Universidad de San Carlos de Guatemala

Facultad de Ingenieria

Escuela de Ciencia y Sistemas

Manual Técnico FISQL

Marvin Emmanuel Pivaral Orellana

201213587

**Servidor de Base de Datos**

**Clases**

1. FActualizar
2. FAlterarObjeto
3. FTabla
4. FAlterarUSuario
5. FAsignacion
6. FBorrar
7. FCaso
8. FContar
9. FCrearBaseDatos
10. FCrearObjeto
11. FCrearTabla
12. FCrearUSuario
13. FDeclaracion
14. FMetodo
15. FMientras
16. FNodoExpresion
17. FObjeto
18. FPara
19. FSelecciona
20. FSeleccionar
21. FSi
22. FUsarBaseDatos

package Funciones.Usql;

import EjecucionUsql.\*;

import Funciones.XML.Funcion;

import Funciones.XML.Parametro;

import Funciones.XML.Procedimiento;

//import Interface.Tools;

import java.util.ArrayList;

import Static.\*;

import static Static.Tools.\*;

/\*\*

\*

\* @author Titus

\*/

public class FMetodo {

public Ambito Ambito;

public ArrayList<Simbolo> Parametros;

public int Fila, Columna;

public String Tipo, Nombre;

public FMetodo(ArrayList<Simbolo> parametro, Ambito ambito, int fila, int columna, String tipo, String nombre) {

this.Ambito = ambito;

this.Parametros = parametro;

this.Fila = fila;

this.Columna = columna;

this.Tipo = tipo;

this.Nombre = nombre;

}

public FMetodo() {

}

public void EjecutarCrearMetodo() {

String cadena = getCadena();

ArrayList<Parametro> parametros = new ArrayList<Parametro>();

for (Simbolo sim : this.Parametros) {

FDeclaracion d = (FDeclaracion) sim.Valor;

if (d.Tipo.equals(Constante.TEntero) || d.Tipo.equals(Constante.TDecimal) || d.Tipo.equals(Constante.TBool) || d.Tipo.equals(Constante.TText) || d.Tipo.equals(Constante.TDate) || d.Tipo.equals(Constante.TDateTime)) {

parametros.add(new Parametro(d.Tipo, d.Nombre));

} else {

parametros.add(new Parametro(Constante.TObjeto, d.Nombre, d.Tipo));

}

}

if (this.Tipo.equals(Constante.TVacio)) {

Procedimiento p = new Procedimiento(this.Nombre, cadena, parametros);

switch (Tools.Base\_de\_datos.DLLCrearProcedimiento(p)) {

case 1:

Tools.InsertarError(Constante.TErrorSemantico, "No ha seleccionado una base de datos.", Fila, Columna);

break;

case 2:

Tools.InsertarError(Constante.TErrorSemantico, "Ya existe un procedimiento " + this.Nombre + " en la base de datos " + Tools.BaseActual.Nombre, Fila, Columna);

break;

case 3:

Tools.InsertarError(Constante.TErrorSemantico, "Ya existe una funcion " + this.Nombre + " en la base de datos " + Tools.BaseActual.Nombre, Fila, Columna);

break;

default:

Tools.Funciones.InsertarVariable(new Variable(this.Tipo, this.Nombre, Constante.TMetodo, this.Fila, this.Columna, this.Ambito, this));

Tools.ImprimirLog("Crear Procedimiento", "Se creo el procedimiento " + this.Nombre + " en la base de datos " + Tools.BaseActual.Nombre);

break;

}

} else {

Funcion f = new Funcion(Tipo, Nombre, cadena, parametros);

switch (Tools.Base\_de\_datos.DLLCrearFuncion(f)) {

case 1:

Tools.InsertarError(Constante.TErrorSemantico, "No ha seleccionado una base de datos.", Fila, Columna);

break;

case 2:

Tools.InsertarError(Constante.TErrorSemantico, "Ya existe un procedimiento " + this.Nombre + " en la base de datos " + Tools.BaseActual.Nombre, Fila, Columna);

break;

case 3:

Tools.InsertarError(Constante.TErrorSemantico, "Ya existe una funcion " + this.Nombre + " en la base de datos " + Tools.BaseActual.Nombre, Fila, Columna);

break;

default:

Tools.Funciones.InsertarVariable(new Variable(this.Tipo, this.Nombre, Constante.TMetodo, this.Fila, this.Columna, this.Ambito, this));

Tools.ImprimirLog("Crear Funcion", "Se creo la funcion " + this.Nombre + " en la base de datos " + Tools.BaseActual.Nombre);

break;

