Trabajo de compiladores

Definición BNF

program ::= class\_declaration | program class\_declaration

identifier ::= (letter|’\_’) (letter | digit | ‘\_’ )\*

letter ::= ‘A’ | ‘B’ | ‘C’ | ‘D’ | ‘E’ | ‘F’ | ‘G’ | ‘H’ | ‘I’ | ‘J’ | ‘K’ | ‘L’ | ‘M’ | ‘N’ | ‘O’ | ‘P’ | ‘Q’ | ‘R’ | ‘S’ | ‘T’ | ‘U’ | ‘V’ | ‘W’ | ‘X’ | ‘Y’ | ‘Z’ | ‘a’ | ‘b’ | ‘c’ | ‘d’ | ‘e’ | ‘f’ | ‘g’ | ‘h’ | ‘i’ | ‘j’ | ‘k’ | ‘l’ | ‘m’ | ‘n’ | ‘o’ | ‘p’ | ‘q’ | ‘r’ | ‘s’ | ‘t’ | ‘u’ | ‘v’ | ‘w’ | ‘x’ | ‘y’ | ‘z’

digit ::= ‘0’ | ‘1’ | ‘2’ | ‘3’ |‘4’ |‘5’ |‘6’ |‘7’ |‘8’ |‘9’

class\_declaration ::= ‘public’ ‘class’ class\_name ‘{‘ optional\_variable\_declaration optional\_method\_declaration ‘}’

class\_object\_creation ::= class\_name object\_name ‘=’ ‘new’ class\_name ‘(‘ ‘)’ ‘;’

class\_name ::= capital\_letter letter\*

capital\_letter ::= ‘A’ | ‘B’ | ‘C’ | ‘D’ | ‘E’ | ‘F’ | ‘G’ | ‘H’ | ‘I’ | ‘J’ | ‘K’ | ‘L’ | ‘M’ | ‘N’ | ‘O’ | ‘P’ | ‘Q’ | ‘R’ | ‘S’ | ‘T’ | ‘U’ | ‘V’ | ‘W’ | ‘X’ | ‘Y’ | ‘Z’

optional\_variable\_declaration ::= optional\_variable\_declaration variable\_declaration | ‘’

optional\_method\_declaration ::= optional\_method\_declaration method\_declaration | ‘’

variable\_declaration ::= [ modifier ] data\_type variable\_name ‘;’

modifier ::= ‘public’ | ‘private’

variable\_name ::= letter (letter | digit | ‘\_’ )\*

object\_name ::= variable\_name

access\_member ::= object ‘.’ class\_member\_name

class\_member\_name ::= variable\_name | method\_name

data\_type ::= ‘int ‘ | ‘char’ | ‘double’ | ‘boolean’

method\_declaration ::= method\_modifier ‘method’ [return\_type] method\_name ‘(‘ [parameters] ‘)’ ‘{‘ optional\_variable\_declaration commands\* ‘}’

method\_modifier ::= ‘public’ | ‘private’

return\_type ::= data\_type | ’void’

method\_name ::= letter (letter | digit)\*

parameters ::= parameter\_declaration | parameters ‘,’ parameter\_declaration

parameter\_declaration ::= data\_type variable\_name

commands ::= assignment\_statement | loop\_statement | control\_statement | print\_statement | return\_statement | loop\_termination\_statement

assignment\_statement ::= variable\_name ‘=’ expression ‘;’

expression ::= compound\_expression | method\_call | class\_object\_creation | literal

literal ::= (letter | digit)\*

compound\_expression ::= ‘(‘ expression ‘)’ | expression ‘+’ expression | expression ‘-‘ expression | expression ‘\*’ expression | expression ‘/’ expression | variable\_name | method\_call | class\_object\_creation

method\_call ::= object\_name ‘.’ method\_name ‘(‘ [parameters] ‘)’ | method\_name ‘(‘ [parameters] ‘)’

loop\_statement ::= ‘do’ ‘{‘ [commands] ‘}’ ‘while’ ‘(‘ condition ‘)’ ‘;’ | ‘for’ ‘(‘ expression1 ‘;’ expression2 ‘;’ expression3 ‘)’ ‘{‘ [commands] ‘}’

condition ::= expression comparative\_logical\_operator expression

comparative\_logical\_expression ::= ‘<’ | ‘>’ | ‘==’ | ‘!=’ | ‘&&’ | ‘||’

expression1 ::= data\_type variable\_name ‘=’ expression

expression2 ::= variable\_name condition expression

expression3 ::= assigment\_statement

control\_statement ::= if\_statement | switch\_statement

if\_statement ::= ‘if’ ‘(‘ condition ‘)’ ‘{‘ [commands] ‘}’ ( ‘else’ ‘if’ ‘(‘ condition ‘)’ ‘{‘ [commands] ‘}’)\* ‘else’ ‘{‘ [commands] ‘}’

switch\_statement ::= ‘switch’ ‘(‘ expression ‘)’ ‘{‘ [ ‘case’ expression ‘:’ [commands]] [‘default’ ‘:’ [commands]] ‘}’

print\_statement ::= ‘out.print’ ‘(‘ ‘”’(letter | digit)\* ‘”’ ‘,’ [variable\_name]\* ‘)’ ‘;’

return\_statement ::= ‘return’ expression ‘;’

loop\_termination ::= ‘break;’