende JPanel y se utiliza para representar y dibujar un círculo en un panel gráfico. Permite definir la posición, el radio y el color del círculo mediante métodos de acceso y modificación. El círculo se dibuja en el método paintComponent(Graphics g), donde se configuran sus atributos visuales.

Atributos

- **private double posicionX**: Coordenada X de la posición del círculo en el panel.
- **private double posicionY**: Coordenada Y de la posición del círculo en el panel.
- **private double radio**: Radio del círculo a dibujar.
- **private String color**: Color del círculo, representado como una cadena de texto. Puede ser "rojo", "azul", "verde", "negro", "amarillo", "celeste", "rosado", "morado", o cualquier otro valor para establecer el color de fondo blanco.

Constructor: Inicializa una instancia de GraficaCirculo y emite un mensaje en la consola para fines de depuración.

- **Parámetros**: Ninguno.
- **Acciones**: Imprime "cir" en la consola al crear una instancia del objeto.

Métodos

- **Getters y Setters**:
    - **public double getPosicionX()**: Devuelve la coordenada X de la posición del círculo.
    - **public void setPosicionX(double posicionX)**: Establece la coordenada X de la posición del círculo.
    - **public double getPosicionY()**: Devuelve la coordenada Y de la posición del círculo.
    - **public void setPosicionY(double posicionY)**: Establece la coordenada Y de la posición del círculo.
    - **public double getRadio()**: Devuelve el radio del círculo.
    - **public void setRadio(double radio)**: Establece el radio del círculo.
    - **public String getColor()**: Devuelve el color del círculo.

    - **public void setColor(String color)**: Establece el color del círculo. Acepta valores como "rojo", "azul", "verde", "negro", "amarillo", "celeste", "rosado", "morado", o cualquier otro valor para establecer el color de fondo blanco.
- **Método paintComponent**:
    - **Descripción**: Sobrescribe el método paintComponent de JPanel para dibujar el círculo en el panel.
    - **Parámetros**:
        - **Graphics g**: Objeto utilizado para dibujar en el panel.
    - **Acciones**:
        - Llama a super.paintComponent(g) para limpiar el panel antes de dibujar.
        - Configura el color de dibujo basado en el atributo color.
        - Dibuja el círculo con el método fillOval usando las coordenadas y el tamaño especificado.

Clase GraficaCuadrado

La clase GraficaCuadrado extiende JPanel y se utiliza para representar y dibujar un cuadrado en un panel gráfico. Permite definir la posición inicial, el tamaño y el color del cuadrado mediante métodos de acceso y modificación. El cuadrado se dibuja en el método paintComponent(Graphics g), donde se configuran sus atributos visuales.

Atributos

- **private double inicialX**: Coordenada X de la esquina superior izquierda del cuadrado en el panel.
- **private double inicialY**: Coordenada Y de la esquina superior izquierda del cuadrado en el panel.
- **private double tamanio**: Tamaño del lado del cuadrado.
- **private String color**: Color del cuadrado, representado como una cadena de texto. Puede ser "rojo", "azul", "verde", "negro", "amarillo", "celeste", "rosado", "morado", o cualquier otro valor para establecer el color de fondo blanco.

Constructor: Inicializa una instancia de GraficaCuadrado. El constructor por defecto no realiza ninguna acción adicional.

- **Parámetros**: Ninguno.

Métodos

- **Getters y Setters**:
    - **public double getInicialX()**: Devuelve la coordenada X de la esquina superior izquierda del cuadrado.

- **public void setInicialX(double inicialX)**: Establece la coordenada X de la esquina superior izquierda del cuadrado.
- **public double getInicialY()**: Devuelve la coordenada Y de la esquina superior izquierda del cuadrado.
- **public void setInicialY(double inicialY)**: Establece la coordenada Y de la esquina superior izquierda del cuadrado.
- **public double getTamanio()**: Devuelve el tamaño del lado del cuadrado.
- **public void setTamanio(double tamanio)**: Establece el tamaño del lado del cuadrado.
- **public String getColor()**: Devuelve el color del cuadrado.
- **public void setColor(String color)**: Establece el color del cuadrado. Acepta valores como "rojo", "azul", "verde", "negro", "amarillo", "celeste", "rosado", "morado", o cualquier otro valor para establecer el color de fondo blanco.