}

}

}

public String getCadena() {

String cadena = "";

if (Tipo.equals(Constante.TVacio)) {

cadena += "CREAR PROCEDIMIENTO " + this.Nombre + "(";

} else {

cadena += "CREAR FUNCION " + this.Nombre + "(";

}

int i = 0;

for (Simbolo sim : Parametros) {

FDeclaracion d = (FDeclaracion) sim.Valor;

if (i == 0) {

cadena += d.getCadenaParametro();

} else {

cadena += ", " + d.getCadenaParametro();

}

i++;

}

if (Tipo.equals(Constante.TVacio)) {

cadena += "){\n";

} else {

cadena += ")" + this.Tipo + "{\n";

}

cadena += "\t" + getCadenaCuerpo(Ambito.TablaSimbolo).replace("\n", "\n\t");

cadena += "\n}\n";

return cadena;

}

public Variable EjecutarMetodo(FLlamadaMetodo llamada, Variable metodo) {

Variable resultado = null;

if (this.Parametros.size() == llamada.Parametros.size()) {

int cont = 0;

//metemos la variables de los parametros a la tabla

while (cont < this.Parametros.size() && Tools.ContarErrores()) {

//aqui le enviamos al padre

FNodoExpresion resultadoparametro = llamada.Parametros.get(cont).ResolverExpresion();

if (Tools.ContarErrores()) {

if (this.Parametros.get(cont).Tipo.equals(resultadoparametro.Tipo) || (resultadoparametro.Tipo.equals(Constante.TObjeto) && this.Parametros.get(cont).Tipo.equals(resultadoparametro.Cadena))) {

///////////////////////

FDeclaracion fd = (FDeclaracion) this.Parametros.get(cont).Valor;

//le asignamos el valor a la declaracion

fd.Valor = resultadoparametro;

//ejecutamos la declaracion---------------->

fd.EjecutarDeclaracion();

} else {

if (resultadoparametro.Tipo.equals(Constante.TObjeto)) {

Tools.InsertarError(Constante.TErrorSemantico, "Se esperaba un tipo " + this.Parametros.get(cont).Tipo + ", no un tipo " + resultadoparametro.Nombre, llamada.Fila, llamada.Columna);

} else {

Tools.InsertarError(Constante.TErrorSemantico, "Se esperaba un tipo " + this.Parametros.get(cont).Tipo + ", no un tipo " + resultadoparametro.Tipo, llamada.Fila, llamada.Columna);

}

}

}

cont++;

}

//ejecutamos el cuerpo

if (Tools.ContarErrores()) {

EjecutarInstrucciones(this.Ambito.TablaSimbolo);

//obtenemos el valor del retun si hay

if (Tools.Tabla.IsRertorno()) {

Variable v = Tools.Tabla.ObtenerTope();

FNodoExpresion exp = (FNodoExpresion) v.Valor;

if (exp.Tipo.equals(this.Tipo) || exp.Tipo.equals(Constante.TObjeto) && exp.Nombre.equals(this.Tipo)) {

resultado = v;

} else {

if (exp.Tipo.equals(Constante.TObjeto) && exp.Nombre.equals(this.Tipo)) {

Tools.InsertarError(Constante.TErrorSemantico, "El metodo " + this.Nombre + " es del tipo " + this.Tipo + " y se retorno un valor tipo " + exp.Nombre, this.Fila, this.Columna);

} else {

Tools.InsertarError(Constante.TErrorSemantico, "El metodo " + this.Nombre + " es del tipo " + this.Tipo + " y se retorno un valor tipo " + exp.Tipo, this.Fila, this.Columna);

}

}

Tools.Tabla.SacarVariable();

}

SacarAmbito(Ambito.TablaSimbolo);

SacarAmbito(this.Parametros);

}

} else {

Tools.InsertarError(Constante.TErrorSemantico, "El metodo esperaba " + this.Parametros.size() + " parametros", llamada.Fila, llamada.Columna);

}

return resultado;

}

public void EjecutarInstrucciones(ArrayList<Simbolo> instrucciones) {

if (Tools.ContarErrores()) {

for (Simbolo instruccion : instrucciones) {

if (Tools.ContarErrores() && !Tools.Tabla.IsDetener() && !Tools.Tabla.IsRertorno()) {

switch (instruccion.Rol) {

case Constante.TImprimir:

EjecutarImprimir(instruccion);

break;

case Constante.TMetodo:

