PRACTICA NUMERO 1

28.2.2021

Carlos Benjamín Pac Flores
Centro Universitario de Occidente (CUNOC)
Ingeniería en Ciencias y Sistemas
Febrero 2022

Introducción

El siguiente documento da presenta una descripción de la estructura del software de la aplicación desarrollada para el entorno de Android, que por medio de una serie de un lenguaje formal el software es capaz de graficar los gráficos de Barras y de Pie respectivamente, el procesado del código es por medio de un analizador lexico y sintactico y por medio de este último poder resolver parámetros basados en operaciones entre números tales como sumas, restas, multiplicaciones y divisiones y agrupaciones.

Plataforma de Desarrollo:

- Sistema de gestión de proyecto Grandle.
- Android Studio
- Librería para análisis léxico JFLEX
- Librería para análisis sintáctico CUP
- Lenguaje Kotlin y Java

Alcances Tecnicos:

Poder visualizar Gráficas de Pie o de Barras por medio de un lenguaje formal y con ayuda de la librería MP Android Chart en su versión 3.1.0 ayuda a visualizar las gráficas ya que ofrece una configuración personalizable, y ser amigable con el usuario.

Requisitos De Software:

- Android en su version minima 4.2
- Ram recomendada 3 GB

Detalle del Sistema:

Estructura del analisis Lexico:

Para el funcionamiento del analizador léxico se establecen las siguientes expresiones regulares para el reconocimiento de los TOKENS del texto.

- **LineTerminator = [\r|\n|\r\n]+**

  Reconoce los fines de línea en el texto, como retornos de carro y nuevas líneas.

- **WhiteSpace = [\t\n]+**

  Reconoce los espacios entre textos, como los ya mencionados espacios y las tabulaciones.

- **Comment = [#][^\n]***

  Reconoce los comentarios incluidos en el código y su alcance máximo es una línea

- **Def = def|Def**

  Reconoce la instrucción para la definición de una gráfica.

- **GraphicBarras = Barras**

  Reconoce la instrucción para la definición de una gráfica de Barras.

- **GraphicPie = Pie**

  Reconoce la instrucción para la definición de una gráfica de Pie.

- **Tittle = titulo**

  Reconoce el parámetro título de una gráfica tanto de barras o de pie.

- **EjeX = ejex**

  Reconoce el parámetro ejex de una gráfica de barras.

- **EjeY = ejey**

  Reconoce el parámetro ejey de una gráfica de barras.

- **Label = etiquetas**

  Reconoce el parámetro etiqueta de una gráfica de pie.

- **Values = valores**

Reconoce el parámetro valores de una gráfica de pie.

- **Link = unir**

  Reconoce el parámetro unir de una gráfica tanto de barras o de pie.

- **Type = tipo**

  Reconoce el parámetro tipo de una gráfica de pie.

- **TypeValue = Porcentaje|Cantidad**

  Reconoce el valor del parámetro tipo de una gráfica de pie.

- **Total = total**

  Reconoce el parámetro total de una gráfica de pie.

- **Extra = extra**

  Reconoce el parámetro extra de una gráfica de pie.

- **Execute = Ejecutar**

  Reconoce la instrucción ejecutar del lenguaje.

- **Numbers = [0-9]+**

  Reconoce un valor entero escrito en el lenguaje.

- **Decimal = [0-9]+[.][0-9]+**

  Reconoce un valor decimal escrito en el lenguaje.

- **[\"][^\n\r\"]+[\"]**

  Reconoce una cadena de datos escrita en el lenguaje.

- **{ + }**

  Reconoce el símbolo de suma.

- **{ - }**

  Reconoce el símbolo de resta.

- **{ * }**

  Reconoce el símbolo de multiplicación.

- **{ / }**

  Reconoce el símbolo de división.

- **{ (}**

  Reconoce el paréntesis de apertura.

- **{) }**

  Reconoce el paréntesis de cierre.

- **{ { }**

  Reconoce la llave de apertura.

- **{ } }**

  Reconoce la llave de cierre.

- **{[\[]}**

  Reconoce el corchete de apertura.

- **{ [\]] }**

  Reconoce el corchete de cierre.

- **{ , }**

  Reconoce el separador de texto denominado COMA.

- **{ ; }**

  Reconoce el fin de una instrucción denominada PUNTO Y COMA.

- **{ : }**

  Reconoce la asignación de un parámetro de una gráfica.

Estructura del Análisis Sintáctico:

Símbolos Terminales:

DEF, GRAPHICBARRA, GRAPHICPIE, TITTLE, EJEX, EJEY, LABEL, VALUES, LINK, TYPE, TYPEVALUE, TOTAL, EXTRA, EXECUTE, MAS, MENOS, MUL, DIV, PA_A, PA_C, CO_A, CO_C, LLA_A, LLA_C, COMA, DOSPUNTOS, PUNTOCOMA, STRING, NUMBERS, DECIMAL

Símbolos No Terminales:

s , ej, EjeGraf, DefgrafPie, DefgrafBarra, contPie, contBarra, contEjeY, contEjeY2, contEjeX, contEjeX2, contUnir, contUnir2, e ,ee, t, f, te, fe

Reglas de Producción:

El símbolo no terminal “ej” es donde se inicia la producción de gramática.

