Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela Ciencias y Sistemas
Laboratorio OLC2
Aux. Javier Navarro

MANUAL TECNICO

Manuel Antonio Fuentes Fuentes
2012-13580
14 de mayo del 2019

DESCRIPCION DE APLICACIÓN

El almacenamiento en la nube es un modelo de almacenamiento de datos basado en redes de computadoras donde los datos están alojados en espacios de almacenamiento virtualizados que últimamente está tomando mucho auge en esta era tecnológica ya que permite a todos los usuarios tener su información en cualquier parte o a los desarrolladores poder contar con diversas tecnologías pagando únicamente por sus servicios. AWS es uno de los principales proveedores de servicios de Cloud Computing, dando una gran diversidad de servicios, entre ellos, bases de datos, herramientas de desarrollo, robótica, IOT, entre muchas más. Algunos de estos servicios se encuentran como free Tier, capas gratuitas, entre los más conocidos de estos esta DynamoDB, que es una base de datos no relacional que provee a los usuarios un uso gratuito de hasta 25 GB. Teniendo esto en cuenta se desea crear un compilador que genere reportes los cuales no se quieren perder conforme el tiempo, para ello se propone hacer uso de DynamoDB como base de datos, en la cual se almacenarán los distintos reportes generados.

Compilador como servicio o CAAS por sus siglas en inglés (Compiler as a service), es un software de compilación en línea, cuya implantación será hecha de forma local por los estudiantes. CAAS llevará un control de todos los procesos del compilador en una base de datos no relacional haciendo uso de los componentes de la nube, para este caso Amazon web services. CAAS será un software donde el compilador estará alojado en un entorno local desarrollado por los estudiantes, este compilador será el encargado de pasar de código fuente a código intermedio y ejecutar este código intermedio.

El código intermedio generado por el compilador será código de tres direcciones, se le llama de esa forma porque solo permite referenciar a 3 direcciones de memoria al mismo tiempo, para lograr esto se descompone cada instrucción a una más sencilla de ejecutar que cumpla con el paradigma de tres direcciones. Además, la aplicación CAAS tendrá la capacidad de generar una serie de reportes que serán usados para diagnosticar errores o ver detalles del proceso de compilación.

- **Reporte de errores:** dará un reporte de todos los errores generados durante la compilación o ejecución y los guardará en DynamoDB.
- **Reporte de tabla de símbolos:** dará un reporte de la tabla de símbolos utilizada durante la compilación y los guardará en DynamoDB.
- **Generación de AST:** dará un reporte con el AST del lenguaje fuente y lo guardará en DynamoDB.
- **Reporte de optimización:** dará un reporte de la optimización del código de tres direcciones generado.

REQUERIMIENTOS MINIMOS

- Procesador: Intel Core i7-7700HQ CPU @ 2.80GHz
- Memoria Ram: 8 GB
- Sistema Operativo: Windows 10 64 Bits
- NodeJs Express 8.12.0
- Bootstrap 4.3.1
- Codemirror 5.45.0
- Jison
- Graphviz 2.38
- Dynamodb SDK 2.7.16

