```
TERMINALS: All words, letters, numbers and symbols as +, -, “...
NON TERMINALS: All words between “<>”.
START SYMBOL: <program>.

<program> ::= <class_declaration>+

<class_declaration> ::= 'public' 'class' <class_name> '{' <class_body> '}'

<class_body> ::= <variable_declaration>* <method_declaration>*

<method_declaration> ::= <modifier> <return_type> <identifier> '(' <parameter_list>? ')' '{'
<variable_declaration>* <statement>*  '}'

<parameter_list> ::= <parameter> (',' <parameter>)*
<parameter> ::= <data_type> <identifier>

<variable_declaration> ::= <modifier>? <data_type> (<identifier> (‘,’ <identifier>)* ‘;’ |
<assignment_statement> | <identifier> '=' <expression> (‘,’ <identifier> '=' <expression>) ‘;’)
<modifier> ::= 'public' | 'private'

<class_name> ::= [A-Z] [a-zA-Z0-9_]*
<identifier> ::= [a-zA-Z_] [a-zA-Z0-9_]*

<return_type> ::= ‘void’ | <data_type>
<data_type> ::= 'int' | 'char' | 'double' | 'boolean' | 'String'

<statement> ::= <assignment_statement>
            | <loop_statement>
            | <control_statement>
            | <print_statement>
            | <return_statement>
            | <break_statement>
            | <comment>

<assignment_statement> ::= <identifier> '=' <expression> ';'
<expression> ::= <literal> | <identifier> | <method_call> | <object_creation> |
<compound_expression>
<literal> ::= <number> | <char_literal> | <string_literal> | <boolean_literal>
<number> ::= 0 | (‘+’|’-’)?[1-9][0-9]* |  (‘+’|’-’)?[1-9][0-9]*’.’[0-9][0-9]*’d’
<char_literal> ::= \‘ [a-zA-Z0-9_] \‘ | \’ ‘\n’ \’
<string_literal> ::= ““‘[a-zA-Z0-9_]*’”’
<boolean_literal> ::= 'true' | 'false'
<compound_expression> ::= <expression> ('+' | '-' | '*' | '/') <expression> | ‘(‘ <expression>
('+' | '-' | '*' | '/') <expression> ’)’

<object_creation> ::= 'new' <class_name> '()'

<method_call> ::= <identifier> '.' <identifier> '(' <argument_list>? ')'
```

```
<argument_list> ::= <expression> (',' <expression>)*


<loop_statement> ::= 'do' '{' <statement>* '}' 'while' '(' <condition> ')' ';'
                 | 'for' '(' <data_type> <identifier> ‘=’ <expression> ';' <condition> ';'
<assignment_statement> ')' '{' <statement>* '}'
<condition> ::= <expression> ('>' | '<' | '==' | '!=' | '&&' | '||') <expression>

<control_statement> ::= 'if' '(' <condition> ')' '{' <statement>* '}' ('else if' '(' <condition> ')' '{'
<statement>* '}')* ('else' '{' <statement>* '}')? | 'switch' '(' <expression> ')' '{' <case_clause>*
<default_clause>? '}'
<case_clause> ::= 'case' <expression> ':' <statement>*
<default_clause> ::= 'default' ':' <statement>*

<print_statement> ::= 'out.print(' <string_literal> [ ',' <identifier> ]? ')'

<return_statement> ::= 'return' <expression> ';'

<break_statement> ::= 'break' ';'

<comment> ::= '//' [a-zA-Z0-9_]* | '/*' [a-zA-Z0-9_]* '*/'
```