Con esta producción se reconoce por medio de recursividad por la izquierda la instrucciones de ejecutar o ninguna de ellas, con el fin de que al momento que sea reconocida se puedan realizar operaciones sobre las mismas.

ej ::= ej EjeGraf

| s

Con esta producción se reconoce por medio de recursividad por la izquierda la definiciones de gráficas, de igual manera que las ejecuciones se utiliza la recursividad por la izquierda para poder evaluar los resultados de una manera más fácil y en orden de tipado.

s ::= s DefgrafBarra

| s DefgrafPie

| DefgrafBarra

| DefgrafPie

Con las siguientes 3 producciones podemos reconocer la estructura principal para la definición de un gráfico.

EjeGraf ::= EXECUTE PA_A STRING PA_C PUNTOCOMA

DefgrafBarra ::= DEF GRAPHICBARRA contBarra LLA_C

DefgrafPie ::= DEF GRAPHICPIE contPie LLA_C

Con las producciones e,t,f son utilizadas para reconocer y realizar operaciones tales como la suma, resta,multiplicación, división, en números enteros o decimales.

e ::= e MAS t

| e MENOS t

| t

t ::= t MUL f

| t DIV f

| f

f ::= NUMBERS

| DECIMAL

| PA_A e PA_C

Con estas dos producciones podemos recuperar los valores numéricos pertenecientes al arreglo de asignación a la propiedad ejey de la gráfica de barras o valores para la gráfica de pie.

contEjeY ::= CO_A e contEjeY2

contEjeY2 ::= COMA e contEjeY2

| CO_C

Con estas dos producciones podemos recuperar los valores numéricos pertenecientes al arreglo de asignación a la propiedad eje de la gráfica de barras o etiquetas para la gráfica de pie.

contEjeX ::= CO_A STRING contEjeX2

contEjeX2 ::= COMA STRING contEjeX2

| CO_C

Con estas dos producciones podemos recuperar la pareja de valores numéricos enteros utilizados para hacer una referencia de utilización para las propiedades ejex y ejey del gráfico de barras y valores y etiquetas para la gráfica de pie.

contUnir ::= LLA_A ee COMA ee LLA_C contUnir2

contUnir2 ::= COMA LLA_A ee COMA ee LLA_C contUnir2

| CO_C

Con las últimas 3 producciones ee,et,fe son utilizadas para reconocer y realizar operaciones tales como la suma, resta, multiplicación, división, en números enteros de exclusividad para el parámetro unir de las gráficas.

ee ::= ee MAS te

| ee MENOS te

| te

te ::= te MUL fe

| te DIV fe

| fe

fe ::= NUMBERS

| PA_A e PA_C

Validación de los valores específicos (análisis semántico):

El lenguaje al ser restringido por ciertos valores que pueden o no estar se utiliza lógica por medio de clases no ajenas al analisis lexico y sintactico en lenguaje java ; como el verificar que los valores introducidos en los parámetros de unir sean enteros y que estén en el rango de acceso a los mismos ya que hacen referencia a un casilla de un array de datos,

otro a mencionar es los parámetros de configuración de un gráfica en el tipo de gráfica de pie ya que dependiendo de su valor usamos una parte de los parámetros permitidos y otro no lo son.

Código de Ejemplo:

```
#Inicio del archivo de ejemplo
def Barras{
    titulo: "Grafica 1";
    ejex:["item1", "item2"];
    ejey:[5 + (3*3),1+34-2];
    unir:[{0,1}, {1,0}];
    #Fin de definicion de grafica
}
Def Pie{
    titulo: "Grafica 2";
    tipo: Cantidad;
    etiquetas: ["Compi1", "Compi2"];
    valores:[5, 10];
    total: 25;
    unir:[{0,1}, {1,0}];
    extra: "Resto";
}
Def Pie{
    titulo: "Grafica 3";
    tipo: Porcentaje;
    etiquetas: ["Compi1","Compi2"];
    valores:[70,120];
    unir:[{0,1},{1,0}];
    extra: "Resto";
}
```

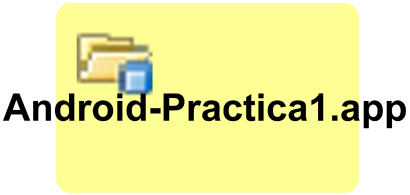
Organización del Proyecto:

Por medio del **MainActivity** y de los componentes de vista recuperamos del contenedor de texto ya una vez recuperado el texto escrito es enviado al **ProsesadorGrafico** una clase que une todas la herramientas necesarias para realizar el graficado inicializado el analizador **Léxico(Lexer)** al cual se le agrega el Reader del texto que se extrajo anteriormente y que a su vez es enviado al **Analizador Sintactico(ParserGraphic)**, ya una vez en el parser por medio de la gramática anteriormente descrita se extraen los datos y se validan y se encapsulan en objetos de **GráficaBarras**, **GráficasPie** y **Gráficas** en objetos para que más adelante en la clase en la clase de **AgregarGraficas** solamente se extraiga la información y se grafique por medio de la librería para crear gráficos MPA Android Chart, igualmente los errores son encapsulados en la clase **ErroresAnalisis** donde se guarda el tipo de error, el error, su ubicación y la descripción del mismo para que sea mostrado de manera tabular más adelante y sea corregido por el usuario.

Ya una vez procesada toda la información la forma de visualizar todo depende en la cantidad de errores que hayan en el análisis si la cantidad es 0 se ejecutarán todos los gráficos en una nueva ventana de la aplicación y también se podrá acceder a las ocurrencias y cantidades de gráficos que se definieron en los códigos en una ventana nuevamente dedicada para el resultado del codigo.