DIAGRAMA DE CLASES

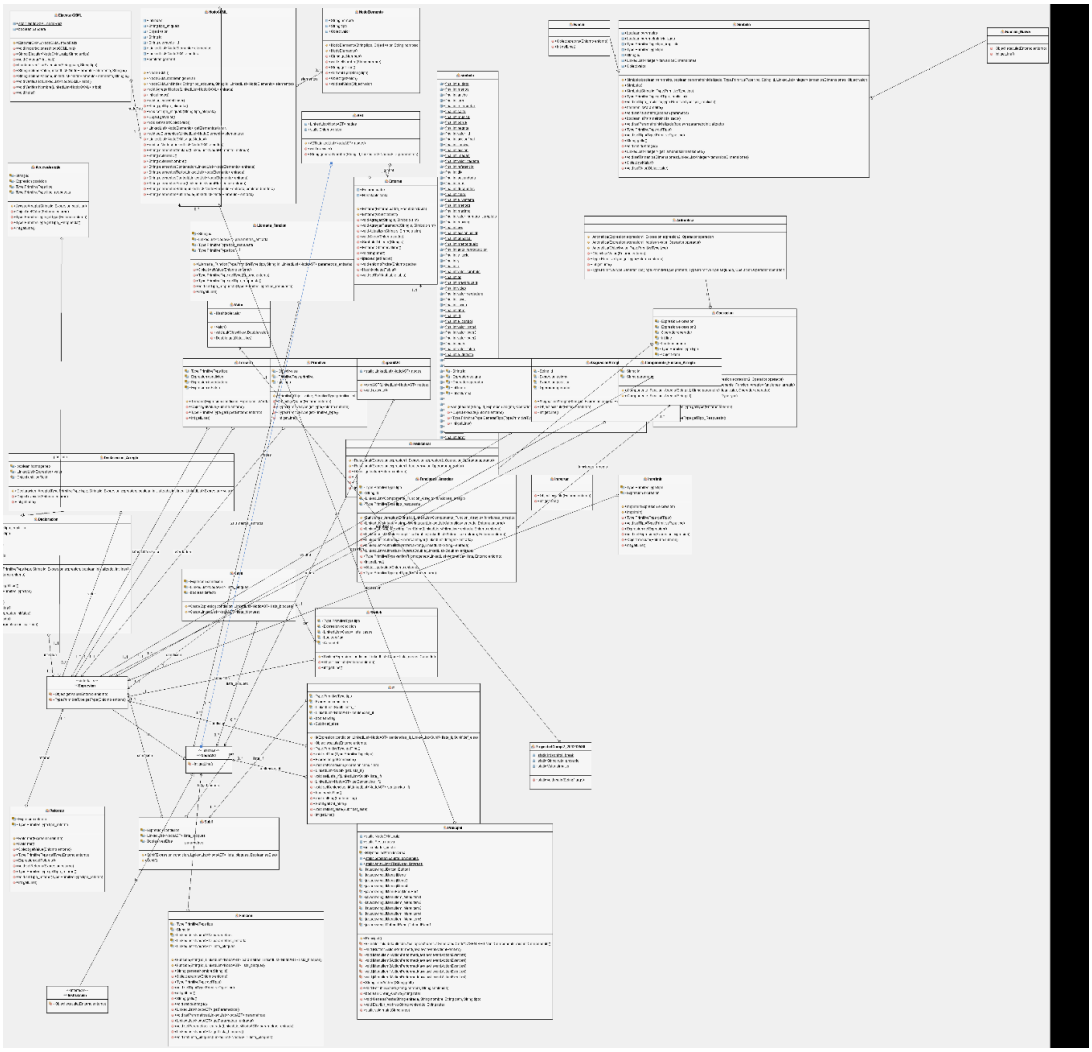
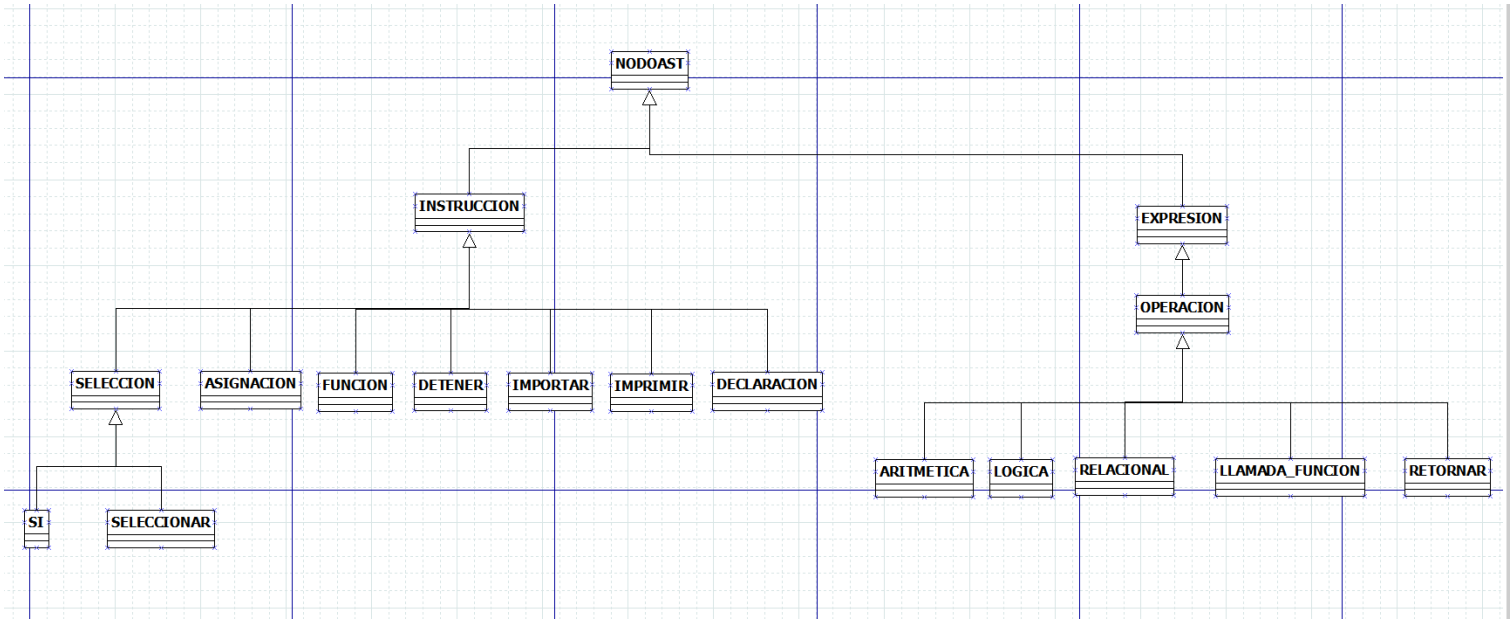


DIAGRAMA DE PAQUETES O HERENCIA



DESCRIPCION DE HERRAMIENTAS UTILIZADAS

- **Node Js Express:**
  Version: 8.12.0

Express.js , o simplemente Express , es un marco de aplicación web para Node.js , lanzado como software libre y de código abierto bajo la Licencia MIT . Está diseñado para construir aplicaciones web y APIs . Se ha llamado el marco de servidor estándar de facto para Node.js.

Es un entorno que trabaja en tiempo de ejecución, de código abierto, multi-plataforma, que permite a los desarrolladores crear toda clase de herramientas de lado servidor y aplicaciones en JavaScript. La ejecución en tiempo real está pensada para usarse fuera del contexto de un explorador web (es decir, ejecutarse directamente en una computadora o sistema operativo de servidor). Como tal, el entorno omite las APIs de JavaScript específicas del explorador web y añade soporte para APIs de sistema operativo más tradicionales que incluyen HTTP y bibliotecas de sistemas de ficheros.

- **Bootstrap:**
  Version: 4.3.1

Bootstrap es un conjunto de herramientas de código abierto para desarrollar con HTML, CSS y JS. Realice rápidamente un prototipo de sus ideas o construya su aplicación completa con nuestras variables y mixins de Sass, sistema de cuadrícula sensible, extensos componentes precompilados y potentes complementos creados en jQuery.

