## 1、MJava的语法规则

Goal-> MainClass { ClassDeclaration } EOF

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

ClassDeclaration->**"class"** Identifier [ **"extends"** Identifier ] **"{"** { VarDeclaration } { MethodDeclaration } **"}"**

VarDeclaration ->Type Identifier **";"**

MethodDeclaration->**"public"** Type Identifier **"("** [ Type Identifier { **","** Type Identifier } ] **")"** **"{"** { VarDeclaration } { Statement } **"return"** Expression **";"** **"}"**

Type->**"int"** **"["** **"]"**|**"boolean"**|**"int"**| Identifier

Statement->**"{"** { Statement } **"}"**|**"if"** **"("** Expression **")"** Statement **"else"** Statement |**"while"** **"("** Expression **")"** Statement |**"System.out.println"** **"("** Expression **")"** **";"**| Identifier **"="** Expression **";"**| Identifier **"["** Expression **"]"** **"="** Expression **";"**

Expression-> Expression ( "&&" | "<" | "+" | "-" | "\*" ) Expression | Expression **"["** Expression **"]"**| Expression **"."** **"length"**| Expression **"."** Identifier **"("** [ Expression { **","** Expression } ] **")"**| IntegerLiteral |**"true"**|**"false"**| Identifier |**"this"**|**"new"** **"int"** **"["** Expression **"]"**|**"new"** Identifier "(" ")"| **"!"** Expression |**"("** Expression **")"**

## 2、语法分析程序实现步骤

### 2.1、消除左递归

MJava主要的递归出现在Expression产生式中

1、改写产生式

Expression-> Expression ( "&&" | "<" | "+" | "-" | "\*" ) Expression

|Expression **"["** Expression **"]"**

|Expression **"."** **"length"**

|Expression **"."** Identifier **"("** [ Expression { **","** Expression } ] **")"**

|IntegerLiteral

|**"true"**

|**"false"**

| Identifier

|**"this"**

|**"new"** **"int"** **"["** Expression **"]"**

|**"new"** Identifier "(" ")"

| **"!"** Expression

|**"("** Expression **")"**

2、多项直接左递归消除

Expression->

IntegerLiteral **A**

|**"true" A**

|**"false" A**

| Identifier **A**

|**"this" A**

|**"new"** **"int"** **"["** **Expression** **"]" A**

|**"new"** Identifier "(" ")" **A**

| **"!"** **Expression** **A**

|**"("** **Expression** **")" A**

A->

( "&&" | "<" | "+" | "-" | "\*" ) **Expression A**

| **"["** **Expression** **"]"** **A**

| **"."** **"length"** **A**

| **"."** Identifier **"("** [ **Expression** { **","** **Expression** } ] **")" A**

| ε

3、重写新的文法G‘[Goal]

Goal-> MainClass { ClassDeclaration } EOF

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

ClassDeclaration->**"class"** Identifier [ **"extends"** Identifier ] **"{"** { VarDeclaration } { MethodDeclaration } **"}"**

VarDeclaration ->Type Identifier **";"**

MethodDeclaration->**"public"** Type Identifier **"("** [ Type Identifier { **","** Type Identifier } ] **")"** **"{"** { VarDeclaration } { Statement } **"return"** Expression **";"** **"}"**

Type->**"int"** **"["** **"]"**|**"boolean"**|**"int"**| Identifier

Statement->**"{"** { Statement } **"}"**|**"if"** **"("** Expression **")"** Statement **"else"** Statement |**"while"** **"("** Expression **")"** Statement |**"System.out.println"** **"("** Expression **")"** **";"**| Identifier **"="** Expression **";"**| Identifier **"["** Expression **"]"** **"="** Expression **";"**