Estructura del Proyecto:

Modulos:



Vista de Paquetes:

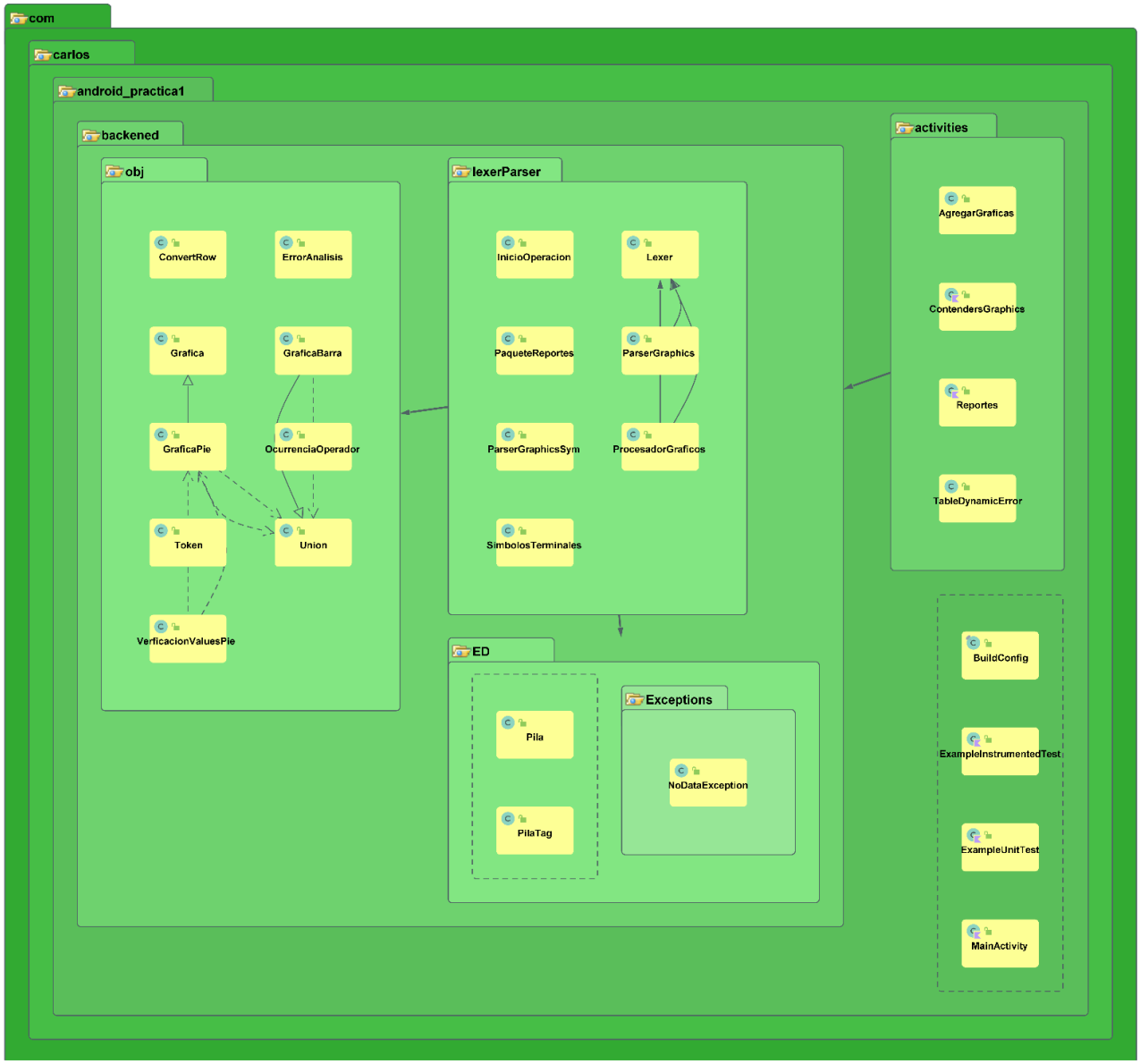
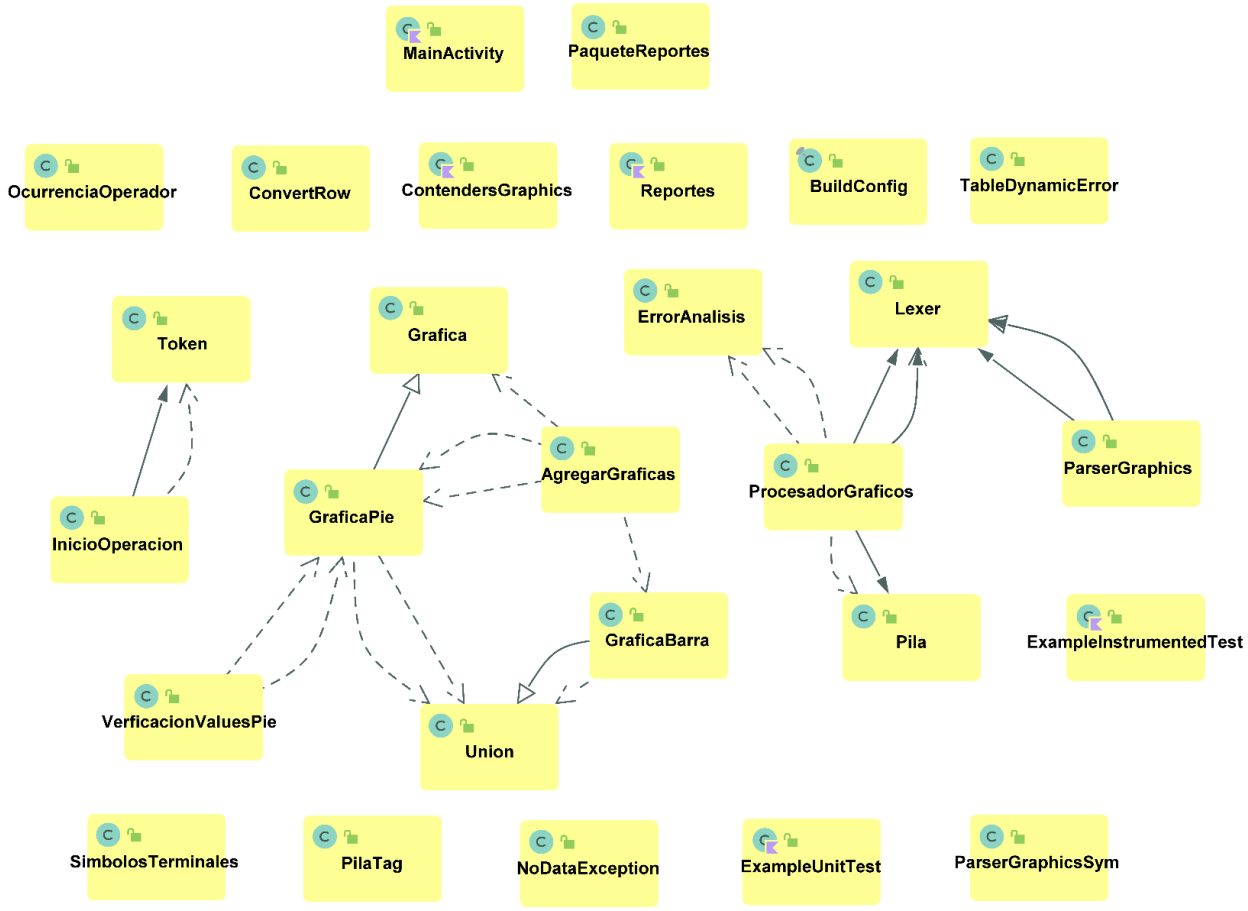
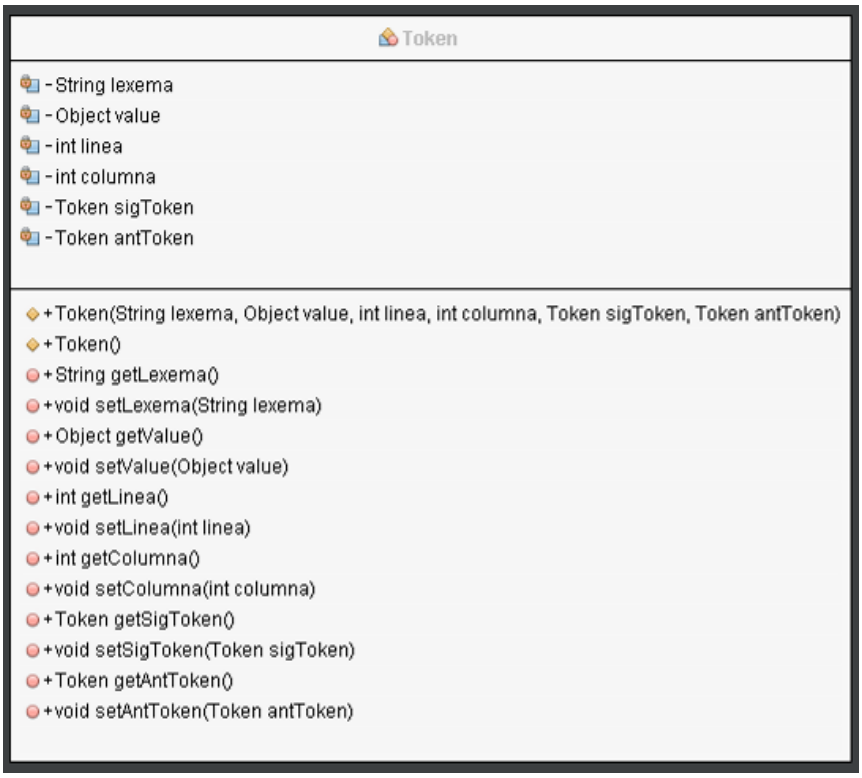