Bootstrap es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. A diferencia de muchos frameworks web, solo se ocupa del desarrollo front-end.

- **Codemirror:**
  Version: 5.45.0

CodeMirror es un editor de texto versátil implementado en JavaScript para el navegador. Está especializado en la edición de código y viene con varios modos de idioma y complementos que implementan una funcionalidad de edición más avanzada.
Una API de programación enriquecida y un sistema de temática CSS están disponibles para personalizar CodeMirror para que se adapte a su aplicación, y extenderlo con una nueva funcionalidad.

- **Jison:**

Jison es esencialmente un clon del generador de analizadores Bison (por lo tanto, Yacc,) pero en JavaScript. Incluye su propio analizador léxico modelado según Flex.

- **Graphviz:**
  Version: 2.38

Graphviz es un programa de visualización gráfica de fuente abierta. La visualización de gráficos es una forma de representar información estructural como diagramas de gráficos y redes abstractas. Tiene importantes aplicaciones en redes, bioinformática, ingeniería de software, diseño de bases de datos y web, aprendizaje automático y en interfaces visuales para otros dominios técnicos.

- **Dynamodb:**
  Version: 2.17.16

Amazon DynamoDB es un servicio de base de datos noSQL ofrecido por Amazon como parte de Amazon Web Services. DynamoDB expone un modelo de datos similar y deriva su nombre de Dynamo (un sistema de almacenamiento interno utilizado inicialmente para su propio sitio web), pero tiene una implementación subyacente diferente. Dynamo tenía un diseño multimaestro que requería que el cliente resolviera conflictos de versiones y DynamoDB usa replicación síncrona en múltiples centros de datos para una alta durabilidad y disponibilidad.

DESCRIPCION DE METODOS Y CLASES PRINCIPALES

Metodos Principales:

- **Ejecutar:** Encargado de la parte lógica de la aplicación donde se analizan los archivos y se realizan las importaciones, además que lleva el control de la interfaz de la aplicación.
- **Compilar:** este metodo nos permite compilar tanto los archivo coline como los archivos 3d.

- **AgregarTablaSimbolos:** este método nos permite agregar los valores de los simbolos a la tabla de simbolos, por medio de una clave que será única independiente mente de donde se encuentre alojada.
- **Execute:** este método se encuentra en las instrucciones que son declaraciones, asignaciones etc. Este va a realizar la acción indicada en su cuerpo dependiendo del tipo de entorno en el que se encuentre.
- **GetValue:** este método se encuentra en todas las instancias que tenga de retornar un valor, como lo son las operaciones aritmeticas, logicas, relacionales, etc.
- **GetTipe:** retorna el tipo de una clase, para poder validar los tipos de asignaciones o declaraciones
- **ScanData:** nos permite leer el contenido que esta almacenado en dynamodb.

Clases Principales:

- **Index.js:** este tiene la lógica del servidor es donde se llegan a realizar las peticiones que requiere el servidor.
- **Conexiondinamo.js:** hace la conexión directa con el servicio de aws dynamodb con esta clase podemos crear tablas, agregar registros, etc.
- **Ejecutar.js:** esta clase lleva la lógica de toda la ejecución del compilador.
- **AST.js:** nos permite definir las características del flujo del árbol ast a ejecutar.
- **Declaracionclase.js:** tiene todas las definiciones de las clases y permite armar el árbol AST y la tabla de simbolos.
- **Metodo.js:** Tiene la visión general del control de todos los elementos que pueden tener al momento de la ejecución.
- **Entorno.js:** este tiene la definición de los entornos y de los símbolos que se utilizan en la ejecución.

