Universidad San Carlos de Guatemala Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Organización de Lenguajes y Compiladores 2 “A”

Ing. Bayron Lopez

Aux. Enio González, Esvin González

“Manual Técnico”

Kristhal Jasmine Meng Marroquín

201314655

Guatemala 4 de Abril del 2017

Objetivos

General

Proporcionar una guía para el lector del desarrollo de la interfaz y el manejo de cada clase con sus respectivos atributos y métodos para facilitar la edición y actualización del software.

Específicos

Brindar al lector una concepción técnica de la funcionalidad de los principales procesos del sistema incluyendo los conceptos de análisis léxico (Jflex), análisis sintáctico (Cup) con la ayuda del IDE NetBeans y lenguaje Java.

Proporcionar un lenguaje técnico por el cual el lector pueda obtener una descripción total del software que se está manejando y de esta manera no tener complicaciones al momento de realizar cambios en el funcionamiento del mismo.

Introducción

La finalidad de este manual técnico es dar a conocer al lector la lógica con la que se ha desarrollado la aplicación, incluyendo la descripción de todas las clases que se utilizaron para llevar a cabo el desarrollo de este software. Indica así el IDE que se utilizó para la creación del mismo, su versión y los requerimientos que debe poseer en su ordenador al momento de realizar alguna edición en el software.

Este Software está basado en la creación de un IDE que nos permite manejar un lenguaje de programación procedural y un lenguaje de programación orientado a objetos, el cual se complementa uno con el otro por medio de importaciones de llamadas. Asi mismo este software cuenta con la modalidad de cliente servidor para un uso sofisticado al momento de compartir proyectos con más usuarios del sistema.

Descripción de la solución

Para la Solución de este proyecto lo primero que se realizo fue un análisis de lo que el usuario nos solicitaba, las restricciones que este necesitaba y los códigos que requerían para el manejo de cada creación y las limitantes que debíamos utilizar para la creación del mismo.

Lo primordial fue el implementar algunas consideraciones como las siguientes:

Debe analizarse lexicalmente cada palabra para confirmar que esta es parte del lenguaje que será utilizado en la aplicación.

Debe analizarse sintácticamente la estructura del lenguaje para confirmar que esta venga en un orden coherente y permitido por la aplicación para la generación de las gráficas.

Todo debe manejarse desde un editor de texto y un menú que nos permita generar todas las acciones permitidas por el sistema.

Utilizar la herramienta de JFlex para poder realizar el análisis léxico y la herramienta de Cup para el análisis Sintáctico.

El usuario desea una aplicación en lenguaje Java que sea capaz de generar figuras mediante un lenguaje que el usuario conoce para generar gráficas y tablas mediante proyectos que puedan compartirse con el usuario.

Expresiones Regulares a implementar en Haskell++

```
texto      = "\""([^\"]*)"\""
numero     = [0-9]+
letras     = [a-zA-ZñÑ]+
id         ={letras}({numero}|"_"|{letras})*
decimal    ={numero}"."{numero}
caracter   = "\'"([a-zA-Z0-9]|"!"|"\""|"·"|"$"|"%"|"="|"?"|"¿"|"Ç"|"_"|"-
"|"."|","|"<"|">"|"@"|"#"|"~"|"½"|"¬"|";"|"["|"]"|"º"|"ª"|""|"!"|"+"|"-"|"*"|"^"|"&"|"\\"|" ")"\'"
```

Gramática a utilizar

```
terminal String numero, texto, identificador, decimal;
terminal String tkn_let, tkn_calcular, tkn_succ, tkn_decc, tkn_min, tkn_max, tkn_sum, tkn_product;
terminal String tkn_revers, tkn_impr, tkn_par, tkn_asc, tkn_desc, tkn_length, tkn_end, tkn_if, tkn_else;
terminal String tkn_then, tkn_case;
terminal String par_izq, par_der, llave_izq, llave_der, cor_izq, cor_der, mas, menos, div, potencia, por, mod;
terminal String sqrt, coma, dolar, tkn_or, tkn_and, mayor, menor, mayor_igual, menor_igual, pyc;
terminal String igualacion, diferente, aumento, dadmiracion, igual, dosp, caracter;

non terminal NodoHK INICIO, LST_SEN, SENT, FUNCION, LST_SENTENCIAS, SENTENCIAS;
non terminal NodoHK IF, CASE, LLAMADA_MET, CALCULAR, LISTA, SUCC, DECC, MIN;
non terminal NodoHK MAX, SUM, PRODUCT, REVERS, IMPR, PAR, ASC, DESC, LENGTH;
non terminal NodoHK LST_PAR, LPAR, LST_PARAMETROS, PARAMETROS, LST_CASOS, CASOS;
non terminal NodoHK LST_DIM, DIM, VALOR, MET_NATIVOS, LST_DIMS, DIMS, ASIGNA_LISTA, OPC_LISTA;
non terminal NodoHK CONCAT, LST_CON, CON;
non terminal NodoHK LOGICAS, RELACIONALES, ARITMETICAS, OPERACIONES;
non terminal NodoHK OTRA_DIM, ODIM, BUSCA_LIST, COND, LLAMADAS;
```