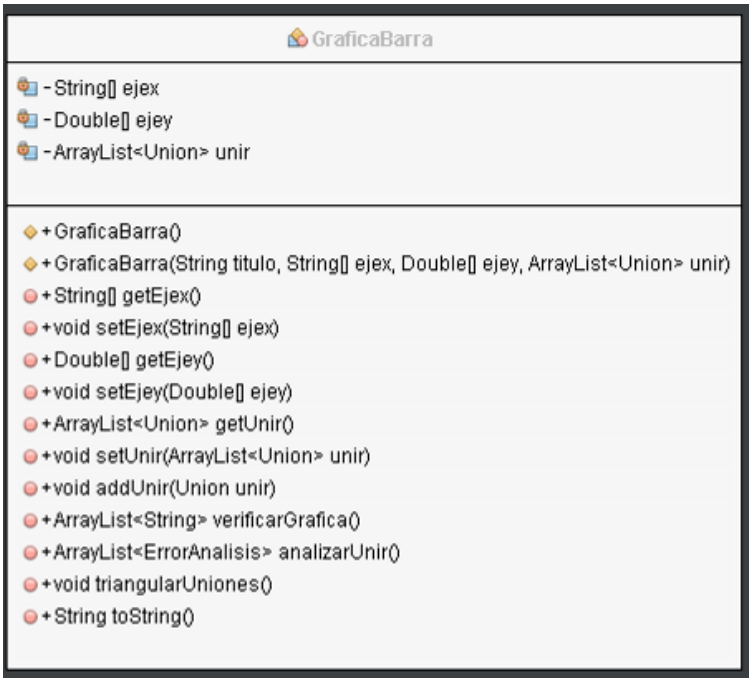
Diagrama de Clases:

