Gramática final

Inicio **->** Clase Clases **$**
Clases **->** Clase Clases **| ε**
Clase **->** class idClase Herencia { Miembros }
Herencia **->** extends idClase **| ε**
Miembros **->** Miembro Miembros **| ε**
Miembro **->** Atributo **|** Ctor **|** Metodo
Atributo **->** Visibilidad Tipo ListaDecVars ;
Metodo **->** FormaMetodo TipoMetodo idMetVar ArgsFormales Bloque
Ctor **->** idClase ArgsFormales Bloque
ArgsFormales **->** (ListaArg)
ListaArg **-> ε |** Arg , ArgFormales
ArgFormales **-> ε |** Arg , ArgFormales
Arg **->** Tipo idMetVar
FormaMetodo **->** static **|** dynamic
Visibilidad **->** public **|** private
TipoMetodo **->** Tipo **|** void
Tipo **->** TipoPrimitivo **|** TipoReferencia
TipoPrimitivo **->** boolean **|** char **|** int
TipoReferencia **->** idClase **|** String **|** TipoPrimitivo [ ]
ListaDecVars **->** idMetVar ListaDV
ListaDV **->** , idMetVar ListaDV **| ε**
Bloque **->** { Sentencias }
Sentencias **->** Sentencia Sentencias **| ε**
Sentencia **->** ;
Sentencia **->** Asignacion ;
Sentencia **->** SentenciaLlamada ;
Sentencia **->** Tipo ListaDecVars ;
Sentencia **->** if (Expresion) Sentencia SentenciaElse
Sentencia **->** while (Expresion) Sentencia
Sentencia **->** Bloque
Sentencia **->** return Expresiones ;
SentenciaElse **-> ε |** else Sentencia
Expresiones **->** Expresion Expresiones **| ε**
Asignacion **->** AccesoVar **=** Expresion **|** AccesoThis **=** Expresion
SentenciaLlamada **->** (Primario)
Expresion **->** ExpOr
ExpOr **->** ExpAnd ExpOrR
ExpOrR **->** || ExpAnd ExpOrR **| ε**
ExpAnd **->** ExpIg ExpAndR
ExpAndR **->** && ExpIg ExpAndR **| ε**
ExpIg **->** ExpComp ExpIgR
ExpIgR **->** OpIgual ExpComp ExpIgR **| ε**
ExpComp **->** ExpAd ExpCompR
ExpCompR **->** OpComp ExpAd **| ε**
ExpAd **->** ExpMul ExpAdR
ExpAdR **->** OpAd ExpMul ExpAdR **| ε**
ExpMul **->** ExpUn ExpMulR
ExpMulR **->** OpMul ExpUn ExpMulR **| ε**
ExpUn **->** OpUn ExpUn **|** Operando
OpIgual **-> == |** !=

OpComp **->** < **|** > **|** <**=** **|** >**=**
OpAd **->** + **|** -
OpUn **->** + **|** - **|** !
OpMul **->** * **|** /
Operando **->** Literal
Operando **->** Primario
Literal **->** null **|** true **|** false **|** intLiteral **|** charLiteral **|** stringLiteral
Primario **->** ExpresionParentizada
Primario **->** AccesoThis
Primario **->** idMetVar MetodoVariable
Primario **->** LlamadaMetodoEstatico
Primario **->** LlamadaCtor
MetodoVariable **->** Encadenado **|** ArgsActuales Encadenado
ExpresionParentizada **->** (Expresion) Encadenado
Encadenado **->** **ε** **|** . idMetVar Acceso **|** AccesoArregloEncadenado
Acceso **->** LlamadaMetodoEncadenado **|** AccesoVarEncadenado
AccesoThis **->** this Encadenado
AccesoVar **->** idMetVar Encadenado
LlamadaMetodo **->** idMetVar ArgsActuales Encadenado
LlamadaMetodoEstatico **->** idClase . LlamadaMetodo Encadenado
LlamadaCtor **->** new LlamadaCtorR
LlamadaCtorR **->** idClase ArgsActuales Encadenado **|** TipoPrimitivo [Expresion] Encadenado
ArgsActuales **->** (ListaExpresiones)
ListaExpresiones **->** Expresion ListaExp **|** **ε**
ListaExp **->** , Expresion ListaExp **|** **ε**
LlamadaMetodoEncadenado **->** ArgsActuales Encadenado
AccesoVarEncadenado **->** Encadenado
AccesoArregloEncadenado **->** [Expresion] Encadenado

Primeros

Inicio= { class }
Clases= { ε, class }
Clase= { class }
Herencia= { extends, ε }
Miembros= { ε, public, private, static, dynamic, idClase }
Miembro= { public, private, static, dynamic, idClase }
Atributo= { public, private }
Visibilidad= { public, private }
Ctor= { idClase }
Metodo= { static, dynamic }
FormaMetodo= { static, dynamic }
ArgsFormales= { (}
ListaArg= { ε, boolean, char, int, idClase, String }
Arg= { boolean, char, int, idClase, String }
Tipo= { boolean, char, int, idClase, String }
TipoPrimitivo= { boolean, char, int }
TipoReferencia= { idClase, String, boolean, char, int }
ArgFormales= { ε, boolean, char, int, idClase, String }
TipoMetodo= { void, boolean, char, int, idClase, String }
ListaDecVars= { idMetVar }
ListaDV= { ,, ε }
Bloque= { { }
Sentencias=
{ ε, ;, if, while, return, idMetVar, this, boolean, char, int, idClase, String, (, { }
Sentencia=
{ ;, if, while, return, idMetVar, this, boolean, char, int, idClase, String, (, { }
SentenciaElse= { ε, else }
Asignacion= { idMetVar, this }
AccesoThis= { this }
AccesoVar= { idMetVar }
SentenciaLlamada= { (}
Expresiones=
{ ε, +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, (, idMetVar, idClase, this, new }
Expresion=
{ +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, (, idMetVar, idClase, this, new }
ExpOr=
{ +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, (, idMetVar, idClase, this, new }
ExpAnd=
{ +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, (, idMetVar, idClase, this, new }
ExpIg=
{ +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, (, idMetVar, idClase, this, new }
ExpComp=
{ +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, (, idMetVar, idClase, this, new }
ExpAd=
{ +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, (, idMetVar, idClase, this, new }