Tabla de precedencia

```
precedence left tkn_or;
precedence left tkn_and;
precedence left igualacion, diferente;
precedence left mayor, menor, mayor_igual, menor_igual;
precedence left mas, menos;
precedence left por, div, mod;
precedence right potencia, sqrt;
```

```
INICIO ::= LST_SEN:nodo
            {:
               parser.nodo=nodo;
            :};

LST_SEN ::= LST_SEN:nodo SENT:se
            {:
               nodo.hijos.add(se);
               RESULT = nodo;
            :}
          | SENT:se
            {:
               NodoHK nodo = new NodoHK("LST_FUNCIONES");
               nodo.hijos.add(se);
               RESULT = nodo;
            :};

SENT ::= FUNCION:nodo {:RESULT = nodo; :};

FUNCION ::= identificador:id LST_PARAMETROS:pa igual LST_SENTENCIAS:lst tkn_end
            {:
               NodoHK nodo = new NodoHK("FUNCION");
               nodo.hijos.add(new NodoHK(id, idleft, idright));
               nodo.hijos.add(pa);
               nodo.hijos.add(lst);
               RESULT = nodo;
            :}
           |identificador:id igual LST_SENTENCIAS:lst tkn_end
            {:
               NodoHK nodo = new NodoHK("FUNCION");
               nodo.hijos.add(new NodoHK(id, idleft, idright));
               nodo.hijos.add(new NodoHK("sinparametros"));
               nodo.hijos.add(lst);
               RESULT = nodo;
            :};

LST_PARAMETROS ::= LST_PARAMETROS:nodo coma PARAMETROS:pa
                    {:
                       nodo.hijos.add(pa);
```

```
                RESULT = nodo;
            :}
        |PARAMETROS:pa
            {:
                NodoHK nodo = new NodoHK("PARAMETROS");
                nodo.hijos.add(pa);
                RESULT = nodo;
            :};

PARAMETROS ::= identificador:id
        {:
            NodoHK nodo = new NodoHK(id, idleft, idright);
            RESULT = nodo;
        :};

LST_SENTENCIAS ::= LST_SENTENCIAS:nodo SENTENCIAS:se
        {:
            nodo.hijos.add(se);
            RESULT = nodo;
        :}
        |SENTENCIAS:se
        {:
            NodoHK nodo = new NodoHK("SENTENCIAS");
            nodo.hijos.add(se);
            RESULT = nodo;
        :};

SENTENCIAS ::= IF:nodo {: RESULT = nodo; :}
        |CASE:nodo {: RESULT = nodo; :}
        |LLAMADA_MET:nodo {: RESULT = nodo; :}
        |LISTA:nodo {: RESULT = nodo; :}
        |MET_NATIVOS:nodo {: RESULT = nodo; :}
        |CALCULAR:nodo {: RESULT = nodo; :}
        |BUSCA_LIST:nodo {: RESULT = nodo; :};

IF ::= tkn_if COND:lo tkn_then LST_SENTENCIAS:lst tkn_else LST_SENTENCIAS:lst2 tkn_end
        {:
            NodoHK nodo = new NodoHK("SINO");
            NodoHK si = new NodoHK("SI");
            si.hijos.add(lo);
            si.hijos.add(lst);
            nodo.hijos.add(si);
            nodo.hijos.add(lst2);
            RESULT = nodo;
        :};

CASE ::= tkn_case ARITMETICAS:lo LST_CASOS:lst tkn_end
        {:
```

```
            NodoHK nodo = new NodoHK("NCASE");
            nodo.hijos.add(lo);
            nodo.hijos.add(lst);
            RESULT = nodo;
          :};

LST_CASOS ::= LST_CASOS:nodo CASOS:ca
          {:
            nodo.hijos.add(ca);
            RESULT = nodo;
          :}
         |CASOS:ca
          {:
            NodoHK nodo = new NodoHK("LST_CASOS");
            nodo.hijos.add(ca);
            RESULT = nodo;
          :};

CASOS ::= VALOR:val dosp LST_SENTENCIAS:lst pyc
        {:
            NodoHK nodo = new NodoHK("CASO");
            nodo.hijos.add(val);
            nodo.hijos.add(lst);
            RESULT = nodo;
        :};

VALOR::= numero:num
        {:
            NodoHK nodo = new NodoHK("entero");
            nodo.hijos.add(new NodoHK(num, numleft, numright));
            RESULT = nodo;
        :}
      |decimal:dec
        {:
            NodoHK nodo = new NodoHK("decimal");
            nodo.hijos.add(new NodoHK(dec, decleft, decright));
            RESULT = nodo;
        :}
      |caracter:car
        {:
            NodoHK nodo = new NodoHK("caracter");
            nodo.hijos.add(new NodoHK(car, carleft, carright));
            RESULT = nodo;
        :};

LLAMADA_MET ::= dolar identificador:id llave_izq LST_PAR:param llave_der dolar
          {:
            NodoHK nodo = new NodoHK("LLAMADA_MET");
```

```
                nodo.hijos.add(new NodoHK(id, idleft, idright));
                nodo.hijos.add(param);
                RESULT = nodo;
            :}
           | dolar identificador:id llave_izq llave_der dolar
            {:
                NodoHK nodo = new NodoHK("LLAMADA_MET");
                nodo.hijos.add(new NodoHK(id, idleft, idright));
                RESULT = nodo;
            :};

LST_PAR ::= LST_PAR:nodo coma LPAR:pa
            {:
                nodo.hijos.add(pa);
                RESULT = nodo;
            :}
          |LPAR:pa
            {:
                NodoHK nodo = new NodoHK("PARAMETROS");
                nodo.hijos.add(pa);
                RESULT = nodo;
            :};

LPAR ::= ARITMETICAS:nodo {: RESULT = nodo; :}
      | cor_izq LST_DIMS:di cor_der
            {:
                NodoHK nodo = new NodoHK("lista");
                nodo.hijos.add(di);
                RESULT = nodo;
            :}
      | MET_NATIVOS:na
            {:
                RESULT = na;
            :};

MET_NATIVOS ::= SUCC:nodo {: RESULT = nodo; :}
            |DECC:nodo {: RESULT = nodo; :}
            |MIN:nodo {: RESULT = nodo; :}
            |MAX:nodo {: RESULT = nodo; :}
            |SUM:nodo {: RESULT = nodo; :}
            |PRODUCT:nodo {: RESULT = nodo; :}
            |REVERS:nodo {: RESULT = nodo; :}
            |IMPR:nodo {: RESULT = nodo; :}
            |PAR:nodo {: RESULT = nodo; :}
            |DESC:nodo {: RESULT = nodo; :}
            |ASC:nodo {: RESULT = nodo; :}
            |LENGTH:nodo {: RESULT = nodo; :}
            |CONCAT:nodo {: RESULT = nodo; :};
```

```
LISTA ::= tkn_let identificador:id igual ASIGNA_LISTA:asg
            {:
               NodoHK nodo = new NodoHK("DECLARA_ASIGNA_LISTA");
               nodo.hijos.add(new NodoHK(id, idleft, idright));
               nodo.hijos.add(asg);
               RESULT = nodo;
            :}
        |tkn_let identificador:id igual LLAMADAS:asg
            {:
               NodoHK nodo = new NodoHK("DECLARA_ASIGNA_LISTA");
               nodo.hijos.add(new NodoHK(id, idleft, idright));
               nodo.hijos.add(asg);
               RESULT = nodo;
            :}

        |tkn_let identificador:id igual BUSCA_LIST:asg
            {:
                        NodoHK nodo = new NodoHK("DECLARA_ASIGNA_LISTA");
               nodo.hijos.add(new NodoHK(id, idleft, idright));
               nodo.hijos.add(asg);
               RESULT = nodo;
            :};

ASIGNA_LISTA ::= cor_izq LST_DIMS:di cor_der
            {:
               NodoHK nodo = new NodoHK("lista");
               nodo.hijos.add(di);
               RESULT = nodo;
            :}
          |texto:cad
            {:
               NodoHK nodo = new NodoHK("cadena");
               nodo.hijos.add(new NodoHK(cad, cadleft, cadright));
               RESULT = nodo;
            :}
          |identificador:id
            {:
               NodoHK nodo = new NodoHK("identificador");
               nodo.hijos.add(new NodoHK(id, idleft, idright));
               RESULT = nodo;
            :};

LLAMADAS ::= LLAMADA_MET:me
          {:
            RESULT = me;
          :}
```

```
        |MET_NATIVOS:na
          {:
            RESULT = na;
          :};

LST_DIMS ::= LST_DIMS:nodo coma DIMS:di
          {:
            nodo.hijos.add(di);
            RESULT = nodo;
          :}
        |DIMS:di
          {:
            NodoHK nodo = new NodoHK("DIMENSIONES");
            nodo.hijos.add(di);
            RESULT = nodo;
          :};

DIMS ::= ARITMETICAS:nodo
          {:
            RESULT = nodo;
          :}
      |caracter:car
          {:
            NodoHK nodo = new NodoHK("caracter");
            nodo.hijos.add(new NodoHK(car, carleft, carright));
            RESULT = nodo;
          :}
      |cor_izq OTRA_DIM:di cor_der
          {:
            RESULT = di;
          :};

OTRA_DIM ::= OTRA_DIM:nodo coma ODIM:di
          {:
            nodo.hijos.add(di);
            RESULT = nodo;
          :}
        |ODIM:di
          {:
            NodoHK nodo = new NodoHK("DIMENSIONES");
            nodo.hijos.add(di);
            RESULT = nodo;
          :};

ODIM ::= ARITMETICAS:nodo
          {:
            RESULT = nodo;
          :}
```

```
    |caracter:car
        {:
            NodoHK nodo = new NodoHK("caracter");
            nodo.hijos.add(new NodoHK(car, carleft, carright));
            RESULT = nodo;
        :}
    |MET_NATIVOS:na
        {:
            RESULT = na;
        :};

SUCC ::= dolar tkn_succ ARITMETICAS:opc dolar
        {:
            NodoHK nodo = new NodoHK("SUCC");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :}
    |dolar tkn_succ MET_NATIVOS:opc dolar
        {:
            NodoHK nodo = new NodoHK("SUCC");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :};

DECC ::= dolar tkn_decc ARITMETICAS:opc dolar
        {:
            NodoHK nodo = new NodoHK("DECC");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :}
    |dolar tkn_decc MET_NATIVOS:opc dolar
        {:
            NodoHK nodo = new NodoHK("DECC");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :};

OPC_LISTA ::= ASIGNA_LISTA:asg
        {:
            RESULT = asg;
        :};

MIN ::= dolar tkn_min OPC_LISTA:opc dolar
        {:
            NodoHK nodo = new NodoHK("MIN");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :}
```

```
    |dolar tkn_min MET_NATIVOS:opc dolar
        {:
            NodoHK nodo = new NodoHK("MIN");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :}
    |dolar tkn_min BUSCA_LIST:opc dolar
        {:
            NodoHK nodo = new NodoHK("MIN");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :};

MAX ::= dolar tkn_max OPC_LISTA:opc dolar
        {:
            NodoHK nodo = new NodoHK("MAX");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :}
    |dolar tkn_max MET_NATIVOS:opc dolar
        {:
            NodoHK nodo = new NodoHK("MAX");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :}
    |dolar tkn_max BUSCA_LIST:opc dolar
        {:
            NodoHK nodo = new NodoHK("MAX");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :};

SUM ::= dolar tkn_sum OPC_LISTA:opc dolar
        {:
            NodoHK nodo = new NodoHK("SUM");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :}
    |dolar tkn_sum MET_NATIVOS:opc dolar
        {:
            NodoHK nodo = new NodoHK("SUM");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :}
    |dolar tkn_sum BUSCA_LIST:opc dolar
        {:
            NodoHK nodo = new NodoHK("SUM");
            nodo.hijos.add(opc);
```

```
            RESULT = nodo;
        :};

PRODUCT ::= dolar tkn_product OPC_LISTA:opc dolar
        {:
            NodoHK nodo = new NodoHK("PRODUCT");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :}
      |dolar tkn_product MET_NATIVOS:opc dolar
        {:
            NodoHK nodo = new NodoHK("PRODUCT");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :}
      |dolar tkn_product BUSCA_LIST:opc dolar
        {:
            NodoHK nodo = new NodoHK("PRODUCT");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :};

REVERS ::= dolar tkn_revers OPC_LISTA:opc dolar
        {:
            NodoHK nodo = new NodoHK("REVERS");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :}
      |dolar tkn_revers MET_NATIVOS:opc dolar
        {:
            NodoHK nodo = new NodoHK("REVERS");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :}
      |dolar tkn_revers BUSCA_LIST:opc dolar
        {:
            NodoHK nodo = new NodoHK("REVERS");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :};

IMPR ::= dolar tkn_impr OPC_LISTA:opc dolar
        {:
            NodoHK nodo = new NodoHK("IMPR");
            nodo.hijos.add(opc);
            RESULT = nodo;
        :}
```

```
        |dolar tkn_impr MET_NATIVOS:opc dolar
            {:
               NodoHK nodo = new NodoHK("IMPR");
               nodo.hijos.add(opc);
               RESULT = nodo;
            :}
        |dolar tkn_impr BUSCA_LIST:opc dolar
            {:
               NodoHK nodo = new NodoHK("IMPR");
               nodo.hijos.add(opc);
               RESULT = nodo;
            :};

PAR ::= dolar tkn_par OPC_LISTA:opc dolar
            {:
               NodoHK nodo = new NodoHK("PAR");
               nodo.hijos.add(opc);
               RESULT = nodo;
            :}
       |dolar tkn_par MET_NATIVOS:opc dolar
            {:
               NodoHK nodo = new NodoHK("PAR");
               nodo.hijos.add(opc);
               RESULT = nodo;
            :}

        |dolar tkn_par BUSCA_LIST:opc dolar
            {:
               NodoHK nodo = new NodoHK("PAR");
               nodo.hijos.add(opc);
               RESULT = nodo;
            :};

ASC ::= dolar tkn_asc OPC_LISTA:opc dolar
            {:
               NodoHK nodo = new NodoHK("ASC");
               nodo.hijos.add(opc);
               RESULT = nodo;
            :}
        |dolar tkn_asc MET_NATIVOS:opc dolar
            {:
               NodoHK nodo = new NodoHK("ASC");
               nodo.hijos.add(opc);
               RESULT = nodo;
            :}
        |dolar tkn_asc BUSCA_LIST:opc dolar
            {:
               NodoHK nodo = new NodoHK("ASC");
```

```
                nodo.hijos.add(opc);
                RESULT = nodo;
            :};

DESC ::= dolar tkn_desc OPC_LISTA:opc dolar
            {:
                NodoHK nodo = new NodoHK("DESC");
                nodo.hijos.add(opc);
                RESULT = nodo;
            :}
        |dolar tkn_desc MET_NATIVOS:opc dolar
            {:
                NodoHK nodo = new NodoHK("DESC");
                nodo.hijos.add(opc);
                RESULT = nodo;
            :}
        |dolar tkn_desc BUSCA_LIST:opc dolar
            {:
                NodoHK nodo = new NodoHK("DESC");
                nodo.hijos.add(opc);
                RESULT = nodo;
            :};

LENGTH ::= dolar tkn_length OPC_LISTA:opc dolar
            {:
                NodoHK nodo = new NodoHK("LENGTH");
                nodo.hijos.add(opc);
                RESULT = nodo;
            :}
         |dolar tkn_length MET_NATIVOS:opc dolar
            {:
                NodoHK nodo = new NodoHK("LENGTH");
                nodo.hijos.add(opc);
                RESULT = nodo;
            :}
         |dolar tkn_length BUSCA_LIST:opc dolar
            {:
                NodoHK nodo = new NodoHK("LENGTH");
                nodo.hijos.add(opc);
                RESULT = nodo;
            :};

CONCAT ::= ASIGNA_LISTA:as1 LST_CON:as2
            {:
                NodoHK nodo = new NodoHK("CONCAT");
                nodo.hijos.add(as1);
                nodo.hijos.add(as2);
                RESULT = nodo;
```

```
            :};

LST_CON ::= LST_CON:nodo CON:co
            {:
               nodo.hijos.add(co);
               RESULT = nodo;
            :}
         | CON:co
            {:
               NodoHK nodo = new NodoHK("LST_CONCAT");
               nodo.hijos.add(co);
               RESULT = nodo;
            :};

CON ::= aumento ASIGNA_LISTA:as
          {:
             RESULT = as;
          :};

BUSCA_LIST ::= identificador:id LST_DIM:dime
            {:
               NodoHK nodo = new NodoHK("INDICE");
               NodoHK iden = new NodoHK("identificador");
               iden.hijos.add(new NodoHK(id, idleft, idright));
               nodo.hijos.add(iden);
               nodo.hijos.add(dime);
               RESULT = nodo;
            :};

LST_DIM ::= LST_DIM:nodo DIM:di
            {:
               nodo.hijos.add(di);
               RESULT = nodo;
            :}
          |DIM:di
            {:
               NodoHK nodo = new NodoHK("DIMENSIONES");
               nodo.hijos.add(di);
               RESULT = nodo;
            :};

DIM ::= dadmiracion CALCULAR:nodo
          {:
             RESULT = nodo;
          :}
       |dadmiracion LLAMADAS:nodo
          {:
             RESULT = nodo;
```

```
        :}
      |dadmiracion identificador:id
         {:
            NodoHK nodo = new NodoHK("identificador");
            nodo.hijos.add(new NodoHK(id, idleft, idright));
            RESULT = nodo;
         :};

COND ::= LOGICAS:log {:RESULT = log;:};

LOGICAS ::= LOGICAS:log1 tkn_or LOGICAS:log2
            {:
               NodoHK nodo = new NodoHK("or");
               nodo.hijos.add(log1);
               nodo.hijos.add(log2);
               RESULT = nodo;
            :}
          | LOGICAS:log1 tkn_and LOGICAS:log2
            {:
               NodoHK nodo = new NodoHK("and");
               nodo.hijos.add(log1);
               nodo.hijos.add(log2);
               RESULT = nodo;
            :}
          | RELACIONALES:nodo
            {:
               RESULT = nodo;
            :};

RELACIONALES ::= RELACIONALES:re1 igualacion RELACIONALES:re2
                 {:
                    NodoHK nodo = new NodoHK("igualacion");
                    nodo.hijos.add(re1);
                    nodo.hijos.add(re2);
                    RESULT = nodo;
                 :}
               |RELACIONALES:re1 diferente RELACIONALES:re2
                 {:
                    NodoHK nodo = new NodoHK("diferente");
                    nodo.hijos.add(re1);
                    nodo.hijos.add(re2);
                    RESULT = nodo;
                 :}
               |RELACIONALES:re1 mayor RELACIONALES:re2
                 {:
                    NodoHK nodo = new NodoHK("mayor");
                    nodo.hijos.add(re1);
                    nodo.hijos.add(re2);
```