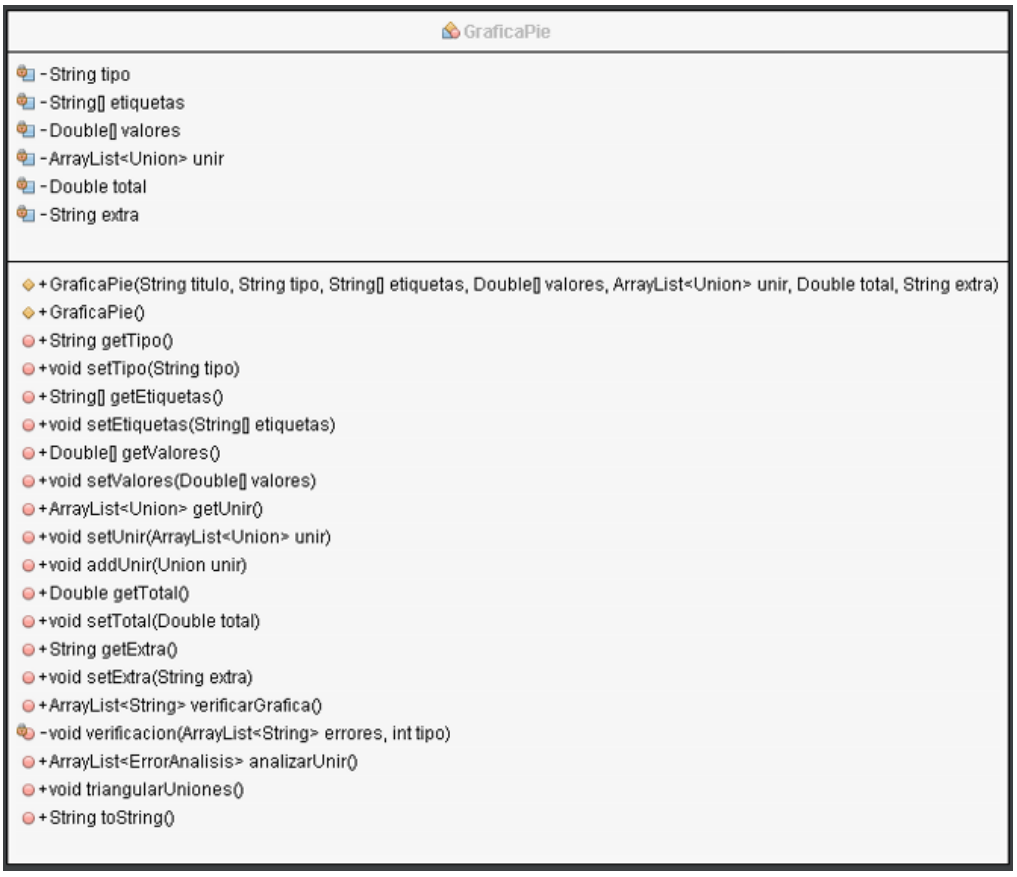
Es presentado de esta manera ya que su descripción general completa no permitirá visualizarlos de una manera correcta, por ende adelante de esta sección se adjunta su descripción completa.