Expression->

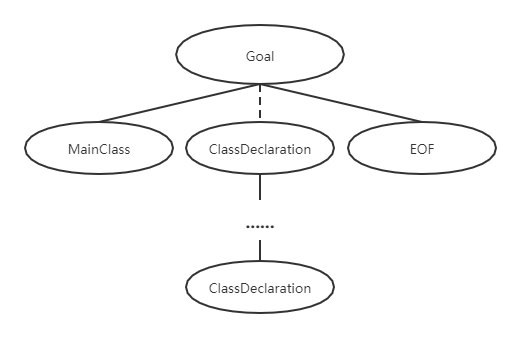
IntegerLiteral **A** |**"true" A** |**"false" A** | Identifier **A** |**"this" A** |**"new"** **"int"** **"["** **Expression** **"]" A** |**"new"** Identifier "(" ")" **A** | **"!"** **Expression** **A** |**"("** **Expression** **")" A**

A->

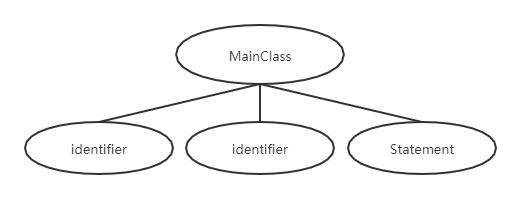
( "&&" | "<" | "+" | "-" | "\*" ) **Expression A** | **"["** **Expression** **"]"** **A** | **"."** **"length"** **A** | **"."** Identifier **"("** [ **Expression** { **","** **Expression** } ] **")" A** | ε

## 2.2抽象语法树

Goal-> MainClass { ClassDeclaration } EOF

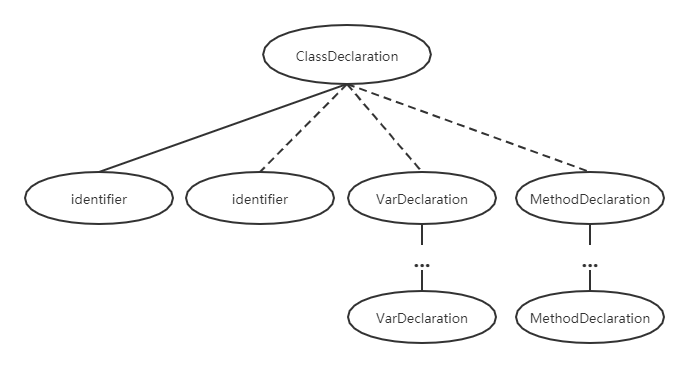


MainClass->**"class"** Identifier **"{"** **"public"** **"static"** **"void"** **"main"** **"("** **"String"** **"["** **"]"** Identifier **")"** **"{"** Statement **"}"** **"}"**

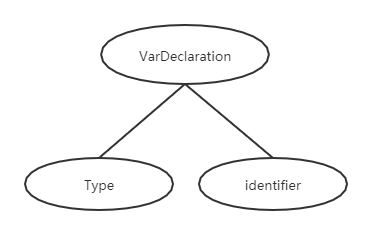


ClassDeclaration->

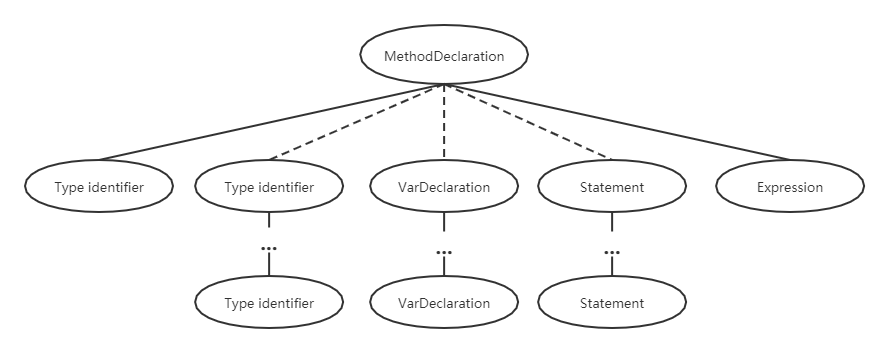
**"class"** Identifier [ **"extends"** Identifier ] **"{"** { VarDeclaration } { MethodDeclaration } **"}"**