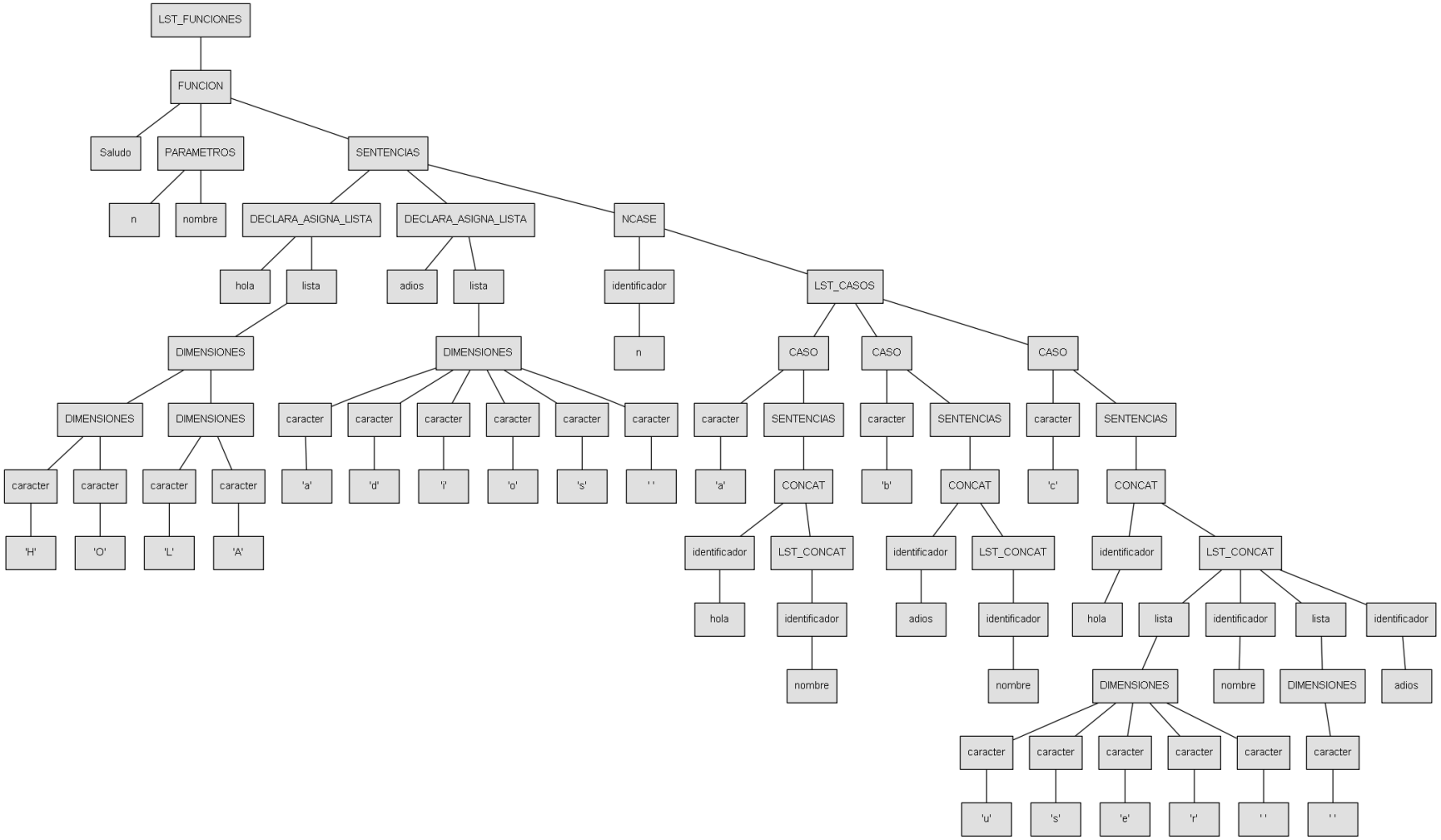
```
                RESULT = nodo;
              :}
            |RELACIONALES:re1 menor RELACIONALES:re2
              {:
                NodoHK nodo = new NodoHK("menor");
                nodo.hijos.add(re1);
                nodo.hijos.add(re2);
                RESULT = nodo;
              :}
            |RELACIONALES:re1 mayor_igual RELACIONALES:re2
              {:
                NodoHK nodo = new NodoHK("mayorI");
                nodo.hijos.add(re1);
                nodo.hijos.add(re2);
                RESULT = nodo;
              :}
            |RELACIONALES:re1 menor_igual RELACIONALES:re2
              {:
                NodoHK nodo = new NodoHK("menorI");
                nodo.hijos.add(re1);
                nodo.hijos.add(re2);
                RESULT = nodo;
              :}
            |cor_izq LST_DIMS:di cor_der
              {:
                RESULT = di;
              :}
            |ARITMETICAS:nodo
              {:
                RESULT = nodo;
              :};

ARITMETICAS ::= CALCULAR:nodo
              {:
                RESULT = nodo;
              :}
           | identificador:id
              {:
                NodoHK nodo = new NodoHK("identificador");
                nodo.hijos.add(new NodoHK(id, idleft, idright));
                RESULT = nodo;
              :}
           | BUSCA_LIST:nodo
              {:
                RESULT = nodo;
              :}
           | texto:tex
              {:
```

```
                NodoHK nodo = new NodoHK("cadena");
                nodo.hijos.add(new NodoHK(tex));
                RESULT = nodo;
            :}

        | par_izq COND:log par_der
            {:
                RESULT = log;
            :}

        | LLAMADA_MET:nodo
            {:
                RESULT = nodo;
            :};


CALCULAR ::= dolar tkn_calcular OPERACIONES:op dolar
            {:
                NodoHK nodo = new NodoHK("CALCULAR");
                nodo.hijos.add(op);
                RESULT = nodo;
            :};

OPERACIONES ::= OPERACIONES:op1 mas OPERACIONES:op2
            {:
                NodoHK nodo = new NodoHK("mas");
                nodo.hijos.add(op1);
                nodo.hijos.add(op2);
                RESULT = nodo;
            :}
          |OPERACIONES:op1 menos OPERACIONES:op2
            {:
                NodoHK nodo = new NodoHK("menos");
                nodo.hijos.add(op1);
                nodo.hijos.add(op2);
                RESULT = nodo;
            :}
          |OPERACIONES:op1 por OPERACIONES:op2
            {:
                NodoHK nodo = new NodoHK("por");
                nodo.hijos.add(op1);
                nodo.hijos.add(op2);
                RESULT = nodo;
            :}
          |OPERACIONES:op1 div OPERACIONES:op2
            {:
                NodoHK nodo = new NodoHK("div");
                nodo.hijos.add(op1);
```

```
        nodo.hijos.add(op2);
        RESULT = nodo;
    :}
|OPERACIONES:op1 mod OPERACIONES:op2
    {:
        NodoHK nodo = new NodoHK("mod");
        nodo.hijos.add(op1);
        nodo.hijos.add(op2);
        RESULT = nodo;
    :}
|OPERACIONES:op1 potencia OPERACIONES:op2
    {:
        NodoHK nodo = new NodoHK("potencia");
        nodo.hijos.add(op1);
        nodo.hijos.add(op2);
        RESULT = nodo;
    :}
|OPERACIONES:op1 sqrt OPERACIONES:op2
    {:
        NodoHK nodo = new NodoHK("sqrt");
        nodo.hijos.add(op1);
        nodo.hijos.add(op2);
        RESULT = nodo;
    :}
|menos OPERACIONES:op
    {:
        NodoHK nodo = new NodoHK("menos");
        nodo.hijos.add(op);
        RESULT = nodo;
    :}
|numero:num
    {:
        NodoHK nodo = new NodoHK("entero");
        nodo.hijos.add(new NodoHK(num, numleft, numright));
        RESULT = nodo;
    :}
|decimal:dec
    {:
        NodoHK nodo = new NodoHK("decimal");
        nodo.hijos.add(new NodoHK(dec, decleft, decright));
        RESULT = nodo;
    :}
|identificador:id
    {:
        NodoHK nodo = new NodoHK("identificador");
        nodo.hijos.add(new NodoHK(id, idleft, idright));
        RESULT = nodo;
    :}
```

```
|par_izq OPERACIONES:nodo par_der
   {:
      RESULT = nodo;
   :}
|LLAMADA_MET:nodo
   {:
      RESULT = nodo;
   :}
|MET_NATIVOS:nodo
   {:
      RESULT = nodo;
   :}
|BUSCA_LIST:nodo
   {:
      RESULT = nodo;
   :}
|CALCULAR:nodo {: RESULT = nodo; :};
```