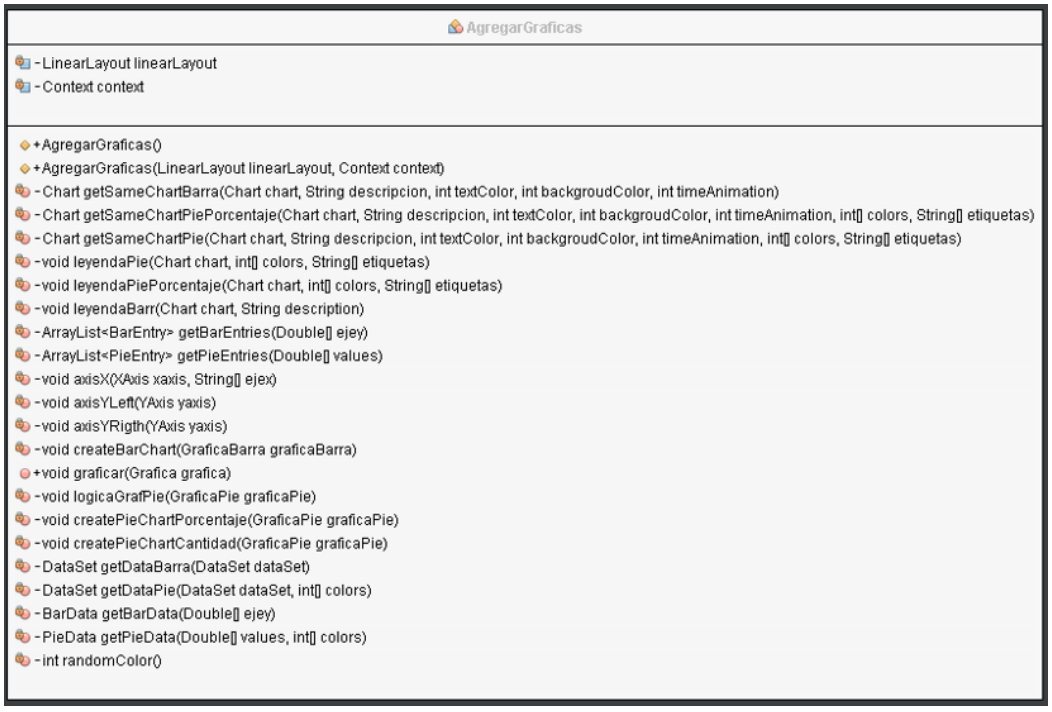


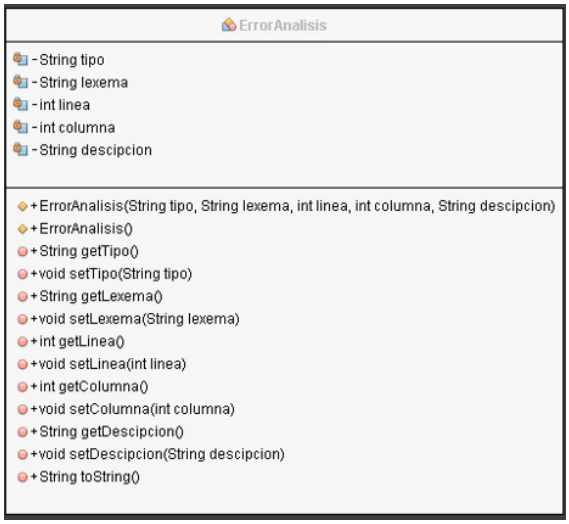
Contenido de las clases utilizadas:

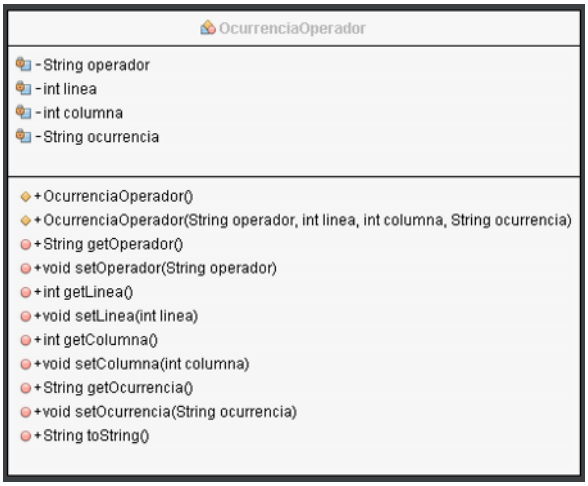

Token
-String lexema
-Object value
-int linea
-int columna
-Token sigToken
-Token antToken
+Token(String lexema, Object value, int linea, int columna, Token sigToken, Token antToken)
+Token()
+String getLexema()
+void setLexema(String lexema)
+Object getValue()
+void setValue(Object value)
+int getLinea()
+void setLinea(int linea)
+int getColumna()
+void setColumna(int columna)
+Token getSigToken()
+void setSigToken(Token sigToken)
+Token getAntToken()
+void setAntToken(Token antToken)

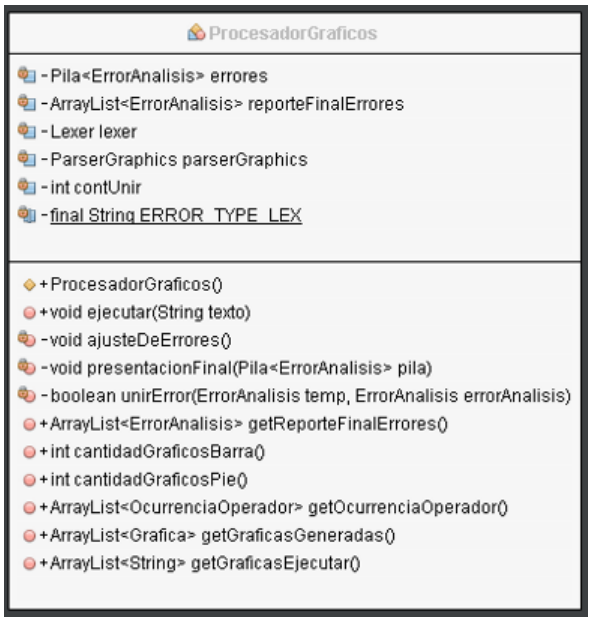

GraficaBarra
-String[] ejex
-Double[] ejey
-ArrayList<Union> unir
+GraficaBarra()
+GraficaBarra(String titulo, String[] ejex, Double[] ejey, ArrayList<Union> unir)
+String[] getEjex()
+void setEjex(String[] ejex)
+Double[] getEjey()
+void setEjey(Double[] ejey)
+ArrayList<Union> getUnir()
+void setUnir(ArrayList<Union> unir)
+void addUnir(Union unir)
+ArrayList<String> verificarGrafica()
+ArrayList<ErrorAnalisis> analizarUnir()
+void triangularUniones()
+String toString()