VarDeclaration ->Type Identifier **";"**



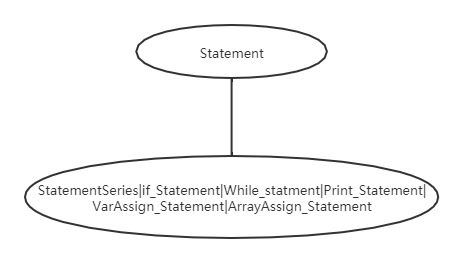
MethodDeclaration->**"public"** Type Identifier **"("** [ Type Identifier { **","** Type Identifier } ] **")"** **"{"** { VarDeclaration } { Statement } **"return"** Expression **";"** **"}"**



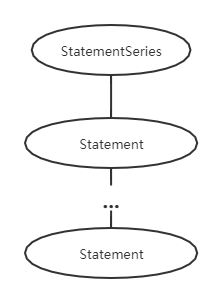
Type->**"int"** **"["** **"]"**|**"boolean"**|**"int"**| Identifier



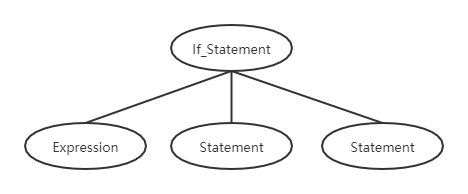
Statement->StatementSeries|if\_Statement|While\_Statement|Print\_Statement|VarAssign\_Statement|ArrayAssign\_Statement



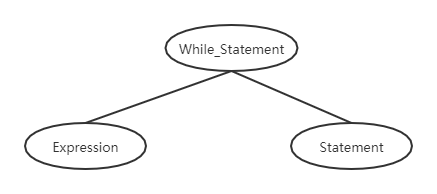
StatementSeries->**"{"** { Statement } **"}"**



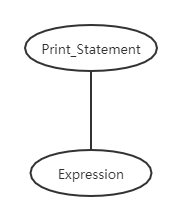
If\_Satement->**"if"** **"("** Expression **")"** Statement **"else"** Statement



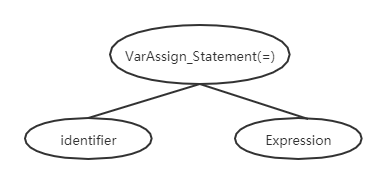
While\_Satement->**"while"** **"("** Expression **")"** Statement



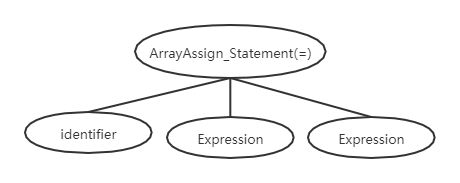
Print\_Statement->**"System.out.println"** **"("** Expression **")"** **";"**

****

VarAssign\_Statement->identifier **"="** Expression **";"**

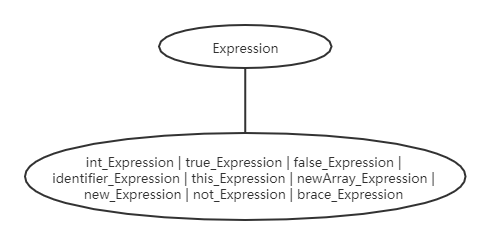
****

ArrayAssign\_Statement->identifier **"["** Expression **"]"** **"="** Expression **";**

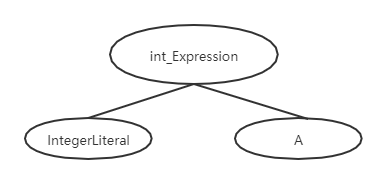


Expression->

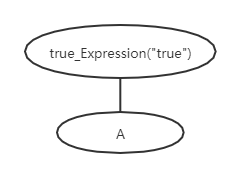
int\_Expression | true\_Expression | false\_Expression | identifier\_Expression | this\_Expression | newArray\_Expression | new\_Expression | not\_Expression | brace\_Expression