EjecutarMetodo(instruccion);

break;

case Constante.TDeclaracion:

EjecutarDeclaracion(instruccion);

break;

case Constante.TAsignacion:

EjecutarAsignacion(instruccion);

break;

case Constante.TRetorno:

EjecutarRetorno(instruccion);

break;

case Constante.TDetener:

EjecutarDetener(instruccion);

break;

case Constante.TSi:

EjecutarSi(instruccion);

break;

case Constante.TMientras:

EjecutarMientras(instruccion);

break;

case Constante.TSeleccion:

EjecutarSelecciona(instruccion);

break;

case Constante.TPara:

EjecutarPara(instruccion);

break;

case Constante.TCrearBaseDatos:

EjecutarCrearBaseDatos(instruccion);

break;

case Constante.TCrearTabla:

EjecutarCrearTabla(instruccion);

break;

case Constante.TUsarBaseDatos:

EjecutarUsarBase(instruccion);

break;

case Constante.TCrearUsuario:

EjecutarCrearUsuario(instruccion);

break;

case Constante.TCrearObjeto:

EjecutarCrearObjeto(instruccion);

break;

case Constante.TLlamadaMetodo:

EjecutarLlamadaMetodo(instruccion);

break;

case Constante.TAlterarTabla:

EjecutarAlterarTabla(instruccion);

break;

case Constante.TAlterarUsuario:

EjecutarAlterarUsuario(instruccion);

break;

case Constante.TAlterarObjeto:

EjecutarAlterarObjeto(instruccion);

break;

case Constante.TEliminarUsuario:

EjecutarEliminarUsuario(instruccion);

break;

case Constante.TEliminarTabla:

EjecutarEliminarTabla(instruccion);

break;

case Constante.TEliminarObjeto:

EjecutarEliminarObjeto(instruccion);

break;

case Constante.TEliminarProcedimiento:

EjecutarEliminarProcedimiento(instruccion);

break;

case Constante.TEliminarFuncion:

EjecutarEliminarFuncion(instruccion);

break;

case Constante.TEliminarBaseDatos:

EjecutarEliminarBaseDatos(instruccion);

break;

case Constante.TInsertarNormal:

EjecutarInsertarNormal(instruccion);

break;

case Constante.TInsertarEspecial:

EjecutarInsertarEspecial(instruccion);

break;

case Constante.TSeleccionar:

EjecutarSeleccionar(instruccion);

break;

case Constante.TActualizar:

EjecutarActualizar(instruccion);

break;

case Constante.TBorrar:

EjecutarBorrar(instruccion);

break;

default:

break;

}

}

}

}

}

public void EjecutarBorrar(Simbolo instruccion){

FBorrar b = (FBorrar)instruccion.Valor;

b.Ejecutar();

}

public void EjecutarActualizar(Simbolo instruccion) {

FActualizar a = (FActualizar) instruccion.Valor;

a.Ejecutar();

}

public void EjecutarSeleccionar(Simbolo instruccion) {

FSeleccionar se = (FSeleccionar) instruccion.Valor;

se.Ejecutar();

}

public void EjecutarInsertarEspecial(Simbolo instruccion) {

FInsertarEspecial in = (FInsertarEspecial) instruccion.Valor;

in.Ejecutar();

}

public void EjecutarInsertarNormal(Simbolo instruccion) {

FInsertarNormal in = (FInsertarNormal) instruccion.Valor;

in.Ejecutar();

}

public void EjecutarEliminarUsuario(Simbolo instruccion) {

FEliminarUsuario usr = (FEliminarUsuario) instruccion.Valor;

usr.Ejecutar();

}

public void EjecutarEliminarTabla(Simbolo instruccion) {

FEliminarTabla usr = (FEliminarTabla) instruccion.Valor;

usr.Ejecutar();

}

public void EjecutarEliminarObjeto(Simbolo instruccion) {

FEliminarObjeto usr = (FEliminarObjeto) instruccion.Valor;

usr.Ejecutar();

}

public void EjecutarEliminarFuncion(Simbolo instruccion) {

FEliminarFuncion usr = (FEliminarFuncion) instruccion.Valor;

usr.Ejecutar();

}

public void EjecutarEliminarProcedimiento(Simbolo instruccion) {

FEliminarProcedimiento usr = (FEliminarProcedimiento) instruccion.Valor;

usr.Ejecutar();

}

public void EjecutarEliminarBaseDatos(Simbolo instruccion) {

FEliminarBaseDatos usr = (FEliminarBaseDatos) instruccion.Valor;

usr.Ejecutar();

