```
Goal ::= MainClass ( ClassDeclaration )* <EOF>
MainClass ::= "class" Identifier "{" "public" "static" "void" "main" "(" "String" "["
"]" Identifier ")" "{" Statement "}" "}"
ClassDeclaration ::= "class" Identifier ( "extends" Identifier )? "{" ( VarDeclaration
)* (
ConstructorDeclaration )* ( MethodDeclaration )* "}"
VarDeclaration ::= Type Identifier ";"
ConstructorDeclaration::= Identifier
"(" ( Type Identifier ( "," Type Identifier )*)? ")"
"{" ( VarDeclaration )* ( Statement )* "}"
MethodDeclaration ::= ("public" | "private" | "protected") Type Identifier
"(" ( Type Identifier ( "," Type Identifier )*)? ")"
"{" ( VarDeclaration )* ( Statement )* "return" Expression ";" "}"

Type ::= “int” ARR | “Boolean” ARR | “char” ARR | “String” ARR | “Float” ARR
       ARR ::= “[” “]” | lamda
================================================================================

Statement ::= "{" ( Statement )* "}"
| "if" "(" Expression ")" Statement ELSE_PART
| "while" "(" Expression ")" Statement
| "System.out.println" "(" Expression ")" ";"
| Identifier ID
       ELSE_PART ::="else" Statement | lamda
       ID ::= "=" Expression ";" | "[" Expression "]" "=" Expression ";"

================================================================================


Expression ::=
| <INTEGER_LITERAL> EXP_DASH
| <FLOAT_LITERAL> EXP_DASH
| "true" EXP_DASH
| "false" EXP_DASH
| Identifier EXP_DASH
| "this" EXP_DASH
| "new" NEW_DASH EXP_DASH
| "!" Expression EXP_DASH
| "(" Expression ")" EXP_DASH

EXP_DASH -> X EXP_DASH | lamda

X -> ( "&&" | "||" | "==" | "!=" | ">" | "<" |"<=" | ">="| "+" | "-" | "*" | "/" )
Expression
| "[" Expression "]"
| ”.” DOT_DASH

DOT_DASH -> "length" | Identifier "(" ( Expression ( "," Expression )* )? ")"

NEW_DASH -> Identifier "(" (Expression ( "," Expression)*)? ")"
| ("int" | "float" | "String" | "char" | "boolean" ) "[" Expression "]"

Identifier ::= <IDENTIFIER>
```