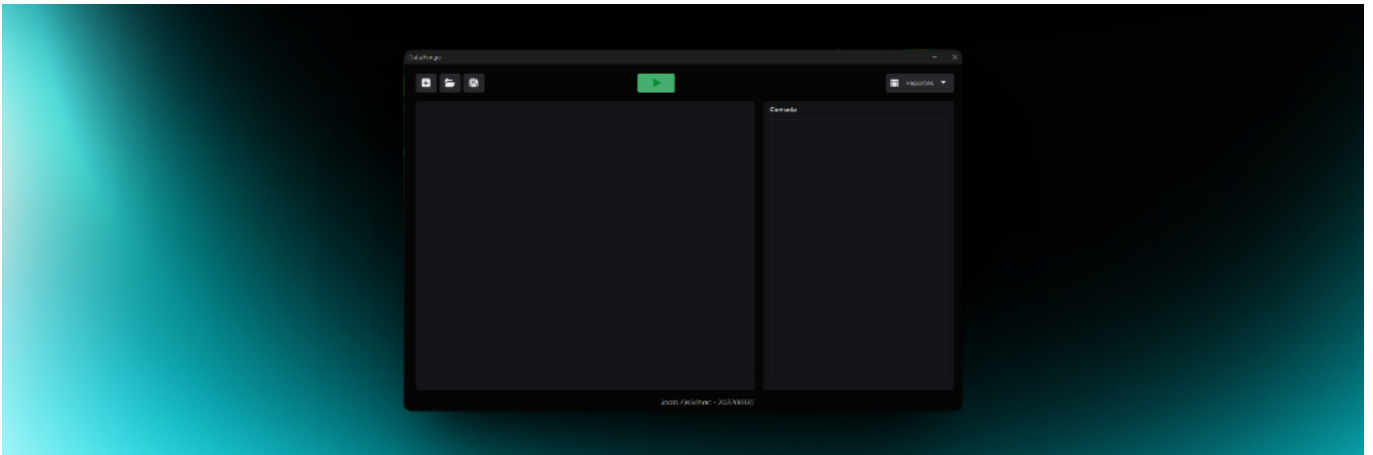







# Proyecto 1

---



 Joab Israel Ajsivinac Ajsivinac  202200135  
 Organización de Lenguajes y Compiladores 1  
 Universidad San Carlos de Guatemala  
 31 Primer Semestre 2024

## Manual Técnico

---

### Tecnologías Utilizadas



- Java
- Librerías de Java
  - FlatLaf 3.0
  - flatlaf-intellij-themes-3.0
  - jflex-full 1.7.0
  - java-cup 11b
  - java-cup 11b-runtime
- Visual Studio Code
- Git
- Html
- NetBeans
- tailwindcss

### Como funciona

La aplicación consta de una ventana principal y una secundaria donde se visualizan las gráficas

## Principal

La ventana principal tiene la tarea de manejar las pestañas, la apertura de archivos, el guardado de archivos, la creación de reportes, y la tarea más importante la de analizar el código.

**Manejo de pestañas** Para esto se hace uso de un arraylist, y un JTabbedPane, el primero es para guardar las rutas absolutas de los archivos, y la segunda es para crear el elemento visual de las pestañas, esto se hace mediante el siguiente código

```
tabbedPane.setTabComponentAt(index, tabComponent);

if (index >= rutas.size()) {
    for (int i = rutas.size(); i < index; i++) {
        rutas.add(null);
    }
    rutas.add(index, selectedFile.getAbsolutePath());
} else {
    rutas.add(index, selectedFile.getAbsolutePath());
}
```

**Guardado de archivos** El programa puede guardar archivos con extensión .df, esto lo hace realizando primero una validación de que exista una pestaña seleccionada, para luego crear un archivo en la ruta indicada, finalmente agregando la nueva ruta al arraylist de rutas.

**Creación de reportes** Para la creación primero se crea una instancia de la ventana Charts.

```
Charts c=new Charts();
```

luego cuando en el listado de instrucciones encuentra una declaración de una gráfica, se procede a tomar que tipo de gráfica es necesitada.

```
if (graphType == TypeVariableG.BARRAS) {
    createBarChart(temp);
} else if (graphType == TypeVariableG.PIE) {
    createPieChart(temp);
} else if (graphType == TypeVariableG.LINEA) {
    createLineChart(temp);
} else if (graphType == TypeVariableG.HISTOGRAMA) {
    createHistogram(temp);
}
```

una vez seleccionado el tipo de gráfica que se va a mostrar, se procede a llamar a los métodos respectivos de la ventana Charts.