}

public void EjecutarAlterarUsuario(Simbolo instruccion) {

FAlterarUsuario usr = (FAlterarUsuario) instruccion.Valor;

usr.Ejecutar();

}

public void EjecutarAlterarTabla(Simbolo instruccion) {

FAlterarTabla at = (FAlterarTabla) instruccion.Valor;

at.Ejecutar();

}

public void EjecutarAlterarObjeto(Simbolo instruccion) {

FAlterarObjeto ao = (FAlterarObjeto) instruccion.Valor;

ao.Ejecutar();

}

public void EjecutarLlamadaMetodo(Simbolo instruccion) {

FLlamadaMetodo lm = (FLlamadaMetodo) instruccion.Valor;

lm.EjecutarLlamadaMetodo();

}

public void EjecutarUsarBase(Simbolo instruccion) {

FUsarBaseDatos usarbase = (FUsarBaseDatos) instruccion.Valor;

usarbase.Ejecutar();

}

public void EjecutarCrearObjeto(Simbolo instruccion) {

FCrearObjeto crearobjeto = (FCrearObjeto) instruccion.Valor;

crearobjeto.Ejecutar();

}

public void EjecutarCrearUsuario(Simbolo instruccion) {

FCrearUsuario crearusuario = (FCrearUsuario) instruccion.Valor;

crearusuario.Ejecutar();

}

public void EjecutarCrearTabla(Simbolo instruccion) {

FCrearTabla creartabla = (FCrearTabla) instruccion.Valor;

creartabla.Ejecutar();

}

public void EjecutarCrearBaseDatos(Simbolo instruccion) {

FCrearBaseDatos crearbase = (FCrearBaseDatos) instruccion.Valor;

crearbase.Ejecutar();

}

public void EjecutarPara(Simbolo instruccion) {

FPara para = (FPara) instruccion.Valor;

para.EjecutarPara();

}

public void EjecutarSi(Simbolo instruccion) {

FSi si = (FSi) instruccion.Valor;

si.EjecutarSi();

}

public void EjecutarSelecciona(Simbolo instruccion) {

FSelecciona selecciona = (FSelecciona) instruccion.Valor;

selecciona.EjecutarSelecciona();

}

public void EjecutarMientras(Simbolo instruccion) {

FMientras mientras = (FMientras) instruccion.Valor;

mientras.EjecutarMientras();

}

public void EjecutarRetorno(Simbolo instruccion) {

FNodoExpresion exp = (FNodoExpresion) instruccion.Valor;

exp = exp.ResolverExpresion();

Tools.Tabla.InsertarVariable(new Variable(exp.Nombre, instruccion.Nombre, instruccion.Nombre, instruccion.Fila, instruccion.Columna, instruccion.Ambito, exp));

}

public void EjecutarDetener(Simbolo instruccion) {

Tools.Tabla.InsertarVariable(new Variable(instruccion.Tipo, instruccion.Nombre, instruccion.Rol, instruccion.Fila, instruccion.Columna, instruccion.Ambito, null));

}

public void EjecutarContinuar(Simbolo instruccion) {

Tools.Tabla.InsertarVariable(new Variable(instruccion.Tipo, instruccion.Nombre, instruccion.Rol, instruccion.Fila, instruccion.Columna, instruccion.Ambito, null));

}

public void EjecutarMetodo(Simbolo instruccion) {

FMetodo metodo = (FMetodo) instruccion.Valor;

metodo.EjecutarCrearMetodo();

}

public void EjecutarDeclaracion(Simbolo instruccion) {

FDeclaracion declaracion = (FDeclaracion) instruccion.Valor;

declaracion.EjecutarDeclaracion();

}

public void EjecutarAsignacion(Simbolo instruccion) {

FAsignacion asigna = (FAsignacion) instruccion.Valor;

asigna.EjecutarAsignacion();

}

public void EjecutarImprimir(Simbolo instruccion) {

FImprimir imprimir = (FImprimir) instruccion.Valor;

imprimir.Imprimir();