GRAMATICA COLINE

```
/* description: Parses and executes mathematical expressions. */

/* lexical grammar */
%lex
%%

\s+                   /* skip whitespace */
```

```
"variable"         return 'VARIABLE'
"metodo"           return 'METODO'
"constructor"      return 'CONSTRUCTOR'
"int"              return 'INT'
"string"           return 'STRING'
"double"           return 'DOUBLE'
"char"             return 'CHAR'
"boolean"          return 'BOOLEAN'
"void"             return 'VOID'
"public"           return 'PUBLIC'
"static"           return 'STATIC'
"final"            return 'FINAL'
"private"          return 'PRIVATE'
"protected"        return 'PROTECTED'
"abstract"         return 'ABSTRACT'
"new"              return 'NEW'
"class"            return 'CLASS'
"extends"          return 'EXTENDS'
"import"           return 'IMPORT'
"println"          return 'PRINTLN'
"print"            return 'PRINT'
"new"              return 'NEW'
"this"             return 'THIS'
"return"           return 'RETORNO'
"null"             return 'NULO'
//sentencias control
"if"               return 'IF'
"else"             return 'ELSE'
"switch"           return 'SWITCH'
"case"             return 'CASE'
"do"               return 'DO'
"while"            return 'WHILE'
"for"              return 'FOR'
"default"          return 'DEFAULT'
"break"            return 'BREAK'
//casteos
//simbolos del lenguaje
"."                return '.'
"{"                return '{'
"}"                return '}'
","                return ','
"]"                return ']'
"["                return '['
":"                return ':'
";"                return ';'
"=="               return '=='
"="                return '='
//aritmeticas
"*"                return '*'
"/"                return '/'
"++"               return '++'
"--"               return '--'
```

```
"-"                 return '-'
"+"                 return '+'
"pow"               return 'POW'
"^"                 return '^'
"!="                return '!='
"%"                 return '%'
"?"                 return '?'
//relacionales
"<="                return '<='
">="                return '>='
">"                 return '>'
"<"                 return '<'
//operaciones logicas
"&&"                return '&&'
"||"                return '||'
"!"                 return '!'
"("                 return '('
")"                 return ')'
"true"              return 'TRUE'
"false"             return 'FALSE'
[0-9]+("."[0-9]+)\b   return 'DECIMAL'
[0-9]+              return 'NUMBER'

"\""[^\"\n]*"\""      return 'STRING'
"'"([a-zA-Z]|[0-9])[^'\n]*"'" return 'CHAR'
([a-zA-Z]|["_"])([a-zA-Z]|[0-9]|["_"])* return 'ID'
<<EOF>>             return 'EOF'
.                   {lista_errores.push(new Errores("LEXICO","TOKEN NO RECONOCIDO
"+yytext,(yylineno+1),yyleng));}

/lex

/* operator associations and precedence */

%left '?'
%nonassoc '++' '--'
%left '+' '-'
%left '*' '/' '%'
%left 'pow'
%right '!'
%left '&&' '||' '^'
%left '==' '!=' '>' '>=' '<' '<='
%left '(' ')'

%start expressions

%% /* language grammar */

expressions
   : e EOF
      { return $1; }
   ;
```

```
e: sentencias_globales{$$=$1;};
sentencias_globales: declaraciones_import sentencias_generales{
                                    $$=$1;
                                    for(var i=0;i<$2.length;i++){
                                      $$.push($2[i]);
                                    }
                                    }
            | sentencias_generales{$$=$1;};

declaraciones_import: declaraciones_import declaracion_import{
                                      $$=$1;
                                      $$.push($2);
                                      }
            | declaracion_import{
                        $$=[];
                        $$.push($1);
                        };

declaracion_import: IMPORT STRING ';'{$$=new Importacion($2.replace("\"",""));}  ;

sentencias_generales: declaraciones_clase{$$=$1;};



declaraciones_clase: declaraciones_clase declaracion_clase{
                                    $$=$1;
                                    $$.push($2);
                                    }
           | declaracion_clase{
                        $$=[];
                        $$.push($1);
                        };

declaracion_clase: modificadores_clase CLASS ID '{' cuerpo_clase '}'{
                                        $$=new Declaracionclase($3,$1,null,$5);
                                        }
          | modificadores_clase CLASS ID EXTENDS ID '{' cuerpo_clase '}'{
                                              $$=new Declaracionclase($3,$1,$5,$7);
                                              };
modificadores_clase:{$$=[];}
            | modificadores_clase2{$$=$1;};
modificadores_clase2: modificadores_clase2 modificador_clase{
                                    $$=$1;
                                    $$.push($2);
                                    }
            | modificador_clase{
                        $$=[];
                        $$.push($1);
                        };

modificador_clase: PUBLIC{$$=Visibilidad.PUBLIC;}
          | PROTECTED{$$=Visibilidad.PROTECTED;}
          | PRIVATE{$$=Visibilidad.PRIVATE;}
```

```
        | ABSTRACT{$$=Visibilidad.ABSTRACT;}
        | STATIC{$$=Visibilidad.STATIC;}
        | FINAL{$$=Visibilidad.FINAL;};


cuerpo_clase: cuerpo_clase cuerpo_clase_sentencias{
                        $$=$1;
                        for(var i=0;i<$2.length;i++){
                           $$.push($2[i]);
                        }
                        }
      | cuerpo_clase_sentencias{
                    $$=$1;
                    };

cuerpo_clase_sentencias: modificadores declaracion_metodos{
                                $$=[];
                                $2.modificadores=$1;
                                $$.push($2);
                              }
            | declaracion_metodos{
                        $$=[];
                        $$.push($1);
            }
            | declaracion_variables{$$=$1;}
            | declaracion_arreglos{
                        $$=[];
                        $$.push($1);