- **Método paintComponent**:
  - **Descripción**: Sobrescribe el método paintComponent de JPanel para dibujar el cuadrado en el panel.
  - **Parámetros**:
    - **Graphics g**: Objeto utilizado para dibujar en el panel.
  - **Acciones**:
    - Llama a super.paintComponent(g) para limpiar el panel antes de dibujar.
    - Configura el color de dibujo basado en el atributo color.
    - Dibuja el cuadrado con el método fillRect usando las coordenadas inicialX, inicialY, y el tamaño tamanio. La altura del cuadrado se establece como la longitud de la cadena del color, lo cual parece ser un error y debería ajustarse para que coincida con el tamaño del lado del cuadrado.

Clase GraficaLinea

La clase GraficaLinea extiende JPanel y se utiliza para representar y dibujar una línea en un panel gráfico. Permite definir las coordenadas de inicio y fin de la línea, así como su color mediante métodos de acceso y modificación. La línea se dibuja en el método paintComponent(Graphics g).

Atributos

- **private double inicialX**: Coordenada X del punto inicial de la línea en el panel.
- **private double inicialY**: Coordenada Y del punto inicial de la línea en el panel.
- **private double finalX**: Coordenada X del punto final de la línea en el panel.
- **private double finalY**: Coordenada Y del punto final de la línea en el panel.
- **private String color**: Color de la línea, representado como una cadena de texto. Puede ser "rojo", "azul", "verde", "negro", "amarillo", "celeste", "rosado", "morado", o cualquier otro valor para establecer el color de fondo blanco.

Constructor Descripción: Inicializa una instancia de GraficaLinea. El constructor por defecto no realiza ninguna acción adicional.

- **Parámetros**: Ninguno.

Métodos

- **Método estático paint**:
    - **Descripción**: Dibuja una línea en las coordenadas especificadas. Este método no se utiliza en el contexto de la clase GraficaLinea, ya que la línea se dibuja en el método paintComponent.
    - **Parámetros**:
        - **Graphics g**: Objeto utilizado para dibujar en el panel.
        - **int posicionX**: Coordenada X del punto inicial de la línea.
        - **int posicionY**: Coordenada Y del punto inicial de la línea.
        - **int x1**: Coordenada X del punto final de la línea.
        - **int y1**: Coordenada Y del punto final de la línea.
    - **Acciones**: Dibuja una línea usando g.drawLine(posicionX, posicionY, x1, y1).
- **Getters y Setters**:
    - **public double getInicialX()**: Devuelve la coordenada X del punto inicial de la línea.
    - **public void setInicialX(double inicialX)**: Establece la coordenada X del punto inicial de la línea.
    - **public double getInicialY()**: Devuelve la coordenada Y del punto inicial de la línea.
    - **public void setInicialY(double inicialY)**: Establece la coordenada Y del punto inicial de la línea.
    - **public double getFinalX()**: Devuelve la coordenada X del punto final de la línea.
    - **public void setFinalX(double finalX)**: Establece la coordenada X del punto final de la línea.
    - **public double getFinalY()**: Devuelve la coordenada Y del punto final de la línea.
    - **public void setFinalY(double finalY)**: Establece la coordenada Y del punto final de la línea.
    - **public String getColor()**: Devuelve el color de la línea.
    - **public void setColor(String color)**: Establece el color de la línea. Acepta valores como "rojo", "azul", "verde", "negro", "amarillo", "celeste", "rosado", "morado", o cualquier otro valor para establecer el color de fondo blanco.
- **Método paintComponent**:
    - **Descripción**: Sobrescribe el método paintComponent de JPanel para dibujar la línea en el panel.
    - **Parámetros**:
        - **Graphics g**: Objeto utilizado para dibujar en el panel.
    - **Acciones**:

- Llama a super.paintComponent(g) para limpiar el panel antes de dibujar.
- Configura el color de dibujo basado en el atributo color.
- Dibuja la línea usando el método fillRect con las coordenadas inicialX, inicialY, finalX, y finalY.

