**ANEXOS**

**Gramática de MiniJava**

Goal ::= (ImportDeclaration)\* MainClass ( TypeDeclaration )\* <EOF>

ImportDeclaration ::= SingleImportDeclaration |

MultipleImportDeclaration

SingleImportDeclaration::= “import” TypeName “;”

MultipleImportDeclaration::= “import” TypeName “.\*” “;”

TypeName ::= Identifier (“.” Identifier)\*

MainClass ::= “class” Identifier “{” “public” “static” “void” “main” “(” “String” “[" "]” Identifier “)”

“{” PrintStatement “}” “}”

TypeDeclaration ::= ClassDeclaration |

ClassExtendsDeclaration

ClassDeclaration ::= “class” Identifier “{” ( VarDeclaration )\* ( MethodDeclaration )\* “}”

ClassExtendsDeclaration::= “class” Identifier “extends” Identifier “{” ( VarDeclaration )\*

(MethodDeclaration )\* “}”

VarDeclaration ::= Type Identifier “;”

MethodDeclaration ::= “public” Type Identifier “(” ( FormalParameterList )? “)”

“{”(VarDeclaration )\* ( Statement (“;”)\* )\*

“return” Expression “;” “}”

FormalParameterList::= FormalParameter ( FormalParameterRest )\*

FormalParameter::= Type Identifier

FormalParameterRest ::= “,” FormalParameter

Type ::= SimpleType |

ArrayType |

Identifier

SimpleType ::= CharType |  
BooleanType |  
IntegerType |

VoidType |

ObjectType |

ArrayType ::= SimpleType “[" "]“

CharType ::= “char”

BooleanType ::= “boolean”

IntegerType ::= “int”

VoidType ::= “void”

ObjectType ::= “object”

Statement ::= Block |

AssignmentStatement |

ArrayAssignmentStatement |  
IfStatement |

IfElseStatement |  
WhileStatement |

SwitchStatement |  
PrintStatement |

Expression

Block ::= “{” ( Statement )\* “}”

AssignmentStatement ::= Identifier “=” AssigmentRest “;”

ArrayAssignmentStatement ::= Identifier “[" Expression "]” “=” AssigmentRest “;”

AssigmentRest::= “(” Type “)” “System.in.read” “(” “) |

Expression

IfStatement ::= “if” “(” Expression “)” Statement

IfElseStatement ::= “if” “(” Expression “)” Statement “else” Statement

WhileStatement ::= “while” “(” Expression “)” Statement

SwitchStatement ::= “switch” "(" Identifier “)” “{” “case” IntegerType “:”

“{” ( Statement )\* “}” “}”

PrintStatement ::= “System.out.println” “(” Expression “)” “;”

Expression ::= AndExpression |

OrExpression |

EqualExpression |

GreatExpression |  
LessExpression |  
PlusExpression |  
MinusExpression |  
TimesExpression |

SubExpression |  
ArrayLookup |  
ArrayLength |  
MessageSend |  
PrimaryExpression

AndExpression ::= PrimaryExpression “&&” PrimaryExpression

OrExpression ::= PrimaryExpression “||” PrimaryExpression

EqualExpression ::= PrimaryExpression “==” PrimaryExpression

GreatExpression ::= PrimaryExpression “>” PrimaryExpression

LessExpression ::= PrimaryExpression “<” PrimaryExpression

PlusExpression ::= PrimaryExpression “+” PrimaryExpression

MinusExpression ::= PrimaryExpression “-” PrimaryExpression

TimesExpression ::= PrimaryExpression “\*” PrimaryExpression

SubExpression : := PrimaryExpression “/” PrimaryExpression

ArrayLookup ::= PrimaryExpression “[" PrimaryExpression "]“

ArrayLength ::= PrimaryExpression “.” “length”

MessageSend ::= PrimaryExpression “.” Identifier “(” ( ExpressionList )? “)”

ExpressionList ::= Expression ( ExpressionRest )\*

ExpressionRest ::= “,” Expression

PrimaryExpression ::= IntegerLiteral |

CharConstant |

StringConstant |  
TrueLiteral |  
FalseLiteral |  
Identifier |  
ThisExpression |  
ArrayAllocationExpression |  
AllocationExpression |  
NotExpression |  
BracketExpression

IntegerLiteral ::= <INTEGER\_LITERAL>

CharConstant ::= <CHAR\_LITERAL>

StringConstant ::= <STRING\_LITERAL>

TrueLiteral ::= “true”

FalseLiteral ::= “false”

Identifier ::= <IDENTIFIER>

ThisExpression ::= “this”

ArrayAllocationExpression ::= “new” SimpleType “[” Expression “]”

AllocationExpression ::= “new” Identifier BracketExpression

NotExpression ::= “!” Expression

BracketExpression ::= “(” ( Expression )? “)”

**Códigos Ejemplo**

class Factorial {

public static void main(String[] a) {

System.out.println(new Fac().ComputeFac(10));

}

}

class Fac {

public int ComputeFac(int num) {

int num\_aux;

if (num < 1)

num\_aux = 1;

else

num\_aux = num \* (this.ComputeFac(num-1));

return num\_aux;

}

}

import java.awt.\*

import java.awt.event.\*;

class Fac extends Frame

{

Button b1;

Label l1;

TextField t1;

public Fac()

{

ClassActions Acciones;

this.addWindowListener(new Cierre());

this.setTitle("Ejemplo del uso de Interacciones de objetos");

this.setSize(220,400);

b1 = new Button("Acción");

b1.setBounds(470,55,100,30); //setBounds(x,y,ancho,alto)

this.add(b1);

l1 = new Label("TextoSalida");

l1.setBounds(100,55,250,30);

this.add(l1);

l1 = new TextField("TextoEntrada");

l1.setBounds(300,55,250,30);

this.add(l1);

//listaners para los acciones objetos

Acciones = new ClassActions();

aceptar.addActionListener(Acciones);

//hago visible el Frame

this.setVisible(true);

}

public int ComputeFac(int num)

{

int num\_aux;

if (num < 1)

num\_aux = 1;

else

num\_aux = num \* (this.ComputeFac(num-1));

return num\_aux;

}

class ClassActions implements ActionListener

{

public void actionPerformed(ActionEvent evento)

{

Object objeto;

objeto = evento.getSource();

if (objeto == b1)

{ this.b1\_ActionPerformed(evento); }

}

}

public void b1\_ActionPerformed(ActionEvent evento)

{

int res,n;

String temp;

temp = t1.getText();

n = Integer.parseInt(temp);

res = this.ComputeFac(n);

temp = String.valueOf(res);

l1.setText(temp);

}

}

class Cierre extends WindowAdapter

{

public void windowClosing(WindowEvent e)

{

System.exit(0);

}

}

class Factorial {

public static void main(String[] a) {

System.out.println((new Fac()).ComputeFac(10));

}

}

**Enlaces de Interés**

Algunos enlaces tomados de Internet referentes a la tarea:

1. Sitio Web de universidad de la India donde se obtuvo la sintaxis inicial del lenguaje.

<http://www.cse.iitm.ac.in/>

1. Sitios Web de ayudas en el uso de la herramienta JavaCC de universidades de Los Ángeles, USA y de Málaga, España:

<http://cs.lmu.edu/~ray/notes/javacc/>

<http://www.lcc.uma.es/~galvez/theme/IntroduccionAJavaCC.pdf>

1. Sitio Web oficial