```
/* imports */

COMPILER program


CHARACTERS

letter    = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz".
digit     = "0123456789".
digitNonZero = "123456789".

cr        = '\r'.
lf        = '\n'.
tab       = '\t'.

ANY = any symbol from the ASCII Character Set
char  = ANY - '"' - '\\' - '\'' - cr - lf.


TOKENS

equal = '='.
greater = '>'.
smaller = '<'.
not = '!'.
AND = "&&".
OR = "||".

commentStart = "/*".
commentEnd = "*/".

number = digitNonZero{digit}| "0".

simpleIdentifier = letter{letter|digit}.
StringValue = '"'char{char}'"'.

charValue = '\''char'\''.

multilineComment = commentStart {char} commentEnd.
singlelineComment = "//" {char} lf.

IGNORE cr + lf + tab


PRODUCTIONS


program
= [packageDeclaration] {packageImport} classDeclaration.

packageDeclaration = "package" identifier.
packageImport = "import" identifier [".*"].
```

```
classDeclaration = "public" "class" simpleIdentifier "{" classBlock "}".

classBlock = {datatypeDeclaration} {methodDeclaration}.

methodDeclaration = "public" "static" ("void"|datatype) simpleIdentifier"("[datatypeDescriptor
{"," datatypeDescriptor}]")" "{" bodyBlock "}".

datatypeDeclaration = "static" ["final"] datatypeDescriptor [ equal ( expression | "new" object
("(" [expression] ")" | "["number"]"))].

assOrMethodCall = identifier (methodCall | assignment).
methodCall = "("[expression {"," expression}]")".
assignment = [arraySelector] equal expression .

bodyBlock = { whileStatement | ifStatement | returnStatement | assOrMethodCall |
datatypeDeclaration  }.

whileStatement = "while" "(" condition ")" "{" bodyBlock "}".
ifStatement = "if" "(" condition ")" "{" bodyBlock "}" [ "else" "{" bodyBlock "}" ].
returnStatement = "return" expression.

datatypeDescriptor = datatype identifier [arraySelector].

identifier = simpleIdentifier {("."simpleIdentifier)}.
arraySelector = "[" [expression] "]".


value = identifier [arraySelector | ("("[expression {"," expression}]")")] | intValue |
charValue | booleanValue | StringValue | "NULL" | not value.
factor = value {('*' | '/' | '%') value}.
term = factor {("+" | "-") factor}.
expression = term {(AND|OR) term}.
condition = expression [(equal equal | not equal | greater [equal] | smaller [equal])
expression].

intValue = ["-"]number.
booleanValue = "true" | "false".

primitive = "int" | "boolean" | "char".
object = "String" | identifier.

datatype = primitive | object.

END program.
```