Clase GraficaPoligono

La clase GraficaPoligono extiende JPanel y se utiliza para representar y dibujar un polígono en un panel gráfico. Permite definir la posición, el número de lados, el tamaño, y el color del polígono. El polígono se dibuja en el método paintComponent(Graphics g).

Atributos

- **private double posicionX**: Coordenada X del centro del polígono en el panel.
- **private double posicionY**: Coordenada Y del centro del polígono en el panel.
- **private int cantLados**: Número de lados del polígono.
- **private double ancho**: Ancho del polígono, que determina el tamaño en el eje X.
- **private double alto**: Alto del polígono, que determina el tamaño en el eje Y.
- **private String color**: Color del polígono, representado como una cadena de texto. Puede ser "rojo", "azul", "verde", "negro", "amarillo", "celeste", "rosado", "morado", o cualquier otro valor para establecer el color de fondo blanco.

Constructor: Inicializa una instancia de GraficaPoligono. El constructor por defecto no realiza ninguna acción adicional.

- **Parámetros**: Ninguno.

Métodos

- **Getters y Setters**:
    - **public double getPosicionX()**: Devuelve la coordenada X del centro del polígono.
    - **public void setPosicionX(double posicionX)**: Establece la coordenada X del centro del polígono.
    - **public double getPosicionY()**: Devuelve la coordenada Y del centro del polígono.
    - **public void setPosicionY(double posicionY)**: Establece la coordenada Y del centro del polígono.
    - **public int getCantLados()**: Devuelve el número de lados del polígono.
    - **public void setCantLados(int cantLados)**: Establece el número de lados del polígono.
    - **public double getAncho()**: Devuelve el ancho del polígono.
    - **public void setAncho(double ancho)**: Establece el ancho del polígono.
    - **public double getAlto()**: Devuelve el alto del polígono.
    - **public void setAlto(double alto)**: Establece el alto del polígono.
    - **public String getColor()**: Devuelve el color del polígono.

    - **public void setColor(String color)**: Establece el color del polígono. Acepta valores como "rojo", "azul", "verde", "negro", "amarillo", "celeste", "rosado", "morado", o cualquier otro valor para establecer el color de fondo blanco.
- **Método paintComponent**:
    - **Descripción**: Sobrescribe el método paintComponent de JPanel para dibujar el polígono en el panel.
    - **Parámetros**:
        - **Graphics g**: Objeto utilizado para dibujar en el panel.
    - **Acciones**:
        - Llama a super.paintComponent(g) para limpiar el panel antes de dibujar.
        - Configura el color de dibujo basado en el atributo color.
        - Calcula los puntos del polígono usando coordenadas polares.
        - Crea un objeto Polygon con los puntos calculados.
        - Dibuja el contorno del polígono usando g2d.drawPolygon(polygon).
        - Rellena el polígono con el color especificado usando g2d.fillPolygon(polygon).

Clase GraficaRectangulo

La clase GraficaRectangulo extiende JPanel y se utiliza para representar un rectángulo en un panel gráfico. Permite definir la posición, el tamaño y el color del rectángulo. Esta clase incluye un constructor que inicializa los atributos del rectángulo.

Atributos

- **private double inicialX**: Coordenada X de la esquina superior izquierda del rectángulo.
- **private double inicialY**: Coordenada Y de la esquina superior izquierda del rectángulo.
- **private double ancho**: Ancho del rectángulo.
- **private double alto**: Alto del rectángulo.
- **private String color**: Color del rectángulo, representado como una cadena de texto. Puede ser cualquier valor válido para especificar el color.

Constructor: Inicializa una instancia de GraficaRectangulo con las coordenadas, dimensiones y color especificados.

- **Parámetros**:
    - **double posicionX**: Coordenada X de la esquina superior izquierda del rectángulo.
    - **double posicionY**: Coordenada Y de la esquina superior izquierda del rectángulo.
    - **double ancho**: Ancho del rectángulo.
    - **double alto**: Alto del rectángulo.
    - **String color**: Color del rectángulo.

