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

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

-ClassDeclaration ::= "class" Identifier ( "extends" Identifier )? "{" (Declaration)\* "}"

- Declaration ::= VarDeclaration | MethodDeclaration

-VarDeclaration ::= Type Identifier ";"

-MethodDeclaration ::= ("public" | "private") Type Identifier "(" Parameter\*

")" "{" ( VarDeclaration )\* ( Statement )\* "return" Expression ";" "}"

-Parameters ::= Type identifier ( "**,**" Type *identifier* )\*

-Type ::= "boolean" | "int" | "float" | "String" | "char"

( RegularType | ArrayType )

- RegularType::= lamda

-ArrayType ::= “[“ “]”

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

| "if" "(" Expression ")" Statement ("else" Statement) ?

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

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

| Identifier Identifier\_Statement

-Identifier\_Statement ::= "=" Expression ";" | "[" Expression "]" "=" Expression ";"

-Expression ::=

| <INTEGER\_LITERAL> Expression\_Alpha

| "true" Expression\_Alpha

| "false" Expression\_Alpha  
| Identifier Expression\_Alpha  
| "this" Expression\_Alpha  
| "new" new\_Expression Expression\_Alpha  
| "!" Expression Expression\_Alpha  
| "(" Expression ")" Expression\_Alpha

-new\_Expression ::= "int" "[" Expression "]" | Identifier "(" ")"

-Expression\_Alpha ::= BinaryOp Expression Expression\_Alpha  
| "[" Expression "]" Expression\_Alpha   
| "." dot\_Expression Expression\_Alpha

| lamda

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

-BinaryOp ::= "&&" | "<" | "+" | "-" | "\*"

- Identifier ::= <IDENTIFIER>