Arbol generado para Haskell++



Expresiones Regulares a Implementar en Graphik

```
texto       = "\""([^\"]*)"\""
caracter    = "\'"([a-zA-Z0-9]|"!"|"\""|"·"|"$"|"%"|"="|"?"|"¿"|"Ç"|"_"|"-
"|"."|","|"<"|">"|"@"|"#"|"~"|"½"|"¬"|";"|"["|"]"|"º"|"ª"|""|"!"|"+"|"-"|"*"|"^"|"&"|"\\"|" ")"\'"
numero      = [0-9]+
letras      = [a-zA-ZñÑ]+
id          ={letras}({numero}|"_"|{letras})*
decimal     ={numero}"."{numero}
nombreArchivo = {id}".gk"
Comment = {TraditionalComment} | {EndOfLineComment}
TraditionalComment   = "#/" ([^/]*)"/#"
EndOfLineComment     = "#"([^\n]*)
```

Gramatica a Utilizar en Graphik

```
terminal String numero, texto, identificador, decimal, caracter;
terminal String tkn_verdadero, tkn_falso, tkn_entero, tkn_decimal, tkn_caracter, tkn_cadena, tkn_bool;
terminal String tkn_vacio, tkn_publico, tkn_privado, tkn_protegido, tkn_importar, tkn_als, tkn_hereda;
terminal String tkn_var, tkn_nuevo, tkn_retornar, tkn_llamar, tkn_inicio, tkn_incluirHK, tkn_llamarHK;
terminal String tkn_si, tkn_sino, tkn_seleccion, tkn_caso, tkn_defecto, tkn_para, tkn_mientras;
terminal String tkn_hacer, tkn_continuar, tkn_terminar, tkn_graphikar, tkn_datos, tkn_columna, tkn_procesar, tkn_donde;
terminal String tkn_dondecada, tkn_dondetodo, tkn_imprimir, tkn_imprimirK;
terminal String par_izq, par_der, cor_izq, cor_der, llave_izq, llave_der, mas, menos, div, potencia, por;
terminal String coma, int_der, tkn_or, tkn_xor, tkn_and, tkn_not, mayor, menor, mayor_igual, menor_igual;
terminal String igualacion, diferente, aumento, decremento, igual, dosp, punto, nombreArchivo;

non terminal NodoGK INICIO, SENTENCIAS, LST_SEN, LST_SENTENCIAS, SENTS, IMPRIMIR;
non terminal NodoGK LST_IMPORTAR, LLAMADASHK, CUERPO, IMPORTAR, LLAMADAS;
non terminal NodoGK VISIBILIDAD, METODO, FUNCION, LST_PARAMETROS, PARAMETRO, MET_INICIO;
non terminal NodoGK TIPO, DECLARACION, VARIABLES, ASIGNACION, LST_VARIABLES, VAR;
non terminal NodoGK SI, SINO, MIENTRAS, HACERMIENTRAS, SELECCIONA, CONTINUAR, TERMINAR;
non terminal NodoGK PARA, LST_CASOS, CASOS, RETORNO, LLAMADA_MET, LST_PARAM, PARAM;
non terminal NodoGK ACCESOBJ, LST_NIVEL, NIVEL, GRAFICAR, LLAMADA_HK;
non terminal NodoGK IMPORTS, LLAMARHK, ACCION;
```

```
non terminal NodoGK ARREGLOS, DARR, AARR, LST_DIM, DIM;
non terminal NodoGK DATOS, COLUMNA, LST_COLUMNA, PROCESAR, DONDE, DONDECADA,
DONDETODO, HASKELL, NORMAL;
non terminal NodoGK SENT_DATOS, SDATOS, EXP, MET_ESPECIALES, LST_DARR;
non terminal NodoGK EXPRESION, LST_CUERPO, ACCESO_MET, ASIG_PARA, OPCIONES;
```