Métodos

- **Getters y Setters**:

- **public double getInicialX()**: Devuelve la coordenada X de la esquina superior izquierda del rectángulo.
- **public void setInicialX(double inicialX)**: Establece la coordenada X de la esquina superior izquierda del rectángulo.
- **public double getInicialY()**: Devuelve la coordenada Y de la esquina superior izquierda del rectángulo.
- **public void setInicialY(double inicialY)**: Establece la coordenada Y de la esquina superior izquierda del rectángulo.
- **public double getAncho()**: Devuelve el ancho del rectángulo.
- **public void setAncho(double ancho)**: Establece el ancho del rectángulo.
- **public double getAlto()**: Devuelve el alto del rectángulo.
- **public void setAlto(double alto)**: Establece el alto del rectángulo.
- **public String getColor()**: Devuelve el color del rectángulo.
- **public void setColor(String color)**: Establece el color del rectángulo.

Clase PanelDibujo

La clase PanelDibujo extiende JPanel y se utiliza para dibujar figuras geométricas en un panel de una interfaz gráfica. Esta clase permite guardar el contenido gráfico en formatos PNG y PDF. Se encarga de pintar en el panel las figuras almacenadas en una lista, así como de proporcionar métodos para exportar la imagen actual del panel en diferentes formatos.

Atributos

- **public static List<Figura> listaFiguras**: Lista estática que contiene las figuras geométricas a ser dibujadas en el panel. Puede contener instancias de Circulo, Linea, Cuadrado, Rectangulo, y Poligono.

Constructores

- **PanelDibujo()**: Inicializa el panel con un color de fondo gris.

Métodos

- **public List<Figura> getListaFiguras()**: Devuelve la lista de figuras que se están dibujando en el panel.
- **public void setListaFiguras(List<Figura> listaFiguras)**: Establece la lista de figuras que se deben dibujar en el panel y repinta el panel.
- **protected void paintComponent(Graphics g)**: Sobrescribe el método paintComponent para dibujar cada figura en la lista listaFiguras. El método verifica el tipo de figura y la dibuja con el color correspondiente.
- **private Color devolverColor(String color)**: Convierte un nombre de color en una instancia de Color. Soporta varios colores predefinidos.

- **public void guardarComoPNG()**: Permite al usuario guardar el contenido del panel como una imagen PNG. Utiliza un JFileChooser para seleccionar la ubicación y el nombre del archivo.
- **public void guardarComoPDF()**: Permite al usuario guardar el contenido del panel como un archivo PDF. Utiliza un JFileChooser para seleccionar la ubicación y el nombre del archivo y la biblioteca iText para crear el archivo PDF.

Paquete main/java/frontend/reporetes

Este paquete contiene todas las tablas que se muestran en los reportes generados en cada compilación.

Clase AnimacionesUsadas

La clase AnimacionesUsadas extiende javax.swing.JPanel y se utiliza para mostrar un reporte en forma de tabla de las animaciones utilizadas y la cantidad de veces que cada una ha sido usada. La tabla está implementada usando JTable y su modelo de datos es un DefaultTableModel.

Constructores

- **AnimacionesUsadas()**: Inicializa el panel y actualiza la tabla con los datos de animaciones usadas.

Métodos

- **private void initComponents()**: Método autogenerado por el Form Editor para inicializar los componentes de la interfaz gráfica. Configura la tabla tablaAnimacionesUsadas dentro de un JScrollPane y define su disposición en el panel.
- **public void actualizarTablaAnimacionesUsadas()**: Actualiza la tabla tablaAnimacionesUsadas con los datos de las animaciones usadas. Actualmente, el método está configurado para crear el modelo de la tabla con dos columnas: "Animación" y "Cantidad de uso". La implementación para rellenar la tabla con datos aún no está completa.

Variables de Instancia

- **private javax.swing.JScrollPane jScrollPane1**: Componente que proporciona una vista desplazable para la tabla tablaAnimacionesUsadas.
- **private javax.swing.JTable tablaAnimacionesUsadas**: Tabla que muestra los datos de las animaciones usadas.