                     }
            | declaracion_clase_interna{
                        $$=[];
                        $$.push($1);
                        };

declaracion_clase_interna: modificadores CLASS ID '{' cuerpo_clase_interna '}'{$$=new
Declaracionclase($3,$1,null,$5);}
            | CLASS ID '{' cuerpo_clase_interna '}'{$$=new Declaracionclase($2,[],null,$4);};

cuerpo_clase_interna: cuerpo_clase_interna cuerpo_interna{
                                $$=$1;
                                for(var i=0;i<$2.length;i++){
                                   $$.push($2[i]);
                                }
                                }
          | cuerpo_interna{$$=$1;};
cuerpo_interna: modificadores declaracion_metodos{
                        $$=[];
                        $2.modificadores=$1;
                        $$.push($2);
                        }
        | declaracion_metodos{
                    $$=[];
                    $$.push($1);
```

```
                    }
        | declaracion_variables{
                    $$=$1;
                    }
        | declaracion_arreglos{
                $$=[];
                $$.push($1);
                };

declaracion_arreglos: modificadores sentencia_arreglo ';'{
                                $$=$2;
                                $$.modificadores=$1;
                              }
            | sentencia_arreglo ';'{
                        $$=$1;
                        };

declaracion_variables: modificadores variables ';'{
                            for(var i=0;i<$2.length;i++){
                               $2[i].modificadores=$1;
                            }
                            $$=$2;
                           }
            | variables ';'{
                    $$=$1;
                   };

variables: tipo lista_id {
                    //DECLARACION DE UNA VARIABLE
                    for(var i=0;i<$2.length;i++){
                       $2[i].tipo=$1;
                    }
                    $$=$2;
                  }
      | ID lista_id {
                  //DECLARACION DE UN OBJETO
                  for(var i=0;i<$2.length;i++){
                     $2[i].tipo=$1;
                  }
                  $$=$2;
                  }
      | tipo lista_id '=' exp {
                      //DECLARACION ASIGNACION DE UNA VARIABLE
                      for(var i=0;i<$2.length;i++){
                         $2[i].tipo=$1;
                      }
                      $2[($2.length-1)].iniValue=$4;
                      $$=$2;
                      }
      | ID lista_id '=' NEW ID '(' ')'{
                          for(var i=0;i<$2.length;i++){
                                  $2[i].tipo=$1;
                          }
```

```
                        $2[($2.length-1)].iniValue=$4;
                        $2[($2.length-1)].lista_valores=[];
                        $2[($2.length-1)].inicializado=true;
                        $$=$2;
                       }
    | ID lista_id '=' NEW ID '(' lista_valores ')'{
                              for(var i=0;i<$2.length;i++){
                                 $2[i].tipo=$1;
                              }
                              $2[($2.length-1)].iniValue=$4;
                              $2[($2.length-1)].lista_valores=$7;
                              $2[($2.length-1)].inicializado=true;
                              $$=$2;
                             }
    | ID lista_id '=' ID '.' ID{
                      for(var i=0;i<$2.length;i++){
                         $2[i].tipo=$1;
                      }
                      $2[($2.length-1)].iniValue=new AccesoObjetos($4,new
Aritmetica(null,null,false,$6,null,Type.ID,0,0));
                      $2[($2.length-1)].inicializado=true;
                      $$=$2;
                      }

    | ID lista_id '=' ID{
                    for(var i=0;i<$2.length;i++){
                      $2[i].tipo=$1;
                    }
                    $2[($2.length-1)].iniValue=new Aritmetica(null,null,false,$4,null,Type.ID,0,0);
                    $2[($2.length-1)].lista_valores=[];
                    $2[($2.length-1)].inicializado=true;
                    $$=$2;
                   }
    | ID lista_id '=' sentencia_llamada{
                        for(var i=0;i<$2.length;i++){
                          $2[i].tipo=$1;
                        }
                        $2[($2.length-1)].iniValue=$4;
                        $2[($2.length-1)].lista_valores=[];
                        $2[($2.length-1)].inicializado=true;
                        $$=$2;
                        }
    | ID lista_id '='  ID lista_dd{
                      for(var i=0;i<$2.length;i++){
                        $2[i].tipo=$1;
                      }

                      $2[($2.length-1)].iniValue=new Aritmetica(null,null,false,$4,null,"ARRAY",0,0);
                      $2[($2.length-1)].iniValue.lista_dimensiones=$5;
                      $2[($2.length-1)].lista_valores=[];
                      $2[($2.length-1)].inicializado=true;
                      $$=$2;
                      };
```

```
lista_valores: lista_valores ',' exp{
                        $$=$1;
                        $$.push($3);
                        }
            | exp{
              $$=[];
              $$.push($1);
            };

lista_id: lista_id ',' ID{
                $$=$1;
                $$.push(new Declaracion(yytext,PrimitiveType.NULO,null,null,0,0,0));
                }
        | ID{
          $$=[];
          $$.push(new Declaracion(yytext,PrimitiveType.NULO,null,[],0,0,0));
          };

declaracion_metodos: tipo ID '(' lista_parametros ')' '{' cuerpo_metodo '}'{
                                            //UN METODO NORMAL
                                            $$=new Metodo($2,$1,$7,$4);
                                            }
              | tipo ID '(' lista_parametros ')' ';'{
                                  //UN METODO ABSTRACTO NORMAL
                                  //id,tipo,nodos,parametros
                                  $$=new Metodo($2,$1,[],$4);
                                  }