ExpMul=
{ +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, (, idMetVar, idClase, this, new }
ExpUn=
{ +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, (, idMetVar, idClase, this, new }
OpUn= { +, -, ! }
Operando=
{ null, true, false, intLiteral, charLiteral, stringLiteral, (, idMetVar, idClase, this, new }
Literal= { null, true, false, intLiteral, charLiteral, stringLiteral }
Primario= { (, idMetVar, idClase, this, new }
MetodoVariable= { ε, ., [, (}
ExpresionParentizada= { (}
LlamadaMetodo= { idMetVar }
LlamadaMetodoEstatico= { idClase }
LlamadaCtor= { new }
ExpOrR= { || , ε }
ExpAndR= { &&, ε }
ExpIgR= { ε, ==, != }
OpIgual= { ==, != }
ExpCompR= { ε, <, >, <=, >= }
ExpAdR= { ε, +, - }
OpAd= { +, - }
ExpMulR= { ε, *, / }
OpMul= { *, / }
OpComp= { <, >, <=, >= }
Encadenado= { ε, ., [}
AccesoArregloEncadenado= { [}
Acceso= { (, ε, ., [}
LlamadaMetodoEncadenado= { (}
AccesoVarEncadenado= { ε, ., [}
LlamadaCtorR= { idClase, boolean, char, int }
ArgsActuales= { (}
ListaExpresiones=
{ ε, +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, (, idMetVar, idClase, this, new }
ListaExp= { ,, ε }

Siguientes

Arg= { , }
FormaMetodo= { void, boolean, char, int, idClase, String }
Visibilidad= { boolean, char, int, idClase, String }
TipoMetodo= { idMetVar }
Tipo= { idMetVar }
TipoPrimitivo= { [, idMetVar }
TipoReferencia= { idMetVar }
ListaDecVars= { ; }
ListaDV= { ; }
Bloque= { ε, else, ;, if, while, return, idMetVar, this, boolean, char, int, idClase, String, (, {, public, private, static, dynamic }
Sentencias= { } }
Sentencia= { ε, else, ;, if, while, return, idMetVar, this, boolean, char, int, idClase, String, (, { }
SentenciaElse= { ε, else, ;, if, while, return, idMetVar, this, boolean, char, int, idClase, String, (, { }
Expresiones= { ; }
Asignacion= { ; }
SentenciaLlamada= { ; }
Expresion= {], ,, ε,), +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, idMetVar, (, idClase, this, new, ; }
ExpOr= {], ,, ε,), +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, idMetVar, (, idClase, this, new, ; }
ExpOrR= {], ,, ε,), +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, idMetVar, (, idClase, this, new, ; }
ExpAnd= { ε, +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, idMetVar, (, idClase, this, new,], ,,), ; }
ExpAndR= { ε, +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, idMetVar, (, idClase, this, new,], ,,), ; }
ExpIg= { &&, ε }
ExpIgR= { &&, ε }
ExpComp= { ε, ==, != }
ExpCompR= { ε, ==, != }
ExpAd= { ε, <, >, <=, >=, ==, != }
ExpAdR= { ε, <, >, <=, >=, ==, != }
ExpMul= { ε, +, - }
ExpMulR= { ε, +, - }
ExpUn= { ε, *, / }
OpIgual= { +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, idMetVar, (, idClase, this, new }
OpComp= { +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, idMetVar, (, idClase, this, new }
OpAd= { +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, idMetVar, (, idClase, this, new }
OpUn= { +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, idMetVar, (, idClase, this, new }
OpMul= { +, -, !, null, true, false, intLiteral, charLiteral, stringLiteral, idMetVar, (, idClase, this, new }
Operando= { ε, *, / }
Literal= { ε, *, / }
Primario= {), ε, *, / }
MetodoVariable= {), ε, *, / }

ExpresionParentizada= {), ε, *, / }
Encadenado= { ε, ., [, =,), *, / }
Acceso= { ε, ., [, =,), *, / }
AccesoThis= { =,), ε, *, / }
AccesoVar= { = }
LlamadaMetodo= { ε, ., [}
LlamadaMetodoEstatico= {), ε, *, / }
LlamadaCtor= {), ε, *, / }
LlamadaCtorR= {), ε, *, / }
ArgsActuales= { ε, ., [}
ListaExpresiones= {) }
ListaExp= {) }
LlamadaMetodoEncadenado= { ε, ., [, =,), *, / }
AccesoVarEncadenado= { ε, ., [, =,), *, / }
AccesoArregloEncadenado= { ε, ., [, =,), *, / }