Clase ColoresUsados

La clase ColoresUsados extiende javax.swing.JPanel y se utiliza para mostrar un reporte en forma de tabla de los colores utilizados en las figuras y la cantidad de veces que cada color ha sido usado. La tabla está implementada usando JTable y su modelo de datos es un DefaultTableModel.

Constructores

- **ColoresUsados()**: Inicializa el panel y actualiza la tabla con los datos de colores usados.

Métodos

- **private void initComponents()**: Método autogenerado por el Form Editor para inicializar los componentes de la interfaz gráfica. Configura la tabla tablaColoresUsados dentro de un JScrollPane y define su disposición en el panel.
- **public void actualizarTablaColoresUsados()**: Actualiza la tabla tablaColoresUsados con los datos de los colores usados en las figuras. La tabla tiene dos columnas: "Color" y "Cantidad de uso". La implementación cuenta el número de veces que cada color aparece en la lista de figuras y actualiza la tabla con estos conteos.

Variables de Instancia

- **private javax.swing.JScrollPane jScrollPane1**: Componente que proporciona una vista desplazable para la tabla tablaColoresUsados.
- **private javax.swing.JTable tablaColoresUsados**: Tabla que muestra los datos de los colores usados.

Clase ObjetosUsados

La clase ObjetosUsados extiende javax.swing.JPanel y se encarga de mostrar un reporte en forma de tabla sobre la cantidad de cada tipo de figura que ha sido creada. Utiliza una JTable y un DefaultTableModel para presentar los datos.

Constructores

- **ObjetosUsados()**: Inicializa el panel y actualiza la tabla con los datos sobre los tipos de objetos usados.

Métodos

- **private void initComponents()**: Método autogenerado por el Form Editor para inicializar los componentes de la interfaz gráfica. Configura la tabla tablaObjetosUsados dentro de un JScrollPane y define su disposición en el panel.
- **public void actualizarTablaObjetosUsados()**: Actualiza la tabla tablaObjetosUsados con los datos de los tipos de figuras usadas. La tabla tiene dos columnas: "Objeto" y "Creados". El método cuenta cuántas veces ha sido creado cada tipo de figura en la lista de figuras del PanelDibujo y actualiza la tabla con estos conteos.

Variables de Instancia

- **private javax.swing.JScrollPane jScrollPane1**: Componente que proporciona una vista desplazable para la tabla tablaObjetosUsados.

- **private javax.swing.JTable tablaObjetosUsados**: Tabla que muestra los datos de los objetos usados.

Clase OperadoresMatematicos

La clase OperadoresMatematicos extiende javax.swing.JPanel y proporciona una interfaz gráfica para mostrar y filtrar una tabla de operadores matemáticos. Utiliza una JTable y un DefaultTableModel para mostrar los datos, y un TableRowSorter para permitir el filtrado de la tabla basado en una selección en un JComboBox.

Constructores

- **OperadoresMatematicos()**: Inicializa el panel, llena el JComboBox con las opciones de filtro y actualiza la tabla con los operadores matemáticos.

Métodos

- **private void initComponents()**: Método autogenerado por el Form Editor para inicializar los componentes de la interfaz gráfica. Configura el JComboBox (filtroOperadoresMatematicos) y la JTable (tablaOperadoresMatematicos) dentro de un JScrollPane. También define la disposición de estos componentes en el panel.
- **private void filtroOperadoresMatematicosItemStateChanged(java.awt.event.ItemEvent evt)**: Maneja los cambios en la selección del JComboBox. Aplica un filtro a la tabla basado en la selección actual. Si se selecciona "TODOS", se eliminan los filtros.
- **private void llenarComboBox()**: Llena el JComboBox con las opciones para filtrar la tabla: "TODOS", "SUMA", "RESTA", "MULTIPLICACIÓN", y "DIVISIÓN".
- **public void actualizarTablaOperadoresMatematicos()**: Actualiza la tabla tablaOperadoresMatematicos con los datos de los operadores matemáticos. Configura la tabla para que las celdas estén centradas y establece un TableRowSorter para permitir el filtrado de las filas basado en el JComboBox. Llena la tabla con los datos de la lista de operadores matemáticos del Lexer.