              | ID ID '(' lista_parametros ')' ';'{
                                  //UN METODO ABSTRACTO QUE DEVUELVE UN OBJETO
                                  $$=new Metodo($2,$1,[],$4);
                                  }
              | ID ID '(' lista_parametros ')' '{' cuerpo_metodo '}'{
                                            //METODO QUE DEVUELVE UN OBJETO
                                            $$=new Metodo($2,$1,$7,$4);
                                        }
              | ID lista_d ID '(' lista_parametros ')' '{' cuerpo_metodo '}'{
                                              //METODO QUE DEVUELVE UN OBJETO
                                              $$=new Metodo($3,$1,$8,$5);
                                              $$.dimensiones=$2.length;
                                            }
              | tipo lista_d ID '(' lista_parametros ')' '{' cuerpo_metodo '}'{
                                                 //METODO QUE DEVUELVE UN OBJETO
                                                 $$=new Metodo($3,$1,$8,$5);
                                                 $$.dimensiones=$2.length;
                                                 }
              | ID '(' lista_parametros ')' '{' cuerpo_metodo '}'{
                                          //CONSTRUCTOR
                                          $$=new Metodo($1,"VOID",$6,$3);
                                          $$.constructor=true;
                                        };
```

```
cuerpo_metodo: cuerpo_metodo sentencias_metodo{
                                $$=$1;
                                for(var i=0;i<$2.length;i++){
                                  $$.push($2[i]);
                                }
                              }
        | sentencias_metodo{
                    $$=$1;
                    };

//-------------------------sentencias dentro de un metodo



sentencias_metodo: declaracion_variables {$$=$1;}
            | sentencia_arreglo ';'{
                        $$=[];
                        $$.push($1);
                        }
            | sentencia_asignacion_arreglo';'{
                                $$=[];
                                $$.push($1);
                              }
            | sentencia_imprimir{
                        $$=[];
                        $$.push($1);
                        }
            | sentencia_if{
                        $$=[];
                        $$.push($1);
                      }
            | sentencia_switch{
                        $$=[];
                        $$.push($1);
                        }
            | sentencia_while{
                        $$=[];
                        $$.push($1);
                        }
            | sentencia_for{
                      $$=[];
                      $$.push($1);
                      }
            | sentencia_dowhile{
                        $$=[];
                        $$.push($1);
                        }
            | sentencia_asignacion ';'{
                            $$=[];
                            $$.push($1);
                           }
            | sentencia_break ';'{
```

```
                $$=[];
                $$.push($1);
              }
        | sentencia_incre_decre{
                $$=[];
                $$.push($1);
                }
        | sentencia_llamada ';'{
                $$=[];
                $$.push($1);
                }
        | sentencia_retorno ';'{
                $$=[];
                $$.push($1);
                }
        | sentencia_objetos ';'{
                $$=[];
                $$.push($1);
                };

sentencia_objetos: ID '.' sentencia_llamada{$$=new AccesoObjetos($1,$3);}
        | ID '.' ID '=' exp {
                $$=new AsignacionObjetos($1,$3,$5);
              };

sentencia_asignacion_arreglo: ID lista_dd '=' exp{
                        $$=new AsignacionArreglos($1,$4,$2);
                      }
            | ID lista_dd '=' NEW ID '(' ')'{
                        $$=new AsignacionArreglos($1,$5,$2);
                        $$.lista_valores=[];
                        $$.constructor_objeto=true;
                        }
            | ID lista_dd '=' NEW ID '(' lista_valores ')'{
                              $$=new AsignacionArreglos($1,$5,$2);
                              $$.lista_valores=$7;
                              $$.constructor_objeto=true;
                            };

sentencia_arreglo: tipo ID lista_d{
                $$=new DeclaracionArreglos($2,$1,[],$3.length,0,0);
              }
        | ID ID lista_d{
                $$=new DeclaracionArreglos($2,$1,[],$3.length,0,0);
                }
        | tipo ID lista_d '=' NEW tipo lista_dd{
                        $$=new DeclaracionArreglos($2,$1,[],$3.length,0,0);
                        $$.inicializado=true;
                        $$.lista_dimensiones=$7;
                        $$.tipo_asignacion=$6;
                        }
        | tipo ID lista_d '=' ID{
                    $$=new DeclaracionArreglos($2,$1,[],$3.length,0,0);
```

```
                        $$.inicializado=true;
                        $$.iniValue=new Aritmetica(null,null,false,$5,null,"ID",0,0);
                    }
        | tipo ID lista_d '=' sentencia_llamada{
                            $$=new DeclaracionArreglos($2,$1,[],$3.length,0,0);
                            $$.inicializado=true;
                            $$.iniValue=$5;
                            }
        | ID ID lista_d '=' NEW ID lista_dd{
                            $$=new DeclaracionArreglos($2,$1,[],$3.length,0,0);
                            $$.inicializado=true;
                            $$.lista_dimensiones=$7;
                            $$.tipo_asignacion=$6;
                            }
        | ID ID lista_d '=' ID{
                        $$=new DeclaracionArreglos($2,$1,[],$3.length,0,0);
                        $$.inicializado=true;
                        $$.iniValue=new Aritmetica(null,null,false,$5,null,"ID",0,0);
                    }
        | ID ID lista_dd '=' sentencia_llamada{