}

public String getCadenaCuerpo(ArrayList<Simbolo> instrucciones) {

String cadena = "";

for (Simbolo instruccion : instrucciones) {

if (Tools.ContarErrores() && !Tools.Tabla.IsDetener() && !Tools.Tabla.IsRertorno()) {

switch (instruccion.Rol) {

case Constante.TImprimir:

cadena += GetCadenaImprimir(instruccion);

break;

case Constante.TMetodo:

cadena += GetCadenaMetodo(instruccion);

break;

case Constante.TDeclaracion:

cadena += GetCadenaDeclaracion(instruccion);

break;

case Constante.TAsignacion:

cadena += GetCadenaAsignacion(instruccion);

break;

case Constante.TRetorno:

cadena += GetCadenaRetorno(instruccion);

break;

case Constante.TDetener:

cadena += GetCadenaDetener(instruccion);

break;

case Constante.TSi:

cadena += GetCadenaSi(instruccion);

break;

case Constante.TMientras:

cadena += GetCadenaMientras(instruccion);

break;

case Constante.TSeleccion:

cadena += GetCadenaSelecciona(instruccion);

break;

case Constante.TPara:

cadena += GetCadenaPara(instruccion);

break;

case Constante.TCrearBaseDatos:

cadena += GetCadenaCrearBaseDatos(instruccion);

break;

case Constante.TCrearTabla:

cadena += GetCadenaCrearTabla(instruccion);

break;

case Constante.TUsarBaseDatos:

cadena += GetCadenaUsarBase(instruccion);

break;

case Constante.TCrearUsuario:

cadena += GetCadenaCrearUsuario(instruccion);

break;

case Constante.TCrearObjeto:

cadena += GetCadenaCrearObjeto(instruccion);

break;

case Constante.TAlterarTabla:

cadena += GetCadenaAlterarTabla(instruccion);

break;

case Constante.TAlterarUsuario:

cadena += GetCadenaAlterarUsuario(instruccion);

break;

case Constante.TAlterarObjeto:

cadena += GetCadenaAlterarObjeto(instruccion);

break;

case Constante.TEliminarUsuario:

cadena += GetCadenaEliminarUsuario(instruccion);

break;

case Constante.TEliminarTabla:

cadena += GetCadenaEliminarTabla(instruccion);

break;

case Constante.TEliminarObjeto:

cadena += GetCadenaEliminarObjeto(instruccion);

break;

case Constante.TEliminarProcedimiento:

cadena += GetCadenaEliminarProcedimiento(instruccion);

break;

case Constante.TEliminarFuncion:

cadena += GetCadenaEliminarFuncion(instruccion);

break;

case Constante.TEliminarBaseDatos:

cadena += GetCadenaEliminarBaseDatos(instruccion);

break;

case Constante.TInsertarNormal:

cadena += GetCadenaInsertarNormal(instruccion);

break;

case Constante.TInsertarEspecial:

cadena += GetCadenaInsertarEspecial(instruccion);

break;

case Constante.TSeleccionar:

cadena += GetCadenaSeleccionar(instruccion);

break;

case Constante.TLlamadaMetodo:

cadena += GetCadenaLlamadaMetodo(instruccion);

break;

case Constante.TActualizar:

cadena += GetCadenaActualizar(instruccion);

break;

case Constante.TBorrar:

cadena += GetCadenaBorrar(instruccion);

break;

default:

break;

}

}

}

return cadena;

}

public String GetCadenaBorrar(Simbolo instruccion){

FBorrar b = (FBorrar)instruccion.Valor;

return b.getCadena() + ";\n";

}

public String GetCadenaActualizar(Simbolo instruccion){

FActualizar fa = (FActualizar)instruccion.Valor;

return fa.getCadena() + ";\n";

}

public String GetCadenaLlamadaMetodo(Simbolo instruccion){

FLlamadaMetodo lm = (FLlamadaMetodo)instruccion.Valor;

return lm.getCadena() + ";\n";

}

public String GetCadenaAlterarTabla(Simbolo instruccion) {

FAlterarTabla at = (FAlterarTabla) instruccion.Valor;

return at.getCadena() + "\n";

}

public String GetCadenaMetodo(Simbolo instruccion) {

FMetodo metodo = (FMetodo) instruccion.Valor;

return metodo.getCadena() + "\n";

}

public String GetCadenaUsarBase(Simbolo instruccion) {

FUsarBaseDatos usarbase = (FUsarBaseDatos) instruccion.Valor;

return usarbase.getCadena() + ";\n";

}

public String GetCadenaCrearObjeto(Simbolo instruccion) {

FCrearObjeto crearobjeto = (FCrearObjeto) instruccion.Valor;

return crearobjeto.getCadena() + "\n";

}

public String GetCadenaCrearUsuario(Simbolo instruccion) {

FCrearUsuario crearusuario = (FCrearUsuario) instruccion.Valor;

return crearusuario.getCadena() + ";\n";