- addPieChart

```
c.addPieChart(val, labels, titulo.replaceAll("\\\"", ""));
```

- addBarChart

```
c.addBarChart(titulo.replaceAll("\\\"", ""), tituloy.replaceAll("\\\"", ""),
titulox.replaceAll("\\\"", ""), ejey, ejex);
```

- addLineChart

```
c.addLineChart(titulo.replaceAll("\\\"", ""), tituloy.replaceAll("\\\"", ""),
titulox.replaceAll("\\\"", ""), ejey, ejex);
```

En el caso del histograma se usa `addBarChart`, ya que tienen el mismo comportamiento, pero sin algunas etiquetas

## Analisis

Para analizar el código se hace uso de `jflex` y `cup`, el primero se encarga de analizar lexicamente el código y el segundo se encarga de analizar sintacticamente los tokens devueltos por `jflex`.

### Análisis Léxico

Dentro del análisis léxico se definieron los siguientes tokens

Descripción	Patrón	Expresión regular	Ejemplo	Nombre del Token
Reservada program	Palabra program	Program	Program	TK_program
Reservada end	Palabra end	End	End	TK_end
Reservada double	Palabra double	Double	Double	TK_double
Reservada char	Palabra char	Char	Char	TK_char
Reservada var	Palabra var	Var	Var	TK_var
Reservada arr	Palabra arr	Arr	Arr	TK_arr
Reservada sum	Palabra sum	Sum	Sum	TK_sum
Reservada res	Palabra res	Res	Res	TK_res
Reservada mul	Palabra mul	Mul	Mul	TK_mul
Reservada div	Palabra div	Div	Div	TK_div

Descripción	Patrón	Expresión regular	Ejemplo	Nombre del Token
Reservada mod	Palabra mod	Mod	Mod	TK_mod
Reservada media	Palabra media	Media	Media	TK_media
Reservada mediana	Palabra mediana	Mediana	Mediana	TK_mediana
Reservada moda	Palabra moda	Moda	Moda	TK_moda
Reservada varianza	Palabra varianza	Varianza	Varianza	TK_varianza
Reservada max	Palabra max	Max	Max	TK_max
Reservada min	Palabra min	Min	Min	TK_min
Reservada print	Palabra print	Print	Print	TK_print
Reservada column	Palabra column	Columna	Columna	TK_column
Reservada console	Palabra console	Console	Console	TK_console
Reservada graphbar	Palabra graphbar	Graphbar	Graphbar	TK_graphbar
Reservada ejex	Palabra ejex	Ejex	Ejex	TK_ejex
Reservada ejey	Palabra ejey	Ejey	Ejey	TK_ejey
Reservada titulox	Palabra titulox	Titulox	Titulox	TK_titulox
Reservada tituloy	Palabra tituloy	Tituloy	Tituloy	TK_tituloy
Reservada titulo	Palabra titulo	Titulo	Titulo	TK_titulo
Reservada exec	Palabra exec	Exec	Exec	TK_exec
Reservada graphline	Palabra graphline	graphline	graphline	TK_graphline
Reservada graphpie	Palabra graphpie	Graphpie	Graphpie	TK_graphpie
Reservada label	Palabra label	Label	Label	TK_label

Descripción	Patrón	Expresión regular	Ejemplo	Nombre del Token
Reservada values	Palabra values	Values	Values	TK_values
Reservada histogram	Palabra histogram	Histogram	Histogram	TK_histogram
Dos puntos	Palabra program	:	:	TK_colon
Corchete de apertura	Palabra program	[	[	TK_lbracket
Corchete de cerradura	Palabra program	]	]	TK_rbracket
Igual	carácter =	=	=	TK_equal
Guión	carácter -	-	-	TK_minus
Menor que	carácter <	<	<	TK_lt
Mayor que	carácter >	>	>	TK_gt
Punto y coma	carácter ;	;	;	TK_semicolon
Coma	carácter ,	,	,	TK_comma
Paréntesis de apertura	carácter (	(	(	TK_lparen
Paréntesis de cerradura	carácter )	)	)	TK_rparen
Identificadores	Secuencia que inicia con un carácter alfanumérico o guion seguido de numeros o letras	( <code>[a-zA-Z][a-zA-Z0-9_]</code> )	variable	TK_id
Cadenas de texto	Secuencia de caracteres encerrados entre comillas	( <code>"[^"\\n]+"</code> )	"cadena"	TK_string
Números enteros y decimales	Secuencia de numeros que pueden tener punto decimal	( <code>[0-9]+(.[0-9])*</code> )	2, 2.2, 3.333	TK_double_v

## Análisis Sintáctico

El analizador sintáctico sigue la siguiente gramática [Gramática](#)