Tabla de Precedencia

```
precedence left tkn_or;
precedence left tkn_xor;
precedence left tkn_and;
precedence right tkn_not;
precedence left igualacion, diferente;
precedence left mayor, menor, mayor_igual, menor_igual;
precedence left mas, menos;
precedence left por, div;
precedence right potencia;
precedence left aumento, decremento;
```

```
INICIO ::=  IMPORTS:imp LLAMARHK:llam LST_CUERPO:cue
            {:
               NodoGK nodo = new NodoGK("GRAPHIK");
               nodo.hijos.add(imp);
               nodo.hijos.add(llam);
               nodo.hijos.add(cue);
               parser.nodo=nodo;
            :};

IMPORTS ::= LST_IMPORTAR:nodo {: RESULT = nodo;:}
         |/**/
            {:
               NodoGK nodo = new NodoGK("IMPORTS");
               RESULT = nodo;
            :};

LLAMARHK ::= LLAMADASHK:nodo {: RESULT = nodo;:}
          |/**/
            {:
               NodoGK nodo = new NodoGK("LLAMADASHK");
               RESULT = nodo;
            :};
```

```
LST_IMPORTAR ::= LST_IMPORTAR:nodo IMPORTAR:imp
                {:
                   nodo.hijos.add(imp);
                   RESULT = nodo;
                :}
               |IMPORTAR:imp
                {:
                   NodoGK nodo = new NodoGK("IMPORTS");
                   nodo.hijos.add(imp);
                   RESULT = nodo;
                :};

IMPORTAR ::= tkn_importar nombreArchivo:id int_der
            {:
               NodoGK nodo = new NodoGK(id, idleft, idright);
               RESULT = nodo;
            :};

LLAMADASHK::= LLAMADASHK:nodo LLAMADAS:llamada
            {:
               nodo.hijos.add(llamada);
               RESULT = nodo;
            :}
           | LLAMADAS:llamada
            {:
               NodoGK nodo = new NodoGK("LLAMADASHK");
               nodo.hijos.add(llamada);
               RESULT = nodo;
            :};

LLAMADAS ::= tkn_incluirHK identificador:id int_der
            {:
               NodoGK nodo = new NodoGK(id, idleft, idright);
               RESULT = nodo;
            :};

LST_CUERPO::= LST_CUERPO:nodo CUERPO:cu
            {:
               nodo.hijos.add(cu);
               RESULT = nodo;
            :}
           |CUERPO:cu
            {:
               NodoGK nodo = new NodoGK("CLASES");
               nodo.hijos.add(cu);
               RESULT = nodo;
            :};
```

```
CUERPO::= tkn_als identificador:id VISIBILIDAD:vis llave_izq LST_SEN:lst llave_der
        {:
           NodoGK nodo = new NodoGK("ALS");
           nodo.hijos.add(new NodoGK(id, idleft, idright));
           nodo.hijos.add(vis);
           nodo.hijos.add(lst);
           RESULT = nodo;
        :}
      |tkn_als identificador:id tkn_hereda identificador:id2 VISIBILIDAD:vis llave_izq LST_SEN:lst
llave_der
        {:
           NodoGK nodo = new NodoGK("ALS");
           nodo.hijos.add(new NodoGK(id, idleft, idright));
           nodo.hijos.add(new NodoGK(id2, id2left, id2right));
           nodo.hijos.add(vis);
           nodo.hijos.add(lst);
           RESULT = nodo;
        :};

VISIBILIDAD ::= dosp tkn_publico:pub
              {:
                 NodoGK nodo = new NodoGK(pub, publeft, pubright);
                 RESULT = nodo;
              :}
            |dosp tkn_privado: pri
              {:
                 NodoGK nodo = new NodoGK(pri, prileft, priright);
                 RESULT = nodo;
              :}
            |dosp tkn_protegido:pro
              {:
                 NodoGK nodo = new NodoGK(pro, proleft, proright);
                 RESULT = nodo;
              :}
            |/*Epsilon*/
              {:
                 NodoGK nodo = new NodoGK("publico");
                 RESULT = nodo;
              :};

LST_SEN ::= LST_SEN:nodo SENTS:sen
          {:
             nodo.hijos.add(sen);
             RESULT = nodo;
          :}
        |SENTS:sen
          {:
             NodoGK nodo = new NodoGK("CUERPO");
```

```
            nodo.hijos.add(sen);
            RESULT = nodo;
        :};

SENTS ::= METODO:nodo {: RESULT = nodo; :}
    | FUNCION:nodo {: RESULT = nodo; :}
    | MET_INICIO:nodo {: RESULT = nodo; :}
    | DATOS:nodo {: RESULT = nodo; :}
    | DECLARACION:nodo int_der {: RESULT = nodo; :}
    | ARREGLOS: nodo int_der {: RESULT = nodo; :};

METODO ::= tkn_vacio identificador:id par_izq LST_PARAMETROS:par par_der VISIBILIDAD:vis llave_izq
LST_SENTENCIAS:lst llave_der
        {:
            NodoGK nodo = new NodoGK("METODO");
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            nodo.hijos.add(par);
            nodo.hijos.add(vis);
            nodo.hijos.add(lst);
            RESULT = nodo;
        :}
      |tkn_vacio identificador:id par_izq par_der VISIBILIDAD:vis llave_izq LST_SENTENCIAS:lst llave_der
        {:
            NodoGK nodo = new NodoGK("METODO");
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            nodo.hijos.add(new NodoGK("sinparametros"));
            nodo.hijos.add(vis);
            nodo.hijos.add(lst);
            RESULT = nodo;
        :};

FUNCION ::= TIPO:tpo identificador:id par_izq LST_PARAMETROS:par par_der VISIBILIDAD:vis llave_izq
LST_SENTENCIAS:lst llave_der
        {:
            NodoGK nodo = new NodoGK("FUNCION");
            nodo.hijos.add(tpo);
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            nodo.hijos.add(par);
            nodo.hijos.add(vis);
            nodo.hijos.add(lst);
            RESULT = nodo;
        :}
      | TIPO:tpo identificador:id par_izq par_der VISIBILIDAD:vis llave_izq LST_SENTENCIAS:lst llave_der
        {:
            NodoGK nodo = new NodoGK("FUNCION");
            nodo.hijos.add(tpo);
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            nodo.hijos.add(new NodoGK("sinparametros"));
```

```
            nodo.hijos.add(vis);
            nodo.hijos.add(lst);
            RESULT = nodo;
        :};

LST_PARAMETROS ::= LST_PARAMETROS:nodo coma PARAMETRO:par
            {:
                nodo.hijos.add(par);
                RESULT = nodo;
            :}
          |PARAMETRO:par
            {:
                NodoGK nodo = new NodoGK("PARAMETROS");
                nodo.hijos.add(par);
                RESULT = nodo;
            :};

PARAMETRO ::= TIPO:tpo identificador:id
        {:
            NodoGK nodo = new NodoGK("PAR");
            nodo.hijos.add(tpo);
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            RESULT = nodo;
        :};

MET_INICIO ::= tkn_vacio tkn_inicio par_izq par_der llave_izq LST_SENTENCIAS:lst llave_der
        {:
            NodoGK nodo = new NodoGK("MAIN");
            nodo.hijos.add(lst);
            RESULT = nodo;
        :};

TIPO ::= tkn_cadena: cad
        {:
            NodoGK nodo = new NodoGK(cad, cadleft, cadright);
            RESULT = nodo;
        :}
      |tkn_caracter:car
        {:
            NodoGK nodo = new NodoGK(car, carleft, carright);
            RESULT = nodo;
        :}
      |tkn_decimal:dec
        {:
            NodoGK nodo = new NodoGK(dec, decleft, decright);
            RESULT = nodo;
        :}
      |tkn_entero:ent
```

```
        {:
            NodoGK nodo = new NodoGK(ent, entleft, entright);
            RESULT = nodo;
        :}
      |tkn_bool:bool
        {:
            NodoGK nodo = new NodoGK(bool, boolleft, boolright);
            RESULT = nodo;
        :}
      |identificador:id
        {:
            NodoGK nodo = new NodoGK(id, idleft, idright);
            RESULT = nodo;
        :};

LST_SENTENCIAS ::= LST_SENTENCIAS:nodo SENTENCIAS:sent
                {:
                    nodo.hijos.add(sent);
                    RESULT = nodo;
                :}
             | SENTENCIAS:sent
                {:
                    NodoGK nodo = new NodoGK("SENTENCIAS");
                    nodo.hijos.add(sent);
                    RESULT = nodo;
                :};

SENTENCIAS ::= SI:nodo {: RESULT = nodo; :}
          | SINO:nodo {: RESULT = nodo; :}
          | SELECCIONA:nodo {: RESULT = nodo; :}
          | PARA:nodo {: RESULT = nodo; :}
          | MIENTRAS:nodo {: RESULT = nodo; :}
          | HACERMIENTRAS:nodo {: RESULT = nodo; :}
          | CONTINUAR:nodo {: RESULT = nodo; :}
          | TERMINAR:nodo {: RESULT = nodo; :}
          | RETORNO:nodo {: RESULT = nodo; :}
          | LLAMADA_MET:nodo int_der {: RESULT = nodo; :}
          | LLAMADA_HK:nodo int_der {: RESULT = nodo; :}
          | GRAFICAR:nodo {: RESULT = nodo; :}
          | IMPRIMIR:nodo {: RESULT = nodo; :}
          | EXPRESION:exp aumento int_der
              {:
                  NodoGK nodo = new NodoGK("aumento");
                  nodo.hijos.add(exp);
                  RESULT = nodo;
              :}
          | EXPRESION:exp decremento int_der
              {:
```

```
                NodoGK nodo = new NodoGK("decremento");
                nodo.hijos.add(exp);
                RESULT = nodo;
            :}
        | DECLARACION:nodo int_der {: RESULT = nodo; :}
        | ASIGNACION:nodo int_der {: RESULT = nodo; :}
        | ARREGLOS:nodo int_der {: RESULT = nodo; :}
        | ACCESOBJ:nodo int_der {: RESULT = nodo; :};

DECLARACION ::= VARIABLES:nodo {: RESULT = nodo; :};

VARIABLES ::= tkn_var TIPO:tpo LST_VARIABLES:lst
            {:
                NodoGK nodo = new NodoGK("DECLARA_VAR");
                nodo.hijos.add(tpo);
                nodo.hijos.add(lst);
                RESULT = nodo;
            :}
        |tkn_var TIPO:tpo identificador:id VISIBILIDAD:vis igual EXPRESION:expr
            {:
                NodoGK nodo = new NodoGK("DECLARA_ASIG_VAR");
                nodo.hijos.add(tpo);
                nodo.hijos.add(new NodoGK(id, idleft, idright));
                nodo.hijos.add(vis);
                nodo.hijos.add(expr);
                RESULT = nodo;
            :}
        |tkn_var TIPO:tip identificador:id VISIBILIDAD:vis igual tkn_nuevo identificador:id2 par_izq
par_der
            {:
                NodoGK nodo = new NodoGK("DECLARA_ASIG_OBJ");
                nodo.hijos.add(tip);
                nodo.hijos.add(new NodoGK(id, idleft, idright));
                nodo.hijos.add(vis);
                nodo.hijos.add(new NodoGK(id2, idleft, idright));
                RESULT = nodo;
            :};

ASIGNACION ::= identificador:id igual EXPRESION:expr
            {:
                NodoGK nodo = new NodoGK("ASIGNACION");
                nodo.hijos.add(new NodoGK(id, idleft, idright));
                nodo.hijos.add(expr);
                RESULT = nodo;
            :}
        |identificador:id igual tkn_nuevo identificador:id2 par_izq par_der
            {:
                NodoGK nodo = new NodoGK("ASIG_OBJ");
```

```
                nodo.hijos.add(new NodoGK(id, idleft, idright));
                nodo.hijos.add(new NodoGK(id2, id2left, id2right));
                RESULT = nodo;
            :}
          |ACCESOBJ:acc igual EXPRESION:exp
            {:
                NodoGK nodo = new NodoGK("ASIG_ACCESOBJ");
                nodo.hijos.add(acc);
                nodo.hijos.add(exp);
                RESULT = nodo;
            :}
          |ACCESOBJ:acc igual tkn_nuevo identificador:id2 par_izq par_der
            {:
                NodoGK nodo = new NodoGK("ACCESOBJ_INSTANCIA");
                nodo.hijos.add(acc);
                nodo.hijos.add(new NodoGK(id2, id2left, id2right));
                RESULT = nodo;
            :};

LST_VARIABLES ::= LST_VARIABLES:nodo coma VAR:vars
            {:
                nodo.hijos.add(vars);
                RESULT = nodo;
            :}
           | VAR:vars
            {:
                NodoGK nodo = new NodoGK("VARIABLES");
                nodo.hijos.add(vars);
                RESULT = nodo;
            :};

VAR ::= identificador:id VISIBILIDAD:vis
        {:
            NodoGK nodo = new NodoGK("VAR");
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            nodo.hijos.add(vis);
            RESULT = nodo;
        :};

SI ::= tkn_si par_izq EXPRESION:expr par_der llave_izq LST_SENTENCIAS:lst llave_der
        {:
            NodoGK nodo = new NodoGK("SI");
            nodo.hijos.add(expr);
            nodo.hijos.add(lst);
            RESULT = nodo;
        :};

SINO ::= SI:nsi tkn_sino llave_izq LST_SENTENCIAS:lst llave_der
```

```
        {:
            NodoGK nodo = new NodoGK("SINO");
            nodo.hijos.add(nsi);
            nodo.hijos.add(lst);
            RESULT = nodo;
        :};

SELECCIONA ::= tkn_seleccion EXPRESION:expr llave_izq LST_CASOS:lst llave_der
            {:
                NodoGK nodo = new NodoGK("SELECCIONA");
                nodo.hijos.add(expr);
                nodo.hijos.add(lst);
                RESULT = nodo;
            :};

LST_CASOS ::= LST_CASOS:nodo CASOS:cas
            {:
                nodo.hijos.add(cas);
                RESULT = nodo;
            :}
          | CASOS:cas
            {:
                NodoGK nodo = new NodoGK("CASOS");
                nodo.hijos.add(cas);
                RESULT = nodo;
            :};

CASOS ::= tkn_caso EXPRESION:exp dosp LST_SENTENCIAS:lstsen
        {:
            NodoGK nodo = new NodoGK("CASE");
            nodo.hijos.add(exp);
            nodo.hijos.add(lstsen);
            RESULT = nodo;
        :}
       |tkn_defecto dosp LST_SENTENCIAS:lstsen
        {:
            NodoGK nodo = new NodoGK("DEFECTO");
            nodo.hijos.add(lstsen);
            RESULT = nodo;
        :};

PARA ::= tkn_para par_izq ASIG_PARA:asig dosp EXPRESION:exp dosp ACCION:acc par_der llave_izq
LST_SENTENCIAS:lstsen llave_der
        {:
            NodoGK nodo = new NodoGK("PARA");
            nodo.hijos.add(asig);
            nodo.hijos.add(exp);
            nodo.hijos.add(acc);
```

```
            nodo.hijos.add(lstsen);
            RESULT = nodo;
          :};

ASIG_PARA ::= tkn_var TIPO:tpo identificador:id igual EXPRESION:expr
            {:
               NodoGK nodo = new NodoGK("DECLARA_ASIG_VAR");
               nodo.hijos.add(tpo);
               nodo.hijos.add(new NodoGK(id, idleft, idright));
               nodo.hijos.add(new NodoGK("publico"));
               nodo.hijos.add(expr);
               RESULT = nodo;
            :}
          | ASIGNACION:asg
            {: RESULT = asg; :};

ACCION ::= EXPRESION:exp aumento
            {:
               NodoGK nodo = new NodoGK("aumento");
               nodo.hijos.add(exp);
               RESULT = nodo;
            :}
          |EXPRESION:exp decremento
            {:
               NodoGK nodo = new NodoGK("decremento");
               nodo.hijos.add(exp);
               RESULT = nodo;
            :}
          |ASIGNACION:nodo {: RESULT = nodo; :};

MIENTRAS ::= tkn_mientras par_izq EXPRESION:expr par_der llave_izq LST_SENTENCIAS:lstsen llave_der
               {:
                  NodoGK nodo = new NodoGK("MIENTRAS");
                  nodo.hijos.add(expr);
                  nodo.hijos.add(lstsen);
                  RESULT = nodo;
               :};

HACERMIENTRAS ::= tkn_hacer llave_izq LST_SENTENCIAS:lstsen llave_der tkn_mientras par_izq
EXPRESION:expr par_der
               {:
                  NodoGK nodo = new NodoGK("HACERMIENTRAS");
                  nodo.hijos.add(expr);
                  nodo.hijos.add(lstsen);
                  RESULT = nodo;
               :};
```

```
CONTINUAR ::= tkn_continuar int_der
            {:
               NodoGK nodo = new NodoGK("CONTINUAR");
               RESULT = nodo;
            :};

TERMINAR ::= tkn_terminar int_der
            {:
               NodoGK nodo = new NodoGK("TERMINAR");
               RESULT = nodo;
            :};

RETORNO ::= tkn_retornar EXPRESION:expr int_der
            {:
               NodoGK nodo = new NodoGK("RETORNO");
               nodo.hijos.add(expr);
               RESULT = nodo;
            :}
          |tkn_retornar int_der
            {:
               NodoGK nodo = new NodoGK("RETORNO");
               RESULT = nodo;
            :};

LLAMADA_MET ::= tkn_llamar identificador:id par_izq LST_PARAM:lstpar par_der
            {:
               NodoGK nodo = new NodoGK("LLAMAR_MET");
               nodo.hijos.add(new NodoGK(id, idleft, idright));
               nodo.hijos.add(lstpar);
               RESULT = nodo;
            :}
            |tkn_llamar identificador:id par_izq par_der
            {:
               NodoGK nodo = new NodoGK("LLAMAR_MET");
               nodo.hijos.add(new NodoGK(id, idleft, idright));
               nodo.hijos.add(new NodoGK("sinparametros"));
               RESULT = nodo;
            :}
            |tkn_llamar tkn_datos:id par_izq par_der
            {:
               NodoGK nodo = new NodoGK("LLAMAR_MET");
               nodo.hijos.add(new NodoGK(id, idleft, idright));
               nodo.hijos.add(new NodoGK("sinparametros"));
               RESULT = nodo;
            :};

LLAMADA_HK ::= tkn_llamarHK identificador:id par_izq LST_PARAM:lstpar par_der
            {:
```

```
            NodoGK nodo = new NodoGK("LLAMARHK");
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            nodo.hijos.add(lstpar);
            RESULT = nodo;
          :}
         |tkn_llamarHK identificador:id par_izq par_der
          {:
            NodoGK nodo = new NodoGK("LLAMARHK");
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            nodo.hijos.add(new NodoGK("sinparametros"));
            RESULT = nodo;
          :};


LST_PARAM ::= LST_PARAM:nodo coma PARAM:par
          {:
            nodo.hijos.add(par);
            RESULT = nodo;
          :}
        | PARAM:par
          {:
            NodoGK nodo = new NodoGK("PARAMETROS");
            nodo.hijos.add(par);
            RESULT = nodo;
          :};

PARAM ::= EXPRESION:expr {:RESULT = expr;:};

ACCESO_MET ::= identificador:id par_izq par_der
          {:
            NodoGK nodo = new NodoGK("LLAMAR_MET");
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            nodo.hijos.add(new NodoGK("sinparametros"));
            RESULT = nodo;
          :}
        | identificador:id par_izq LST_PARAM:lstpar par_der
          {:
            NodoGK nodo = new NodoGK("LLAMAR_MET");
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            nodo.hijos.add(lstpar);
            RESULT = nodo;
          :};

ACCESOBJ ::= identificador:id LST_NIVEL:lst
          {:
            NodoGK nodo = new NodoGK("ACCESOBJ");
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            nodo.hijos.add(lst);
```

```
            RESULT = nodo;
          :}
        |tkn_llamar identificador:id LST_NIVEL:lst
          {:
            NodoGK nodo = new NodoGK("ACCESOBJ_LLAMADA");
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            nodo.hijos.add(lst);
            RESULT = nodo;
          :};

LST_NIVEL ::= LST_NIVEL:nodo NIVEL:niv
          {:
            nodo.hijos.add(niv);
            RESULT = nodo;
          :}
         |NIVEL:niv
          {:
            NodoGK nodo = new NodoGK("NIVELES");
            nodo.hijos.add(niv);
            RESULT = nodo;
          :};

NIVEL ::= punto identificador:id
        {:
            NodoGK nodo = new NodoGK(id, idleft, idright);
            RESULT = nodo;
        :}
       |punto ACCESO_MET:nodo {: RESULT = nodo; :}
       |punto identificador:id LST_DIM:dims
          {:
            NodoGK nodo = new NodoGK("AccesoArreglo");
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            nodo.hijos.add(dims);
            RESULT = nodo;
          :};

GRAFICAR ::= tkn_graphikar par_izq EXPRESION:expr1 coma EXPRESION:expr2 par_der int_der
          {:
            NodoGK nodo = new NodoGK("GRAFICAR");
            nodo.hijos.add(expr1);
            nodo.hijos.add(expr2);
            RESULT = nodo;
          :};

IMPRIMIR ::= tkn_imprimir par_izq EXPRESION:expr par_der int_der
          {:
            NodoGK nodo = new NodoGK("IMPRIMIR");
            nodo.hijos.add(expr);
```

```
            RESULT = nodo;
          :}
        |tkn_imprimirK par_izq EXPRESION:expr par_der int_der
          {:
            NodoGK nodo = new NodoGK("IMPRIMIR");
            nodo.hijos.add(expr);
            RESULT = nodo;
          :};

ARREGLOS::= DARR:nodo {: RESULT = nodo; :}
        |AARR:nodo {: RESULT = nodo; :};

DARR::= tkn_var TIPO:tpo identificador:id LST_DIM:dims VISIBILIDAD:vis
        {:
          NodoGK nodo = new NodoGK("DECLARA_ARR");
          nodo.hijos.add(tpo);
          nodo.hijos.add(new NodoGK(id, idleft, idright));
          nodo.hijos.add(dims);
          nodo.hijos.add(vis);
          RESULT = nodo;
        :}
      |tkn_var TIPO:tpo identificador:id LST_DIM:dims VISIBILIDAD:vis igual EXPRESION:asg
        {:
          NodoGK nodo = new NodoGK("DECLARA_ASIG_ARR");
          nodo.hijos.add(tpo);
          nodo.hijos.add(new NodoGK(id, idleft, idright));
          nodo.hijos.add(dims);
          nodo.hijos.add(vis);
          nodo.hijos.add(asg);
          RESULT = nodo;
        :};

LST_DIM::= LST_DIM:nodo DIM:dims
        {:
          nodo.hijos.add(dims);
          RESULT = nodo;
        :}
       |DIM:dims
        {:
          NodoGK nodo = new NodoGK("DIMS");
          nodo.hijos.add(dims);
          RESULT = nodo;
        :};

DIM ::= cor_izq EXPRESION:nodo cor_der {: RESULT = nodo; :};


AARR::= identificador:id LST_DIM:dims igual EXPRESION:expr
```

```
        {:
            NodoGK nodo = new NodoGK("ASIGNA_ARR");
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            nodo.hijos.add(dims);
            nodo.hijos.add(expr);
            RESULT = nodo;
        :};

DATOS::= tkn_vacio tkn_datos:id par_izq par_der llave_izq SENT_DATOS:sent llave_der
        {:
            NodoGK nodo = new NodoGK("DATOS");
            nodo.hijos.add(sent);
            RESULT = nodo;
        :};

SENT_DATOS::= SENT_DATOS:nodo SDATOS:dato
            {:
                nodo.hijos.add(dato);
                RESULT = nodo;
            :}
          | SDATOS:dato
            {:
                NodoGK nodo = new NodoGK("SENTENCIAS");
                nodo.hijos.add(dato);
                RESULT = nodo;
            :};

SDATOS::=  PROCESAR:nodo {: RESULT = nodo; :};

OPCIONES ::= DONDE:nodo {: RESULT = nodo; :}
         | DONDECADA:nodo {: RESULT = nodo; :}
         | DONDETODO:nodo {: RESULT = nodo; :};

COLUMNA::= tkn_columna par_izq EXPRESION:expr par_der
        {:
            NodoGK nodo = new NodoGK("columna");
            nodo.hijos.add(expr);
            RESULT = nodo;
        :};

EXP ::= EXPRESION:nodo {: RESULT = nodo; :}
      |COLUMNA:nodo  {: RESULT = nodo; :};

PROCESAR ::= tkn_procesar igual MET_ESPECIALES:met int_der OPCIONES:opc
            {:
                NodoGK nodo = new NodoGK("PROCESAR");
                nodo.hijos.add(met);
                nodo.hijos.add(opc);
```

```
            RESULT = nodo;
        :};

MET_ESPECIALES::= HASKELL:nodo {: RESULT = nodo; :}
        | NORMAL:nodo {: RESULT = nodo; :}
        | EXP:nodo {: RESULT = nodo; :};

HASKELL ::= tkn_llamarHK identificador:id par_izq LST_COLUMNA:cols par_der
        {:
            NodoGK nodo = new NodoGK("LLAMADA_HK_DATOS");
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            nodo.hijos.add(cols);
            RESULT = nodo;
        :};

NORMAL ::= tkn_llamar identificador:id par_izq LST_COLUMNA:cols par_der
        {:
            NodoGK nodo = new NodoGK("LLAMADA_MET_DATOS");
            nodo.hijos.add(new NodoGK(id, idleft, idright));
            nodo.hijos.add(cols);
            RESULT = nodo;

        :};

LST_COLUMNA ::= LST_COLUMNA:cols coma COLUMNA:col
            {:
                cols.hijos.add(col);
                RESULT = cols;
            :}
          |COLUMNA:col
            {:
                NodoGK nodo = new NodoGK("COLUMNAS");
                nodo.hijos.add(col);
                RESULT = nodo;
            :};

DONDE ::= tkn_donde par_izq EXP:exp par_der igual EXPRESION:expr int_der
        {:
            NodoGK nodo = new NodoGK("DONDE");
            nodo.hijos.add(exp);
            nodo.hijos.add(expr);
            RESULT = nodo;
        :};

DONDECADA ::= tkn_dondecada par_izq EXP:exp par_der int_der
        {:
            NodoGK nodo = new NodoGK("DONDECADA");
            nodo.hijos.add(exp);
```

```
            RESULT = nodo;
          :};

DONDETODO ::= tkn_dondetodo par_izq EXP:exp par_der int_der
          {:
            NodoGK nodo = new NodoGK("DONDETODO");
            nodo.hijos.add(exp);
            RESULT = nodo;
          :};

EXPRESION ::= EXPRESION:expr1 tkn_or EXPRESION:expr2
          {:
            NodoGK nodo = new NodoGK("or");
            nodo.hijos.add(expr1);
            nodo.hijos.add(expr2);
            RESULT = nodo;
          :}
         |EXPRESION:expr1 tkn_xor EXPRESION:expr2
          {:
            NodoGK nodo = new NodoGK("xor");
            nodo.hijos.add(expr1);
            nodo.hijos.add(expr2);
            RESULT = nodo;
          :}
         |EXPRESION:expr1 tkn_and EXPRESION:expr2
          {:
            NodoGK nodo = new NodoGK("and");
            nodo.hijos.add(expr1);
            nodo.hijos.add(expr2);
            RESULT = nodo;
          :}
         |tkn_not EXPRESION:expr1
          {:
            NodoGK nodo = new NodoGK("not");
            nodo.hijos.add(expr1);
            RESULT = nodo;
          :}
         |EXPRESION:expr1 igualacion EXPRESION:expr2
          {:
            NodoGK nodo = new NodoGK("igualacion");
            nodo.hijos.add(expr1);
            nodo.hijos.add(expr2);
            RESULT = nodo;
          :}
         |EXPRESION:expr1 diferente EXPRESION:expr2
          {:
            NodoGK nodo = new NodoGK("diferente");
            nodo.hijos.add(expr1);
```