}

public String GetCadenaCrearTabla(Simbolo instruccion) {

FCrearTabla creartabla = (FCrearTabla) instruccion.Valor;

return creartabla.getCadena() + ";\n";

}

public String GetCadenaCrearBaseDatos(Simbolo instruccion) {

FCrearBaseDatos crearbase = (FCrearBaseDatos) instruccion.Valor;

return crearbase.getCadena() + ";\n";

}

public String GetCadenaPara(Simbolo instruccion) {

FPara para = (FPara) instruccion.Valor;

return para.getCadena() + "\n";

}

public String GetCadenaSi(Simbolo instruccion) {

FSi si = (FSi) instruccion.Valor;

return si.getCadena() + "\n";

}

public String GetCadenaSelecciona(Simbolo instruccion) {

FSelecciona selecciona = (FSelecciona) instruccion.Valor;

return selecciona.getCadena() + "\n";

}

public String GetCadenaMientras(Simbolo instruccion) {

FMientras mientras = (FMientras) instruccion.Valor;

return mientras.getCadena() + "\n";

}

public String GetCadenaRetorno(Simbolo instruccion) {

FNodoExpresion exp = (FNodoExpresion) instruccion.Valor;

return "RETORNO " + exp.getCadena() + ";\n";

}

public String GetCadenaDetener(Simbolo instruccion) {

return "DETENER;\n";

}

public String GetCadenaDeclaracion(Simbolo instruccion) {

FDeclaracion declaracion = (FDeclaracion) instruccion.Valor;

return declaracion.getCadena() + ";\n";

}

public String GetCadenaAsignacion(Simbolo instruccion) {

FAsignacion asigna = (FAsignacion) instruccion.Valor;

return asigna.getCadena() + ";\n";

}

public String GetCadenaImprimir(Simbolo instruccion) {

FImprimir imprimir = (FImprimir) instruccion.Valor;

return imprimir.getCadena() + ";\n";

}

public String GetCadenaAlterarUsuario(Simbolo instruccion) {

FAlterarUsuario usr = (FAlterarUsuario) instruccion.Valor;

return usr.getCadena() + ";\n";

}

public String GetCadenaAlterarObjeto(Simbolo instruccion) {

FAlterarObjeto usr = (FAlterarObjeto) instruccion.Valor;

return usr.getCadena();

}

public String GetCadenaEliminarUsuario(Simbolo instruccion) {

FEliminarUsuario usr = (FEliminarUsuario) instruccion.Valor;

return usr.getCadena() + ";\n";

}

public String GetCadenaEliminarTabla(Simbolo instruccion) {

FEliminarTabla usr = (FEliminarTabla) instruccion.Valor;

return usr.getCadena() + ";\n";

}

public String GetCadenaEliminarObjeto(Simbolo instruccion) {

FEliminarObjeto usr = (FEliminarObjeto) instruccion.Valor;

return usr.getCadena() + ";\n";

}

public String GetCadenaEliminarFuncion(Simbolo instruccion) {

FEliminarFuncion usr = (FEliminarFuncion) instruccion.Valor;

return usr.getCadena() + ";\n";

}

public String GetCadenaEliminarProcedimiento(Simbolo instruccion) {

FEliminarProcedimiento usr = (FEliminarProcedimiento) instruccion.Valor;

return usr.getCadena() + ";\n";

}

public String GetCadenaEliminarBaseDatos(Simbolo instruccion) {

FEliminarBaseDatos usr = (FEliminarBaseDatos) instruccion.Valor;

return usr.getCadena() + ";\n";

}

public String GetCadenaInsertarNormal(Simbolo instruccion) {

FInsertarNormal in = (FInsertarNormal) instruccion.Valor;

return in.getCadena() + ";\n";

}

public String GetCadenaInsertarEspecial(Simbolo instruccion) {

FInsertarEspecial in = (FInsertarEspecial) instruccion.Valor;

return in.getCadena() + ";\n";

}

public String GetCadenaSeleccionar(Simbolo instruccion) {

FSeleccionar fs = (FSeleccionar)instruccion.Valor;

return fs.getCadena() + ";\n";

}

public void SacarAmbito(ArrayList<Simbolo> ambito) {

int cont = ambito.size() - 1;

while (cont >= 0) {

if (ambito.get(cont).Rol.equals(Constante.TDeclaracion)) {

if (Tools.Tabla.ExisteVariableTope(ambito.get(cont).Nombre)) {

Tools.Tabla.SacarVariable(ambito.get(cont).Nombre);

}

}

cont--;

}

}

}