Grammar After Removing Ambiguity

Goal ::=	MainClass ClassDeclaration` <EOF>
MainClass ::=	“class” Identifier “{“ “public” “static” “void” “main” “(“ “String” “[“ “]” Identifier “)” “{“ Statement “}” “}”
ClassDeclaration` ::=	ClassDeclaration ClassDeclaration` | λ
ClassDeclaration ::=	“class” Identifier Extends` “{“ VarDeclaration` ConstructorDeclaration` MethodDeclaration` “}”
Extends` ::=	“extends” Identifier | λ
VarDeclaration` ::=	VarDeclaration VarDeclaration` | λ
VarDeclaration ::=	Type Identifier “;”
ConstructorDeclaration` ::=	ConstructorDeclaration ConstructorDeclaration` | λ
ConstructorDeclatation ::=	Identifier “(“ TypeIdentifier “)” “{“ VarDeclaration` Statement` “}”
MethodDeclaration` ::=	MethodDeclaration MethodDeclaration` | λ
MethodDeclaration ::=	(“public” | “private | “protected”) Type Identifier “(“ TypeIdentifier “)” “{“ VarDeclaration` Statement` “return” Expression “;” “}”
TypeIdentifier ::=	Type Identifier CommaTypeIdentifier | λ_
CommaTypeIdentifier ::=	“,” Type Identifier CommaTypeIdentifier | λ
Statement` ::=	Statement Statement` | λ
Type ::=	(“int” | “boolean” | “float” | “String” | “char”) Brackets
Brackets ::=	“[“ “]” | λ
Statement ::=	“{“ Statement` “}” | “if” “(“ Expression “)” Statement Else | “while” “(“ Expression “)” Statement | “System.out.println” “(“ Expression “)” “;” | Identifier EqualOrBracketExpression
Else ::=	“else” Statement | λ
EqualOrBracketExpression ::=	“=” Expression “;” | “[“ Expression “]” “=” Expression “;”
Expression ::=	(<INTEGER_LITERAL> | <FLOAT_LITERAL> | “true” | “false” | Identifier | “this” | “new” TypeOrIdentifier | “!” Expression | “(“ Expression “)”) Expression`
TypeOrIdentifier ::=	(“int” | “float” | “String” | “char” | “boolean”) “[“ Expression “]” | Identifier “(“ ExpressionArgu “)”
ExpressionArgu ::=	Expression CommaExpressionArgu | λ

CommaExpressionArgu ::=	“,” Expression CommaExpressionArgu | λ
Expression` ::=	ExpressionItems Expression’ | λ
ExpressionItems::=	("&&" | "||" | "==" | "!=" | ">" | "<" | "<=" | ">=" | "+" | "-" | "*" | "/") Expression | "[" Expression "]" | “.” AfterDot
AfterDot ::=	"length" | Identifier "(" ExpressionArgu ")"
Identifier ::=	<IDENTIFIER>