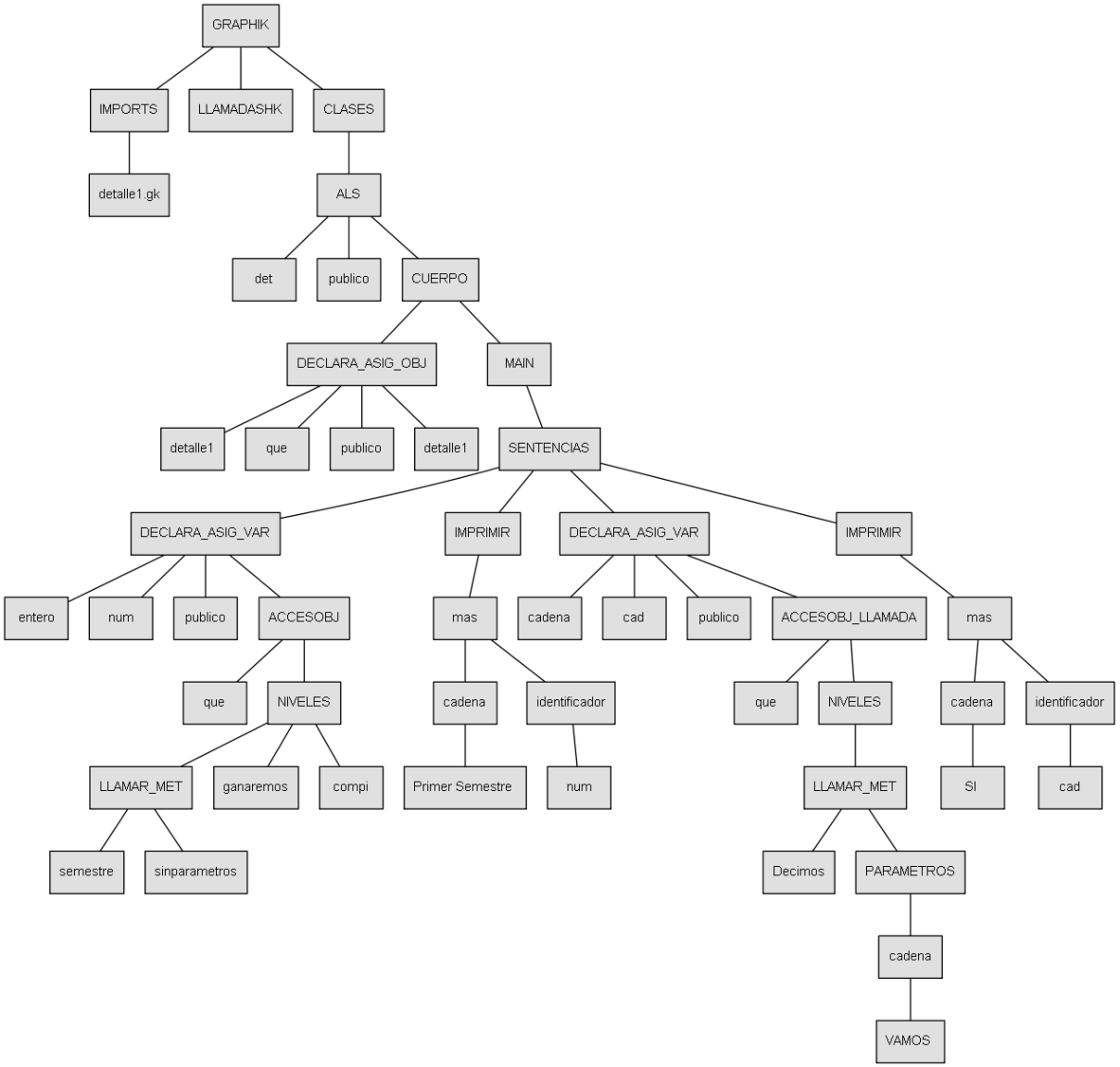
```
    nodo.hijos.add(expr2);
    RESULT = nodo;
  :}
|EXPRESION:expr1 menor EXPRESION:expr2
  {:
    NodoGK nodo = new NodoGK("menor");
    nodo.hijos.add(expr1);
    nodo.hijos.add(expr2);
    RESULT = nodo;
  :}
|EXPRESION:expr1 mayor EXPRESION:expr2
  {:
    NodoGK nodo = new NodoGK("mayor");
    nodo.hijos.add(expr1);
    nodo.hijos.add(expr2);
    RESULT = nodo;
  :}
|EXPRESION:expr1 menor_igual EXPRESION:expr2
  {:
    NodoGK nodo = new NodoGK("menorI");
    nodo.hijos.add(expr1);
    nodo.hijos.add(expr2);
    RESULT = nodo;
  :}
|EXPRESION:expr1 mayor_igual EXPRESION:expr2
  {:
    NodoGK nodo = new NodoGK("mayorI");
    nodo.hijos.add(expr1);
    nodo.hijos.add(expr2);
    RESULT = nodo;
  :}
|EXPRESION:expr1 mas EXPRESION:expr2
  {:
    NodoGK nodo = new NodoGK("mas");
    nodo.hijos.add(expr1);
    nodo.hijos.add(expr2);
    RESULT = nodo;
  :}
|EXPRESION:expr1 menos EXPRESION:expr2
  {:
    NodoGK nodo = new NodoGK("menos");
    nodo.hijos.add(expr1);
    nodo.hijos.add(expr2);
    RESULT = nodo;
  :}
|EXPRESION:expr1 por EXPRESION:expr2
  {:
    NodoGK nodo = new NodoGK("por");
```

```
    nodo.hijos.add(expr1);
    nodo.hijos.add(expr2);
    RESULT = nodo;
  :}
|EXPRESION:expr1 div EXPRESION:expr2
  {:
    NodoGK nodo = new NodoGK("div");
    nodo.hijos.add(expr1);
    nodo.hijos.add(expr2);
    RESULT = nodo;
  :}
|EXPRESION:expr1 potencia EXPRESION:expr2
  {:
    NodoGK nodo = new NodoGK("pot");
    nodo.hijos.add(expr1);
    nodo.hijos.add(expr2);
    RESULT = nodo;
  :}
|numero:entero
  {:
    NodoGK nodo = new NodoGK("entero");
    nodo.hijos.add(new NodoGK(entero, enteroleft, enteroright));
    RESULT = nodo;
  :}
|texto: cadena
  {:
    NodoGK nodo = new NodoGK("cadena");
    nodo.hijos.add(new NodoGK(cadena, cadenaleft, cadenaright));
    RESULT = nodo;
  :}
|identificador:id
  {:
    NodoGK nodo = new NodoGK("identificador");
    nodo.hijos.add(new NodoGK(id, idleft, idright));
    RESULT = nodo;
  :}
|identificador:id LST_DIM:dims
  {:
    NodoGK nodo = new NodoGK("AccesoArreglo");
    nodo.hijos.add(new NodoGK(id, idleft, idright));
    nodo.hijos.add(dims);
    RESULT = nodo;
  :}
|decimal:dec
  {:
    NodoGK nodo = new NodoGK("decimal");
    nodo.hijos.add(new NodoGK(dec, decleft, decright));
    RESULT = nodo;
```