Variables de Instancia

- **private javax.swing.JComboBox<String> filtroOperadoresMatematicos**: Componente desplegable que permite al usuario seleccionar un tipo de operador matemático para filtrar en la tabla.
- **private javax.swing.JScrollPane jScrollPane1**: Componente que proporciona una vista desplazable para la tabla tablaOperadoresMatematicos.
- **private javax.swing.JTable tablaOperadoresMatematicos**: Tabla que muestra los datos de los operadores matemáticos.
- **private TableRowSorter<DefaultTableModel> sorter**: TableRowSorter utilizado para permitir el filtrado de filas en la tabla según la selección del JComboBox.

Clase ReporteErrores

La clase ReporteErrores extiende javax.swing.JPanel y proporciona una interfaz gráfica para mostrar un reporte de errores en una tabla. Utiliza una JTable y un DefaultTableModel para presentar los datos de los errores, y permite centrar el texto en las celdas.

Constructores

- **ReporteErrores()**: Inicializa el panel y actualiza la tabla con los datos de errores al momento de la creación del objeto.

Métodos

- **private void initComponents()**: Método autogenerado por el Form Editor para inicializar los componentes de la interfaz gráfica. Configura el JComboBox (filtroOperadoresMatematicos) y la JTable (tablaErrores) dentro de un JScrollPane. Define la disposición de estos componentes en el panel.
- **public void actualizarTablaErrores()**: Actualiza la tabla tablaErrores con los datos de los errores. Configura la tabla para que las celdas estén centradas y llena la tabla con los datos de la lista de errores obtenida desde el Lexer.

Variables de Instancia

- **private javax.swing.JComboBox<String> filtroOperadoresMatematicos**: Componente desplegable que en la implementación actual no está en uso, pero puede ser utilizado para filtrar los errores según el tipo en el futuro.
- **private javax.swing.JScrollPane jScrollPane1**: Componente que proporciona una vista desplazable para la tabla tablaErrores.
- **private javax.swing.JTable tablaErrores**: Tabla que muestra los datos de los errores.
- **DefaultTableModel modelo**: Modelo de datos de la tabla que se utiliza para manejar las filas y columnas de la JTable.

Notas Adicionales

- **Centrado de la Tabla**: Las celdas de la tabla se centran para una presentación más uniforme de los datos.
- **Actualización de la Tabla**: La tabla se llena con los datos de errores desde la lista Lexer.errores, que se espera que contenga instancias de ErrorReporte con información sobre los errores.

Paquete main/flex

Este paquete contiene el archivo de configuración para el analizador léxico.

lexer.flex

Directivas Iniciales

- %public: La clase generada por JFlex será pública.
- %class Lexer: Define el nombre de la clase generada.
- %unicode: Permite la entrada de caracteres Unicode.
- %line y %column: Incluye la información de línea y columna en los tokens generados.
- %ignorecase: Hace que la entrada no distinga entre mayúsculas y minúsculas.
- %cup: Indica que se generará un archivo compatible con CUP.

Definición de Tokens

- ID = [a-zA-Z_][a-zA-Z0-9_]*[a-zA-Z0-9-]*: Define una expresión regular para identificar identificadores.

Código Java Dentro de %{ %}

- public static ArrayList<ErrorReporte> errores = new ArrayList<>();: Lista estática para almacenar errores léxicos.
- public static ArrayList<Token> tokens = new ArrayList<>();: Lista estática para almacenar tokens generados.
- public static ArrayList<OperadorAritmetico> operadorAritmetico = new ArrayList<>();: Lista estática para almacenar operadores aritméticos.
- public ArrayList<ErrorReporte> getErrores() { return errores; }: Método para obtener los errores.

Definición de Estados y Reglas Léxicas

- **Palabras Clave y Símbolos**: La sección principal define los tokens y palabras clave del lenguaje. Cada expresión regular está asociada a un símbolo específico que se devuelve al analizador (por ejemplo, return new Symbol(ParserSym.AZUL, yyline+1, yycolumn+1, yytext());).
- **Espacios en Blanco**: Los espacios en blanco y saltos de línea se ignoran.
- **Operadores Aritméticos**: Los operadores aritméticos se almacenan en operadorAritmetico y se retornan como símbolos.
- **Identificadores y Números**: Los identificadores y números se reconocen y se retornan como símbolos.
- **Errores Léxicos**: Cualquier carácter desconocido se maneja como un error léxico y se agrega a la lista de errores.