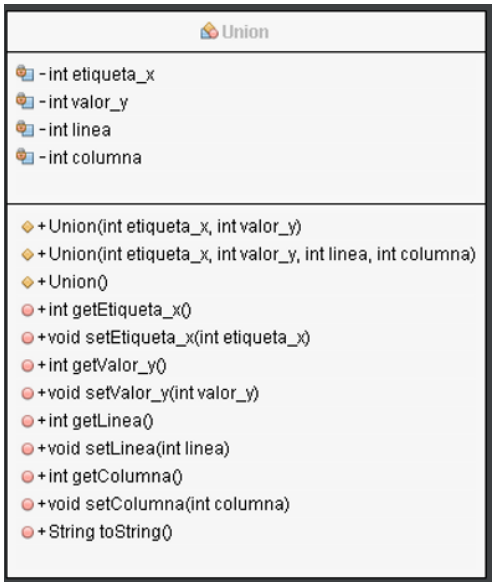

GraficaPie
-String tipo
-String[] etiquetas
-Double[] valores
-ArrayList<Union> unir
-Double total
-String extra
+GraficaPie(String titulo, String tipo, String[] etiquetas, Double[] valores, ArrayList<Union> unir, Double total, String extra)
+GraficaPie()
+String getTipo()
+void setTipo(String tipo)
+String[] getEtiquetas()
+void setEtiquetas(String[] etiquetas)
+Double[] getValores()
+void setValores(Double[] valores)
+ArrayList<Union> getUnir()
+void setUnir(ArrayList<Union> unir)
+void addUnir(Union unir)
+Double getTotal()
+void setTotal(Double total)
+String getExtra()
+void setExtra(String extra)
+ArrayList<String> verificarGrafica()
-void verificacion(ArrayList<String> errores, int tipo)
+ArrayList<ErrorAnalisis> analizarUnir()
+void triangularUniones()
+String toString()


AgregarGraficas
-LinearLayout linearLayout
-Context context
+AgregarGraficas()
+AgregarGraficas(LinearLayout linearLayout, Context context)
-Chart getSameChartBarra(Chart chart, String descripcion, int textColor, int backgroudColor, int timeAnimation)
-Chart getSameChartPiePorcentaje(Chart chart, String descripcion, int textColor, int backgroudColor, int timeAnimation, int[] colors, String[] etiquetas)
-Chart getSameChartPie(Chart chart, String descripcion, int textColor, int backgroudColor, int timeAnimation, int[] colors, String[] etiquetas)
-void leyendaPie(Chart chart, int[] colors, String[] etiquetas)
-void leyendaPiePorcentaje(Chart chart, int[] colors, String[] etiquetas)
-void leyendaBarr(Chart chart, String description)
-ArrayList<BarEntry> getBarEntries(Double[] ejey)
-ArrayList<PieEntry> getPieEntries(Double[] values)
-void axisX(XAxis xaxis, String[] ejex)
-void axisYLeft(YAxis yaxis)
-void axisYRigth(YAxis yaxis)
-void createBarChart(GraficaBarra graficaBarra)
+void graficar(Grafica grafica)
-void logicaGrafPie(GraficaPie graficaPie)
-void createPieChartPorcentaje(GraficaPie graficaPie)
-void createPieChartCantidad(GraficaPie graficaPie)
-DataSet getDataBarra(DataSet dataSet)
-DataSet getDataPie(DataSet dataSet, int[] colors)
-BarData getBarData(Double[] ejey)
-PieData getPieData(Double[] values, int[] colors)
-int randomColor()

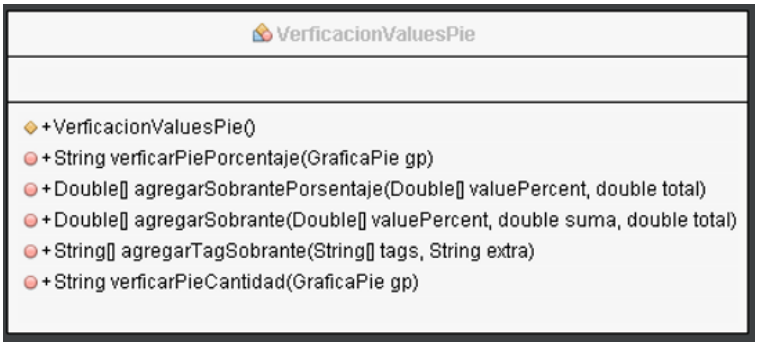

ErrorAnalisis
- String tipo
- String lexema
- int linea
- int columna
- String descipcion
+ ErrorAnalisis(String tipo, String lexema, int linea, int columna, String descipcion)
+ ErrorAnalisis()
+ String getTipo()
+ void setTipo(String tipo)
+ String getLexema()
+ void setLexema(String lexema)
+ int getLinea()
+ void setLinea(int linea)
+ int getColumna()
+ void setColumna(int columna)
+ String getDescipcion()
+ void setDescipcion(String descipcion)
+ String toString()