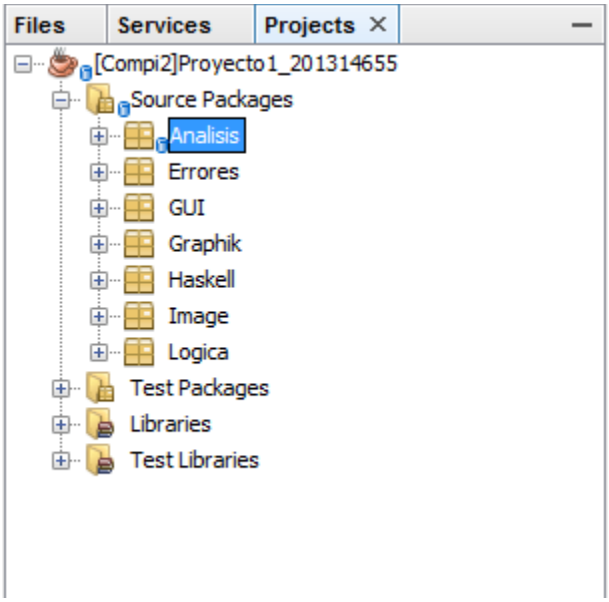
```
  :}
|tkn_verdadero:verdadero
  {:
     NodoGK nodo = new NodoGK("bool");
     nodo.hijos.add(new NodoGK(verdadero, verdaderoleft, verdaderoright));
     RESULT = nodo;
  :}
|tkn_falso:falso
  {:
     NodoGK nodo = new NodoGK("bool");
     nodo.hijos.add(new NodoGK(falso, falsoleft, falsoright));
     RESULT = nodo;
  :}
|caracter:carac
  {:
     NodoGK nodo = new NodoGK("caracter");
     nodo.hijos.add(new NodoGK(carac));
     RESULT=nodo;
  :}
|menos EXPRESION:expr
  {:
     NodoGK nodo = new NodoGK("menos");
     nodo.hijos.add(expr);
     RESULT = nodo;
  :}
|par_izq EXPRESION:nodo par_der
  {:
     RESULT = nodo;
  :}
|LLAMADA_MET:nodo
  {:
     RESULT = nodo;
  :}
|LLAMADA_HK:nodo
  {:
     RESULT = nodo;
  :}
|EXPRESION:exp aumento
  {:
     NodoGK nodo = new NodoGK("aumento");
     nodo.hijos.add(exp);
     RESULT = nodo;
  :}
|EXPRESION:exp decremento
  {:
     NodoGK nodo = new NodoGK("decremento");
     nodo.hijos.add(exp);
     RESULT = nodo;
```

```
            :}
          |ACCESOBJ:nodo
            {:
               RESULT = nodo;
            :}
          |ACCESO_MET:nodo
            {:
               RESULT = nodo;
            :}
          |llave_izq LST_DARR:nodo llave_der
            {:
               RESULT = nodo;
            :};


LST_DARR ::= LST_DARR:nodo coma EXPRESION:dims
            {:
               nodo.hijos.add(dims);
               RESULT = nodo;
            :}
          |EXPRESION:exp
            {:
               NodoGK nodo = new NodoGK("ARREGLO");
               nodo.hijos.add(exp);
               RESULT = nodo;

            :};
```