                            $$=new DeclaracionArreglos($2,$1,[],$3.length,0,0);
                            $$.inicializado=true;
                            $$.iniValue=$5;
                           };

lista_d: lista_d '['']'{
                $$=$1;
                $$.push(1);
                }
     | '[' ']'{
            $$=[];
            $$.push(1);
          };
lista_dd: lista_dd '[' exp ']'{
                    $$=$1;
                    $$.push($3);
                    }
     | '[' exp ']'{
            $$=[];
            $$.push($2);
            };


sentencia_retorno: RETORNO exp{$$=new Retorno($2);}
        | RETORNO{$$=new Retorno(null);};

sentencia_llamada: ID '(' ')'{
                    $$=new Llamada_Metodo($1,[]);
                    }
         | ID '(' lista_valores ')'{
                        $$=new Llamada_Metodo($1,$3);
                        }
         | THIS '(' ')'{
```

```
                $$=new Llamada_Metodo($1,[]);
                }
          | THIS'(' lista_valores ')'{
                        $$=new Llamada_Metodo($1,$3);
                        } ;
sentencia_incre_decre: ID '--' ';'{
                    var temp=new Aritmetica(null,null,false,$1,null,Type.ID,0,0);
                    $$=new Aritmetica(temp,null,true,null,"--",null,0,0);
                    }
                | ID '++' ';'{
                        var temp=new Aritmetica(null,null,false,$1,null,Type.ID,0,0);
                        $$=new Aritmetica(temp,null,true,null,"++",null,0,0);
                    }
                | '--' ID ';'{
                        var temp=new Aritmetica(null,null,false,$2,null,Type.ID,0,0);
                        $$=new Aritmetica(temp,null,true,null,"--",null,0,0);
                        }
                | '++' ID  ';'{
                        var temp=new Aritmetica(null,null,false,$2,null,Type.ID,0,0);
                        $$=new Aritmetica(temp,null,true,null,"++",null,0,0);
                        };
sentencia_break: BREAK{$$=new Detener($1);};
sentencia_asignacion: ID '=' exp{
                    $$=new Asignacion($1,$3,0);
                    }
              | ID '=' NEW ID '('  ')'{
                            $$=new Asignacion($1,$4,0);
                         }
              | ID '=' NEW ID '(' lista_valores ')'{
                                $$=new Asignacion($1,$4,0);
                                $$.lista_valores=$6;
                               }
              | THIS '.' elementos_this '=' exp{
                                //id,iniValue,dimensiones
                                var temp_this=new Este($3);
                                $$=new Asignacion(temp_this,$5,0);
                                };
elementos_this: elementos_this '.'  elemento_this{
                               $$=$1;
                               $$.push($3);
                               }
            | elemento_this{
                    $$=[];
                    $$.push($1);
                  };

elemento_this: ID{$$=new Aritmetica(null,null,false,$1,null,Type.ID,0,0);;}
          | ID '(' lista_valores ')'{
                          $$=new Llamada_Metodo($1,$3);
                        }
          | ID '(' ')'{$$=new Llamada_Metodo($1,[]);};

//-------------------------------------------SENTENCIA SWITCH
```

```
sentencia_switch: SWITCH '(' exp ')' '{' listas_cases case_default '}'{
                                        $$=new Selecciona($3,$6,$7);
                                        }
            | SWITCH '(' exp ')' '{' listas_cases '}'{
                                          $$=new Selecciona($3,$6,null);
                                       };
listas_cases: listas_cases lista_case{
                            $$=$1;
                            $$.push($2);
                          }
          | lista_case{
                    $$=[];
                    $$.push($1);
                    };
lista_case: CASE exp ':' cuerpo_metodo{
                            $$=new Caso($2,$4);
                            };
case_default: DEFAULT ':' cuerpo_metodo{
                            $$=new Caso(null,$3);
                            };
//---------------------------------------------FIN SWITCH

//---------------------------------------------SENTENCIA WHILE
sentencia_while: WHILE '(' exp ')' '{' cuerpo_metodo '}'{$$=new Mientras($3,$6,true);};
//---------------------------------------------FIN WHILE

//---------------------------------------------SENTENCIA FOR
sentencia_for:FOR '(' for_inicio ';' exp ';' exp ')' '{' cuerpo_metodo '}'{
                                                    //es un for normal
                                                    //inicializado,condicion,aumento,nodos
                                                    $$=new Para($3,$5,$7,$10,true);
                                                    }
          | FOR '(' for_inicio ':' exp ')' '{' cuerpo_metodo '}'{
                                                //aqui se va a realizar el foreach
                                                };
for_inicio: variables{$$=$1;}
          | sentencia_asignacion{$$=$1;};
//---------------------------------------------FIN FOR

//---------------------------------------------SENTENCIA DOWHILE
sentencia_dowhile: '{' cuerpo_metodo '}' DO WHILE '(' exp ')' ';'{$$=new Mientras($7,$2,false);};
//---------------------------------------------FIN DOWHILE

//---------------------------------------------SENTENCIA IF
sentencia_if: IF '(' exp ')' '{' cuerpo_metodo '}' sentencia_elseif sentencia_else{
                                                        //condicion,nodos,subifs,defecto
                                                        $$=new Si($3,$6,$8,$9);
                                                        }
          | IF '(' exp ')' '{' cuerpo_metodo '}' sentencia_elseif{
                                                //condicion,nodos,subifs,defecto
                                                $$=new Si($3,$6,$8,null);
                                                }
          | IF '(' exp ')' '{' cuerpo_metodo '}' sentencia_else{
```

```
                                        //condicion,nodos,subifs,defecto