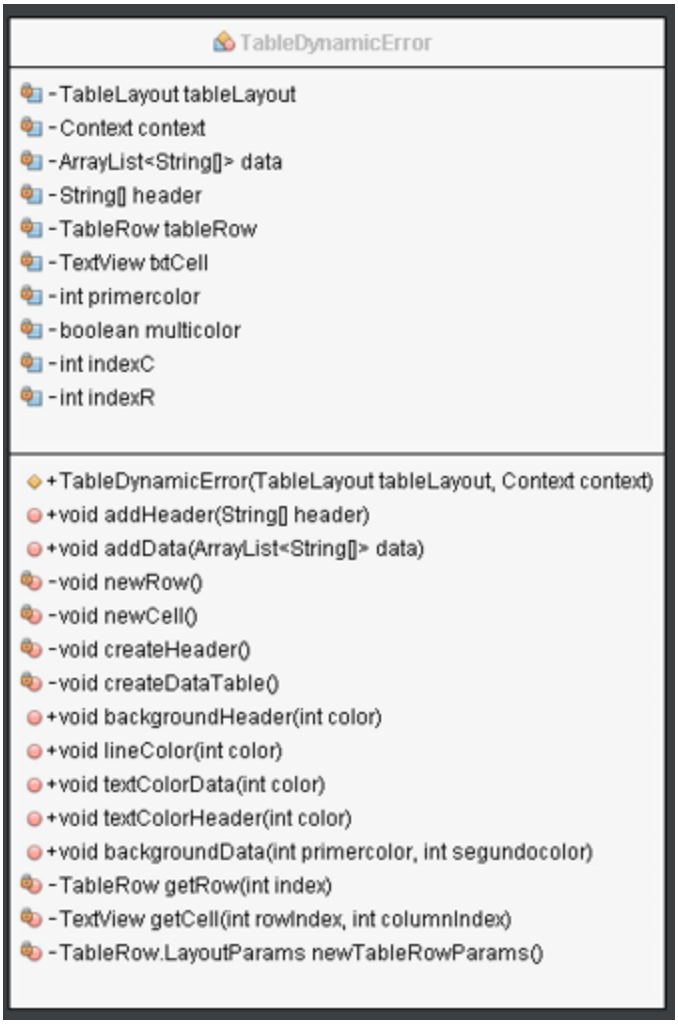

OcurrenciaOperador
- String operador
- int linea
- int columna
- String ocurrencia
+ OcurrenciaOperador()
+ OcurrenciaOperador(String operador, int linea, int columna, String ocurrencia)
+ String getOperador()
+ void setOperador(String operador)
+ int getLinea()
+ void setLinea(int linea)
+ int getColumna()
+ void setColumna(int columna)
+ String getOcurrencia()
+ void setOcurrencia(String ocurrencia)
+ String toString()


ProcesadorGraficos
- Pila<ErrorAnalisis> errores
- ArrayList<ErrorAnalisis> reporteFinalErrores
- Lexer lexer
- ParserGraphics parserGraphics
- int contUnir
- final String ERROR_TYPE_LEX
+ ProcesadorGraficos()
+ void ejecutar(String texto)
- void ajusteDeErrores()
- void presentacionFinal(Pila<ErrorAnalisis> pila)
- boolean unirError(ErrorAnalisis temp, ErrorAnalisis errorAnalisis)
+ ArrayList<ErrorAnalisis> getReporteFinalErrores()
+ int cantidadGraficosBarra()
+ int cantidadGraficosPie()
+ ArrayList<OcurrenciaOperador> getOcurrenciaOperador()
+ ArrayList<Grafica> getGraficasGeneradas()
+ ArrayList<String> getGraficasEjecutar()


Union
- int etiqueta_x
- int valor_y
- int linea
- int columna
+ Union(int etiqueta_x, int valor_y)
+ Union(int etiqueta_x, int valor_y, int linea, int columna)
+ Union()
+ int getEtiqueta_x()
+ void setEtiqueta_x(int etiqueta_x)
+ int getValor_y()
+ void setValor_y(int valor_y)
+ int getLinea()
+ void setLinea(int linea)
+ int getColumna()
+ void setColumna(int columna)
+ String toString()


SimbolosTerminales
+ final String[] terminalNames
+ SimbolosTerminales()
+ List<String> obtenerSimbolos(List<Integer> expected_tokens)


VerficacionValuesPie
+VerficacionValuesPie()
+String verficarPiePorcentaje(GraficaPie gp)
+Double[] agregarSobrantePorsentaje(Double[] valuePercent, double total)
+Double[] agregarSobrante(Double[] valuePercent, double suma, double total)
+String[] agregarTagSobrante(String[] tags, String extra)
+String verficarPieCantidad(GraficaPie gp)


TableDynamicError
-TableLayout tableLayout
-Context context
-ArrayList<String[]> data
-String[] header
-TableRow tableRow
-TextView txtCell
-int primercolor
-boolean multicolor
-int indexC
-int indexR
+TableDynamicError(TableLayout tableLayout, Context context)
+void addHeader(String[] header)
+void addData(ArrayList<String[]> data)
-void newRow()
-void newCell()
-void createHeader()
-void createDataTable()
+void backgroundHeader(int color)
+void lineColor(int color)
+void textColorData(int color)
+void textColorHeader(int color)
+void backgroundData(int primercolor, int segundocolor)
-TableRow getRow(int index)
-TextView getCell(int rowIndex, int columnIndex)
-TableRow.LayoutParams newTableRowParams()


PilaTag
-NodoPilaTag<T> raiz
-int size
+PilaTag()
+void push(T data, String tag)
+T pop()
+T peek()
+boolean buscar(String tag)
+T retornar(String tag)
+boolean isEmpty()
+int getSize()
+ArrayList<T> toArrayList()
+T[] toArray(Class<T[]> arrayType)
+void imprimirPila()


Grafica
-String titulo
-String[] tag
-Double[] val
-int linUnir
-int colUnir
+Grafica()
+Grafica(String titulo)
+String getTitulo()
+void setTitulo(String titulo)
+String[] getTag()
+void setTag(String[] tag)
+Double[] getVal()
+void setVal(Double[] val)
+int getLinUnir()
+void setLinUnir(int linUnir)
+int getColUnir()
+void setColUnir(int colUnir)
Pila
-NodoPila<T> raiz
-int size
+Pila()
+void push(T data)
+T pop()
+T peek()
+boolean isEmpty()
+int getSize()
+ArrayList<T> toArrayList()
+T[] toArray(Class<T[]> arrayType)
+void imprimirPila()
InicioOperacion
-Double value
-Token token
+InicioOperacion(Double value, Token token)
+InicioOperacion()
+Double getValue()
+void setValue(Double value)
+Token getToken()
+void setToken(Token token)
NoDataException
+NoDataException()
+NoDataException(String message)