Arbol Generado para Graphik



Paquetes que se utilizaron



Clases que se utilizaron

Paquete Analisis	Descripcion
CompilarC.bat	Archivo utilizado con parámetros del cmd que compilan el archivo parserConsole.cup
CompilarCSV.bat	Archivo utilizado con parámetros del cmd que compilan el archivo parserCSV.cup
CompilarG.bat	Archivo utilizado con parámetros del cmd que compilan el archivo parserGraphik.cup
CompilarH.bat	Archivo utilizado con parámetros del cmd que compilan el archivo parserHaskell.cup

lexicoCSV.java	Clase que me genera el archivo de scannerCSV.jflex donde esta implementado para usarse en java.
lexicoConsole.java	Clase que me genera el archivo de scannerConsole.jflex donde esta implementado para usarse en java.
lexicoHaskell.java	Clase que me genera el archivo de scannerHaskell.jflex donde esta implementado para usarse en java.
parserCSV.cup	Archivo donde esta implementada la gramatica con terminales y no terminales para reconocer la carga del archivo CSV
parserConsole.cup	Archivo donde esta implementada la gramatica con terminales y no terminales para reconocer los comandos de consola.
parserGraphik.cup	Archivo donde esta implementada la gramatica con terminales y no terminales para reconocer los comandos de graphik.
parserHaskell.cup	Archivo donde esta implementada la gramatica con terminales y no terminales para reconocer los comandos de haskell.
scannerCSV.jflex	Archivo donde están implementadas las palabras reservadas del archivo CSV.
scannerConsole.jflex	Archivo donde están implementadas las palabras reservadas de la Consola.
scannerGraphik.jflex	Archivo donde están implementadas las palabras reservadas de Graphik.
scannerHaskell.jflex	Archivo donde están implementadas las palabras reservadas de Haskell.
sintacticoCSV.java	Clase que genera cup para el uso de la gramatica que reconoce CSV desde java.

sintacticoConsole.java	Clase que genera cup para el uso de la gramatica que reconoce Consola desde java.
sintacticoGraphik.java	Clase que genera cup para el uso de la gramatica que reconoce Graphik desde java.
sintacticoHaskell.java	Clase que genera cup para el uso de la gramatica que reconoce Haskell desde java.
Sym.java	Clase que genera java para guardar los símbolos de la clase Graphik.
symCSV.java	Clase que genera java para guardar los símbolos de la clase CSV.
symH.java	Clase que genera java para guardar los símbolos de la clase Haskell.

Paquete Errores	Descripcion
Errores.java	Clase donde se tienen los metodos que generar nuevos errores y los guardan en una lista estatica que prevalece durante la ejecución.
NError.java	Clase constructora de errores que guarda la línea, columna y descripcion de los errores.

Paquete GUI	Descripcion
GraficarTreeHaskell.java	Clase que genera la interfaz para el árbol de expresiones de haskell.
GraphikSyntax.java	Clase que genero mediante TokenMaker para hacer el pintado de las palabras.
HaskellSyntax.java	Clase que genero mediante TokenMaker para hacer el pintado de las palabras.

Login.java	Clase que utilizo para crear el login de los usuarios al servidor.
Principal.java	Clase que contiene todos los metodos de la pantalla principal.
Proyecto1_201314655.java	Clase que contiene el main que llama a Principal que es mi pantalla por defecto.
TablaResultados.java	Clase utilizada para generar Tablas de resultados para las operaciones Procesar, Donde, Donde Cada y Donde Todo.
Tabs.java	Clase que extiende de un JPanel, la cual sirve para el manejo de multiples pesta;as
TextLineNumber.java	Clase utilizada para mostrar los números de línea en un JTextArea.

Paquete Graphik	Descripcion
ClaseGK.java	Clase utilizada para guardar todas mis clases generadas por el usuario.
ClonarCosas.java	Clase utilizada para clonar los metodos, funciones y clases para la instanciación del objeto.
EjecutarGK.java	Clase donde ejecuto todo el código de Graphik.
LlamadasHK.java	Clase utilizada para la integración de llamadas a metodos haskell desde graphik.
MetodoGK.java	Clase utilizada para guardar todos mis metodos generados por el usuario.
NodoGK.java	Clase utilizada para generar el Arbol AST para mi lenguaje Graphik.
RecorridoAST.java	Clase utilizada para guardar todas las clases, variables globales, metodos y parámetros.

Resultado.java	Clase utilizada para almacenar los valores generados para las variables de Graphik.
SimboloGK.java	Clase utilizada para guardar las variables generadas por el usuario.
TablaSimbolosGK.java	Clase estatica que mantiene en memoria la tabla de símbolos de Graphik.

Paquete Haskell	Descripcion
EjecucionHK.java	Clase donde ejecuto todo el código de Haskell.
NodoHK.java	Clase utilizada para generar el Arbol AST para mi lenguaje Haskell.
NodoTabla.java	Clase utilizada para guardar los metodos y variables generados por Haskell.
RecorridoHK.java	Clase utilizada para guardar todas las clases, variables globales, metodos y parámetros.
TablaSimbolos.java	Clase estatica que mantiene en memoria la tabla de símbolos de Haskell.
Value.java	Clase utilizada para almacenar los valores generados para las variables de Haskell.

Paquete Logica	Descripcion
CargarCSV.java	Clase utilizada para todas las operaciones con los archivos CSV.
Constantes.java	Clase utilizada para guardar las constantes utilizadas en el manejo de errores.
DynamicTree.java	Clase utilizada para generar el Arbol de Funciones de Haskell.
Graficar.java	Clase utilizada para implementar todas las funciones de graficar.

GraphvizJava.java	Clase utilizada para Graficar el Arbol AST con Graphviz.
ManejoArchivos.java	Clase utilizada para administrar las operaciones de archivos, abrir, guardar, guardar como, etc.

Librerias Utilizadas

Nombre	Descripcion
Java Cup	Librería que permite compilar archivos cup.
JFlex	Librería que permite compilar archivos jflex.
JCommon	Librería utilizada para implementar series en las graficas con la ayuda de la librería JFreeChart.
JFreeChart	Librería utilizada para generar graficas de diferentes tipos.
Rsyntaxtextarea	Librería utilizada para generar un TextArea en el cual se pueden implementar funciones de pintar y de autocompletado.
LibThrift	Librería utilizada para la conexión de cliente y servidor de diferentes lenguajes de programación.

Requerimientos del Sistema

Esta aplicación esta desarrollada en lenguaje java en el IDE de NetBeans, implementando con JFLEX y CUP, los requerimientos para poder compilar y editar esta aplicación consta de las siguientes implementaciones.

NetBeans IDE 8.2

Este IDE no necesita de mucho espacio en el disco duro ocupa aproximadamente 950 MB la instalación completa, en este caso como utilizamos el sistema operativo de Windows para descargar e instalar en nuestro sistema operativo podemos descargar el ejecutable desde la pagina oficial:

https://netbeans.org/downloads/start.html?platform=windows&lang=en&option=all

JFlex y Cup

Estas herramientas son ligeras no piden restricciones de sistema operativo o mínimo de memoria RAM, en este caso como utilizamos el sistema operativo de Linux para descargar e instalar en nuestro sistema operativo basta con tan solo escribir la siguiente línea de comando en la terminal:

http://www.jflex.de/download.html

https://www.cs.princeton.edu/~appel/modern/java/CUP/