                                        $$=new Si($3,$6,[],$8);
                                        }
        | IF '(' exp ')' '{' cuerpo_metodo '}'{
                                    //condicion,nodos,subifs,defecto
                                    $$=new Si($3,$6,[],null);
                                    };

sentencia_else: ELSE '{' cuerpo_metodo '}'{
                                $$=new Subsi(null,$3);
                            };

sentencia_elseif: sentencia_elseif ELSE IF '(' exp ')' '{' cuerpo_metodo '}'{
                                                $$=$1;
                                                $$.push(new Subsi($5,$8));
                                                }
            | ELSE IF '(' exp ')' '{' cuerpo_metodo '}'{
                                        $$=[];
                                        $$.push(new Subsi($4,$7));
                                        };
//-------------------------------------------FIN SENTENCIA IF









sentencia_imprimir: PRINT '(' exp ')' ';'{
                            $$=new Imprimir($3,false);
                            }
                | PRINTLN '(' exp ')' ';'{
                                $$=new Imprimir($3,true);
                                };
//------------------------- finalizan las sentencias que van dentro de un metodo



//aqui se tiene que agregar las asignaciones
lista_parametros: {$$=[];}
            | parametros{$$=$1;};
parametros: parametros ',' parametro{
                        $$=$1;
                        $$.push($3);
                        }
        | parametro{
                $$=[];
                $$.push($1);
                };
```

```
parametro: tipo ID{$$=new Declaracion($2,$1,null,[],0,0,0);}
       | ID ID{$$=new Declaracion($2,$1,null,[],0,0,0);}
       | tipo ID lista_d{$$=new DeclaracionArreglos($2,$1,[],$3.length,0,0);}
       | ID ID lista_d{$$=new DeclaracionArreglos($2,$1,[],$3.length,0,0);};


modificadores: modificadores modificador{
                            $$=$1;
                            $$.push($2);
                            }
         | modificador{
                $$=[];
                $$.push($1);
                };

modificador: ABSTRACT{$$=Visibilidad.ABSTRACT;}
          | STATIC{$$=Visibilidad.STATIC;}
          | FINAL{$$=Visibilidad.FINAL;}
          | visibilidad{$$=$1;};

visibilidad: PUBLIC{$$=Visibilidad.PUBLIC;}
         | PRIVATE{$$=Visibilidad.PRIVATE;}
         | PROTECTED{$$=Visibilidad.PROTECTED;};


tipo: INT{$$=PrimitiveType.INTEGER;}
   | STRING{$$=PrimitiveType.STRING;}
   | DOUBLE{$$=PrimitiveType.DOUBLE;}
   | CHAR{$$=PrimitiveType.CHAR;}
   | BOOLEAN{$$=PrimitiveType.BOOLEAN;}
   | VOID{$$=PrimitiveType.VOID;};

exp: '!' exp
      {
        $$=new Logica($2,null,true,null,"!",null,0,0);
      }
   | exp '&&' exp
      {
        $$=new Logica($1,$3,false,null,"&&",0,0);
      }
   | exp '||' exp
      {
        $$=new Logica($1,$3,false,null,"||",0,0);
      }
   | exp '^' exp
      {
        $$=new Logica($1,$3,false,null,"^",0,0);
      }
   | exp '>' exp
      {
        $$=new Relacional($1,$3,false,null,">",0,0);
      }
```

```
| exp '<' exp
  {
    $$=new Relacional($1,$3,false,null,"<",0,0);
  }
| exp '>=' exp
  {
    $$=new Relacional($1,$3,false,null,">=",0,0);
  }
| exp '<=' exp
  {
    $$=new Relacional($1,$3,false,null,"<=",0,0);
  }
| exp '==' exp
  {
    $$=new Relacional($1,$3,false,null,"==",0,0);
  }
| exp '!=' exp
  {
    $$=new Relacional($1,$3,false,null,"!=",0,0);
  }
| exp '+' exp
  {
    $$=new Aritmetica($1,$3,false,null,"+",null,0,0);
  }
| exp '-' exp
  {
    $$=new Aritmetica($1,$3,false,null,"-",null,0,0);
  }
| exp '*' exp
  {
    $$=new Aritmetica($1,$3,false,null,"*",null,0,0);
  }
| exp '/' exp
  {
    $$=new Aritmetica($1,$3,false,null,"/",null,0,0);
  }
| POW '(' exp ',' exp ')'
  {
    $$=new Aritmetica($3,$5,false,null,"^",null,0,0);
  }
| exp '%' exp
  {
    $$=new Aritmetica($1,$3,false,null,"%",null,0,0);
  }
| '(' exp ')'{
        $$ = $2;
     }
| '-' exp{
      $$=new Aritmetica($2,null,true,null,"-",null,0,0);
   }
| ID '--'{
      var temp=new Aritmetica(null,null,false,$1,null,Type.ID,0,0);
      $$=new Aritmetica(temp,null,true,null,"--",null,0,0);
```

```
        }
| ID '++'{
        var temp=new Aritmetica(null,null,false,$1,null,Type.ID,0,0);
        $$=new Aritmetica(temp,null,true,null,"++",null,0,0);
        }
| '--' ID {
        var temp=new Aritmetica(null,null,false,$2,null,Type.ID,0,0);
        $$=new Aritmetica(temp,null,true,null,"--",null,0,0);
        }
| '++' ID  {
        var temp=new Aritmetica(null,null,false,$2,null,Type.ID,0,0);
        $$=new Aritmetica(temp,null,true,null,"++",null,0,0);
        }
| NUMBER
  {
     $$=new Aritmetica(null,null,false,Number(yytext),null,PrimitiveType.INTEGER,0,0);
  }
| DECIMAL
  {
     $$=new Aritmetica(null,null,false,Number(yytext),null,PrimitiveType.DOUBLE,0,0);
  }
| STRING
  {
     $$=new Aritmetica(null,null,false,yytext,null,PrimitiveType.STRING,0,0);
  }
| FALSE
  {
     $$=new Aritmetica(null,null,false,"false",null,PrimitiveType.BOOLEAN,0,0);
  }
| TRUE
  {
     $$=new Aritmetica(null,null,false,"true",null,PrimitiveType.BOOLEAN,0,0);
  }
| CHAR
  {
     $$=new Aritmetica(null,null,false,yytext,null,PrimitiveType.CHAR,0,0);
  }
| ID{
     $$=new Aritmetica(null,null,false,yytext,null,Type.ID,0,0);
  }
| sentencia_ternario{$$=$1;}
| THIS '.' elementos_this{
                        //id,iniValue,dimensiones
                        $$=new Este($3);
                     }
| sentencia_llamada{$$=$1;}
| NULO{$$=new Aritmetica(null,null,false,"null",null,PrimitiveType.NULO,0,0);}
| ID lista_dd{
        $$=new Aritmetica(null,null,false,$1,null,"ARRAY",0,0);
        $$.lista_dimensiones=$2;
     }
| ID '.' sentencia_llamada{$$=new AccesoObjetos($1,$3);}
| ID '.' ID{$$=new AsignacionObjetos($1,$3,null);};
```

```
sentencia_ternario: exp '?' exp ':' exp{$$=new Ternario($1,$3,$5);};
%%
function yyerror(s,linea,columna){
   var error_1=new Errores("Sintactico",s,linea,columna);
   lista_errores.push(error_1);
}
//yyerror(hash.token,hash.line,hash.loc.first_column);
```