## Funciones Aritmeticas

Para la realización de operaciones se hace uso de lo siguiente:

```

private double evaluateArith(ArithmeticExp data) {
    double resultado = 0.0f;
    VariableValue v1 = (VariableValue) data.getV1();
    VariableValue v2 = (VariableValue) data.getV2();
    String op = data.getOp();
    double operando1, operando2;
    if (v1.getType() == TypeVariable.AR) {
        ArithmeticExp temp = (ArithmeticExp) v1.getValue();
        operando1 = evaluateArith(temp);
    } else if (v1.getType() == TypeVariable.ID) {
        String temp = (String) v1.getValue();
        Information info = (Information) table.get(temp);
        Object resp = info.getValue();
        operando1 = (double) resp;
    } else if (v1.getType() == TypeVariable.ST) {
        StatisticalExp e = (StatisticalExp) v1.getValue();
        operando1 = evaluateStats(e);
    } else {
        operando1 = (double) v1.getValue();
    }
    if (v2.getType() == TypeVariable.AR) {
        ArithmeticExp temp = (ArithmeticExp) v2.getValue();
        operando2 = evaluateArith(temp);
    } else if (v2.getType() == TypeVariable.ID) {
        String temp = (String) v2.getValue();
        Information info = (Information) table.get(temp);
        Object resp = info.getValue();
        operando2 = 0;
        operando2 = (double) resp;
    } else if (v2.getType() == TypeVariable.ST) {
        StatisticalExp e = (StatisticalExp) v2.getValue();
        operando2 = evaluateStats(e);
    } else {
        operando2 = (double) v2.getValue();
    }

    switch (op) {
        case "/":
            resultado = operando1 / operando2;
            break;
        case "*":
            resultado = operando1 * operando2;
            break;
        case "%":
            resultado = operando1 % operando2;
            break;
        case "+":
            resultado = operando1 + operando2;
            break;
        case "-":
            resultado = operando1 - operando2;
            break;
        default:
    }
}

```

```

        resultado = 0;
    }

    return resultado;
}

```

Lo primero que se realiza es buscar los valores de los operandos, ya que pueden ser, otras operaciones, números, expresiones estadísticas, o variables, en el caso de encontrar otra expresión aritmética entra en una recursión. Luego con el switch case se procede a verificar qué operación se debe realizar, y eso lo guarda en la variable resultado.

Operaciones disponibles:

- Suma
- Resta
- Multiplicación
- División
- Módulo

## Funciones Estadísticas

Para la realización de funciones estadísticas es importante conocer que solo permite arreglos como parámetros y se usa la siguiente función:

```

private double evaluateStats(StatisticalExp data) {
    String op = data.getType_s();
    ArrayList<Double> arrayListDeDoubles = new ArrayList<>();

    if (data.getValues() instanceof String) {
        String temp = (String) data.getValues();
        Information info = (Information) table.get(temp);
        arrayListDeDoubles = (ArrayList<Double>) info.getValue();
    } else {
        VariableValue v = (VariableValue) data.getValues();
        ArrayList<VariableValue> array = (ArrayList<VariableValue>)
v.getValue();

        for (VariableValue variableValue : array) {
            if (variableValue.getType() == TypeVariable.DOUBLE) {
                arrayListDeDoubles.add((Double) variableValue.getValue());
            } else if (variableValue.getType() == TypeVariable.AR) {
                ArithmeticExp temp = (ArithmeticExp) variableValue.getValue();
                arrayListDeDoubles.add(evaluateArith(temp));
            } else if (variableValue.getType() == TypeVariable.ID) {
                String id = (String) variableValue.getValue();
                Information info = (Information) table.get(id);
                Object resp = info.getValue();
                if (resp instanceof Double) {
                    arrayListDeDoubles.add((double) resp);
                }
            }
        }
    }
}

```

```

    }
}

return operate(arrayListDeDoubles, op);
}

```

En este caso se crea un array de tipo double donde se guardan los valores recibidos, para luego validar el tipo de variable con la que se esta trabajando, para luego llamar a operate y recibir el valor del resultado. La función operate es la siguiente:

```

public double operate(ArrayList<Double> list, String op) {
    double resultado = 0.0f;
    switch (op) {
        case "Media":
            resultado = Statistics.Mean(list);
            break;
        case "Mediana":
            resultado = Statistics.Median(list);
            break;
        case "Moda":
            resultado = Statistics.Mode(list);
            break;
        case "Varianza":
            resultado = Statistics.Variance(list);
            break;
        case "Max":
            resultado = Statistics.Maximum(list);
            break;
        case "Min":
            resultado = Statistics.Minimum(list);
            break;
        default:
            throw new AssertionError();
    }
    return resultado;
}

```

Esta función se encarga de llamar a las funciones correctas de la clase Statistics, la cual calcula las diferentes operaciones disponibles.

Operaciones Disponibles:

- Media
- Mediana
- Moda
- Varianza
- Max
- Min