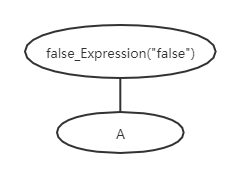
int\_Expression ->IntegerLiteral **A**



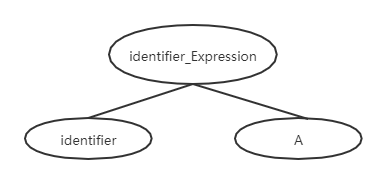
true\_Expression **->"true" A**

****

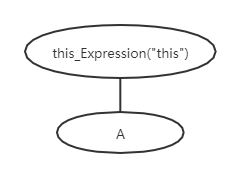
false\_Expression **->"false" A**

****

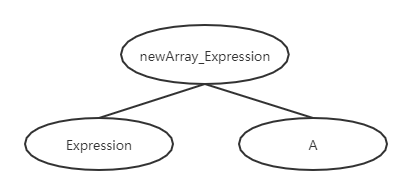
identifier\_Expression ->Identifier **A**

****

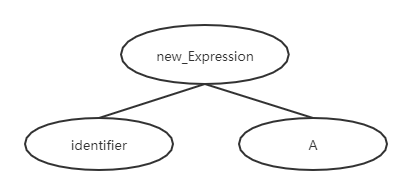
this\_Expression **->"this" A**

****

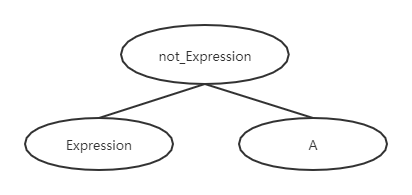
newArray\_Expression **->"new"** **"int"** **"["** **Expression** **"]" A**

****

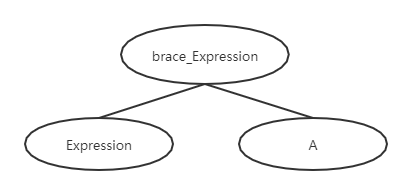
new\_Expression **->"new"** Identifier "(" ")" **A**

****

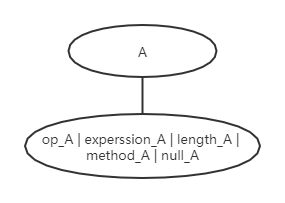
not\_Expression **->"!"** **Expression** **A**

****

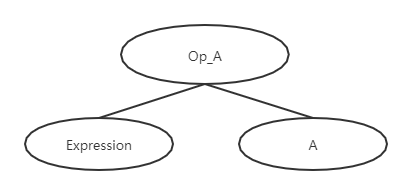
brace\_Expression ->**"("** **Expression** **")" A**

****

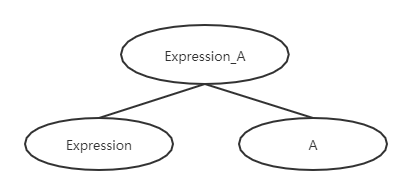
A->op\_A | experssion\_A | length\_A | method\_A | null\_A



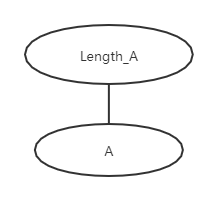
Op\_A ->( "&&" | "<" | "+" | "-" | "\*" ) **Expression A**

****

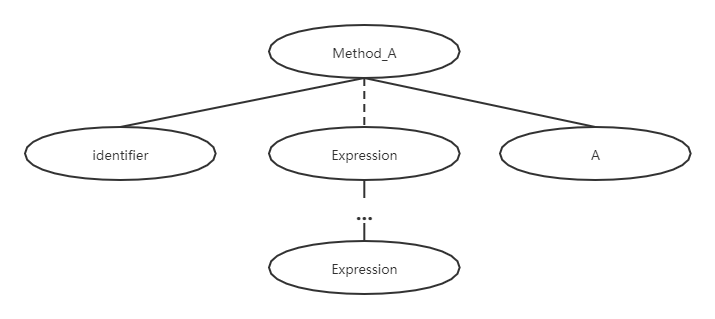
Experssion\_A ->**"["** **Expression** **"]"** **A**

****

Length\_A ->**"."** **"length"** **A**

****

Method\_A->**"."** Identifier **"("** [ **Expression** { **","** **Expression** } ] **")" A**

****

Null\_A->ε

