Grammar

Goal ::= MainClass ( ClassDeclaration )\* <EOF>

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

ClassDeclaration ::= "class" Identifier ( "extends" Identifier )? "{" ( VarDeclaration )\* ( MethodDeclaration )\* "}"

VarDeclaration ::= Type Identifier ";"

MethodDeclaration ::= ("public" | "private") Type Identifier "(" ( Type Identifier ( "," Type Identifier )\* )? ")" "{" ( VarDeclaration )\* ( Statement )\* "return" Expression ";" "}"

Type ::= "int" Open\_close

| "boolean" Open\_close

| "float" Open\_close

| "String" Open\_close

| "char" Open\_close

Open\_close ::= “[“ “]” | e

Statement ::= "{" ( Statement )\* "}"

| If\_statment

| "while" "(" Expression ")" Statement

| "System.out.println" "(" Expression ")" ";"

| Identifier Identifier-dash

If\_statment::= Matched | Unmatched

Matched::= “if” “(“ Expression “)” Matched “else” Matched

| Statement

Unmatched::= “if” “(“ Expression “)” Unmatched\_dash

Unmatched\_dash::= Statement

| Matched “else” Unmatched

Identifier-dash ::= "=" Expression ";"

| "[" Expression "]" "=" Expression ";"

Expression ::= FINAL Expression-DOUBLE-DASH

| "!" Expression Expression-DOUBLE-DASH

| "(" Expression ")" Expression-DOUBLE-DASH

Expression-DOUBLE-DASH::= Expression-dash Expression-DOUBLE-DASH

| e (lamda or empty)

Expression-dash::= ( "&&" | "<" | "+" | "-" | "\*" ) Expression

| "[" Expression "]"

| "." dot-dash

dot-dash::= "length" | Identifier "(" ( Expression ( "," Expression )\* )? ")"

FINAL ::= <INTEGER\_LITERAL>| "true" | “false” |”this”| Identifier| “new” New-Dash

New-Dash::= “int” “[“ Expression “]” | Identifier “(“ “)”

Identifier ::= <IDENTIFIER>

Sample Program

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 ;

}

}