**通配符运算符和Nongreedy子规则**

(...)?，(...)\*和(...)+是贪婪的，他们消耗的多输入成为可能，但有时这不是什么需要。所以我们需要使用不同的语法：.\*?。我们可以在子规则?，\*或+后缀添加?。

**Nongreedy Lexer Subrules**

这是非常常见的C风格的注释词法分析器规则，它会消耗任何字符，直到它看到尾随\*/：

COMMENT : '/\*' .\*? '\*/' -> skip ; // .\*? matches anything until the first \*/

这是另一个匹配允许“作为转义引号字符的字符串的示例：

grammar Nongreedy;

s : STRING+ ;

STRING : '"' ( '\\"' | . )\*? '"' ; // match "foo", "\"", "x\"\"y"

WS : [ \r\t\n]+ -> skip ;

$ antlr4 Nongreedy.g4

$ javac Nongreedy \* .java

$ grun Nongreedy s -tokens

= > “ quote：\” “

= > EOF

< = [@ 0,0：9 = ' ”引用：\“” '，< 1 >，1：0]

[@ 1,11：10 = ' <EOF> '，< - 1>，2：0]

这是词法分析者选择令牌规则的方式：

1. 主要目标是匹配识别输入最多字符的词法分析器规则。

INT : [0-9]+ ;

DOT : '.' ; // match period

FLOAT : [0-9]+ '.' ; // match FLOAT upon '34.' not INT then DOT

1. 如果多个词法分析器规则与同一输入序列匹配，则优先级将转到语法文件中首先出现的规则。

DOC : '/\*\*' .\*? '\*/' ; // both rules match /\*\* foo \*/

CMT : '/\*' .\*? '\*/' ;

1. Nongreedy子规则匹配最少字符数。

/\*\* Match anything except \n inside of double angle brackets \*/

STRING : '<<' ~'\n'\*? '>>' ; // Input '<<foo>>>>' matches STRING then END

END : '>>' ;

1. 在词法规则中穿越了一个非同意的子规则之后，从那时起所有的决策都是“第一场比赛胜利”。

例如 `.\*? (’a’|’ab’)`是死代码，永远无法匹配。

**Nongreedy Parser Subrules**

这里想要获取A和B：

class A {

String name = "parrt";

}

class B {

int x;

int getDubX() {

return 2\*x;

}

}

这个语法做到了。

grammar Island;

file : clazz\* ;

clazz : 'class' ID '{' ignore '}' ;

ignore : (method|.)\*? ;

method : type ID '()' block ;

type : 'int' | 'void' ;

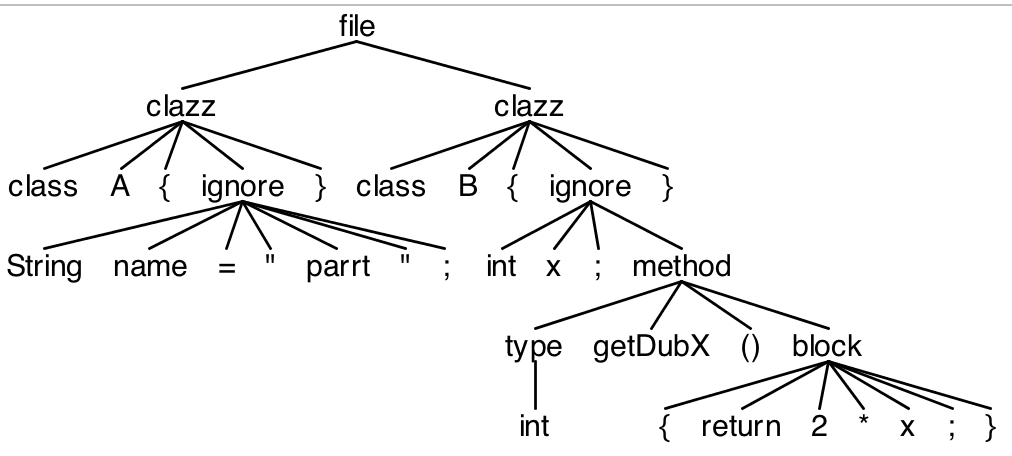
block : '{' (block | .)\*? '}' ;

ID : [a-zA-Z] [a-zA-Z0-9]\* ;

WS : [ \r\t\n]+ -> skip ;

ANY : . ;

你得到：

[](https://github.com/antlr/antlr4/blob/master/doc/images/nonnested-fuzzy.png)

现在让我们尝试一些嵌套类

class A {

String name = "parrt";

class Nested {

any filthy shite we want in here { }}}}}}

}

}

class B {

int x;

int getDubX() {

return 2\*x;

}

}

grammar Island;

file : clazz\* ;

clazz : 'class' ID '{' ignore '}' ;

ignore : (method|clazz|.)\*? ; // <- only change is to add clazz alt here

method : type ID '()' block ;

type : 'int' | 'void' ;

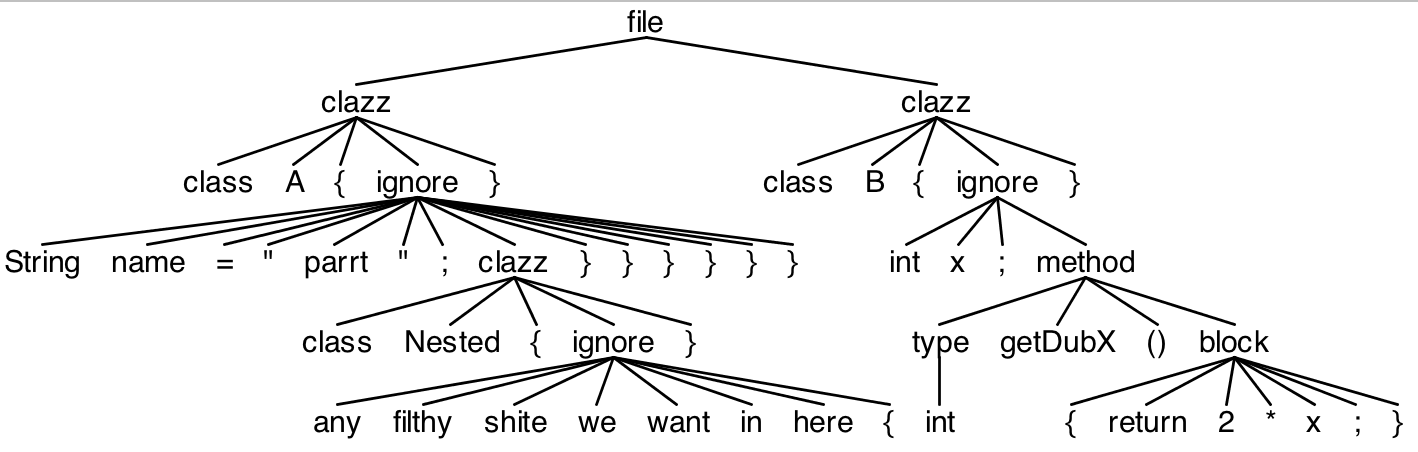
block : '{' (block | .)\*? '}' ;

ID : [a-zA-Z] [a-zA-Z0-9]\* ;

WS : [ \r\t\n]+ -> skip ;

ANY : . ;

你得到：

[](https://github.com/antlr/antlr4/blob/master/doc/images/nested-fuzzy.png)