Expresiones Regulares

Las expresiones regulares sirven para definir las reglas de un lenguaje, ya que estos indican como se crearán los tokens que luego serán enviados al analizador sintáctico.

Identificadores o nombres de objetos:
ID = [a-zA-Z_][a-zA-Z0-9_]*[a-zA-Z0-9-]*

Colores:
"azul" = azul
"rojo" = rojo
"amarillo" = amarillo
"verde" = verde
"negro" = negro
"celeste" = celeste
"rosado" = rosado
"morado" = morado

Palabras reservadas:
"línea" = línea
"curva" = curva

"graficar" = graficar
"animar" = animar
"circulo" = circulo
"cuadrado" = cuadrado
"rectangulo" = rectangulo
"linea" = linea
"poligono" = poligono
"objeto" = objeto
"anterior" anterior

[\t\n\r\f] = saltos de línea

Operadores aritméticos
"+" = suma
"-" = resta
"*" = multiplicacion
"/" = division
"(" = parentesis_abre
")" = parentesis_cierra
"," = coma

[0-9]+\.[0-9]+ = numero_decimal
[0-9]+ = numero_entero

Gramáticas Formales

Las gramáticas permiten validar si la entrada es correcta, en este caso los comandos para graficar y para animar, así como también para operaciones aritméticas. A continuación se presenta la gramática formal para esta práctica:

$$\sum=\{N, T, S, P\}$$

Donde:
N = Alfabeto de símbolos no terminales.
T = Alfabeto de símbolos terminales.
S = Símbolo inicial de la gramática, S pertenece a N.
P = Producciones.

N = {programa, instrucción, graficar, animar, forma, color, tipoanimacion, inicial, errorSintactico},

T = { PARENTESIS_ABRE, PARENTESIS_CIERRA, COMA, AZUL, ROJO, AMARILLO, VERDE, NEGRO, CELESTE, ROSADO, MORADO, LINEA, CURVA, GRAFICAR, CIRCULO, CUADRADO, RECTANGULO, LINEA_OBJ, POLIGONO, ANIMAR, OBJETO, ANTERIOR, NUMERO, ID, SUMA, RESTA, MULTIPLICACION, DIVISION},

S = {inicial}

P = {

inicial -> programa

programa -> instruccion programa
| error:err
| ID
| Ɛ

instruccion -> GRAFICAR forma
| ANIMAR OBJETO ANTERIOR animar;

forma -> CIRCULO PARENTESIS_ABRE ID COMA expresion COMA expresion COMA expresion COMA color PARENTESIS_CIERRA

| CUADRADO PARENTESIS_ABRE ID COMA expresion COMA expresion COMA expresion COMA color PARENTESIS_CIERRA

| LINEA_OBJ PARENTESIS_ABRE ID COMA expresion COMA expresion COMA expresion COMA expresion COMA color PARENTESIS_CIERRA

| RECTANGULO PARENTESIS_ABRE ID COMA expresion COMA expresion COMA expresion COMA expresion COMA color PARENTESIS_CIERRA

| POLIGONO PARENTESIS_ABRE ID COMA expresion COMA expresion COMA expresion COMA expresion COMA expresion COMA color PARENTESIS_CIERRA

| error

color -> AZUL
| ROJO
| AMARILLO
| VERDE
| NEGRO
| CELESTE
| ROSADO
| MORADO

animar -> PARENTESIS_ABRE tipoanimacion COMA expresion COMA expresion COMA expresion PARENTESIS_CIERRA

tipoanimacion -> CURVA
| LINEA
| error

expresion -> expresion SUMA expresion
| expresion RESTA expresion
| expresion MULTIPLICACION expresion
| expresion DIVISION expresion
| PARENTESIS_ABRE expresion PARENTESIS_CIERRA
| NUMERO:n
}