

一.ANTRL 是什么

ANTLR 是用JAVA写的语言识别工具，它用来声明语言的语法，简称为“元语言”(meta-language)。

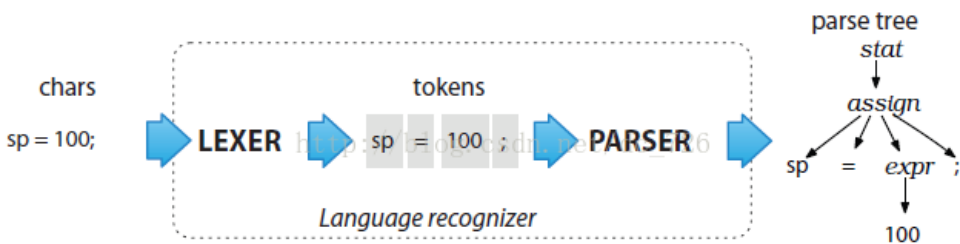
ANTLR 语法识别一般分为二个阶段：

1.词法分析阶段 (lexical analysis)

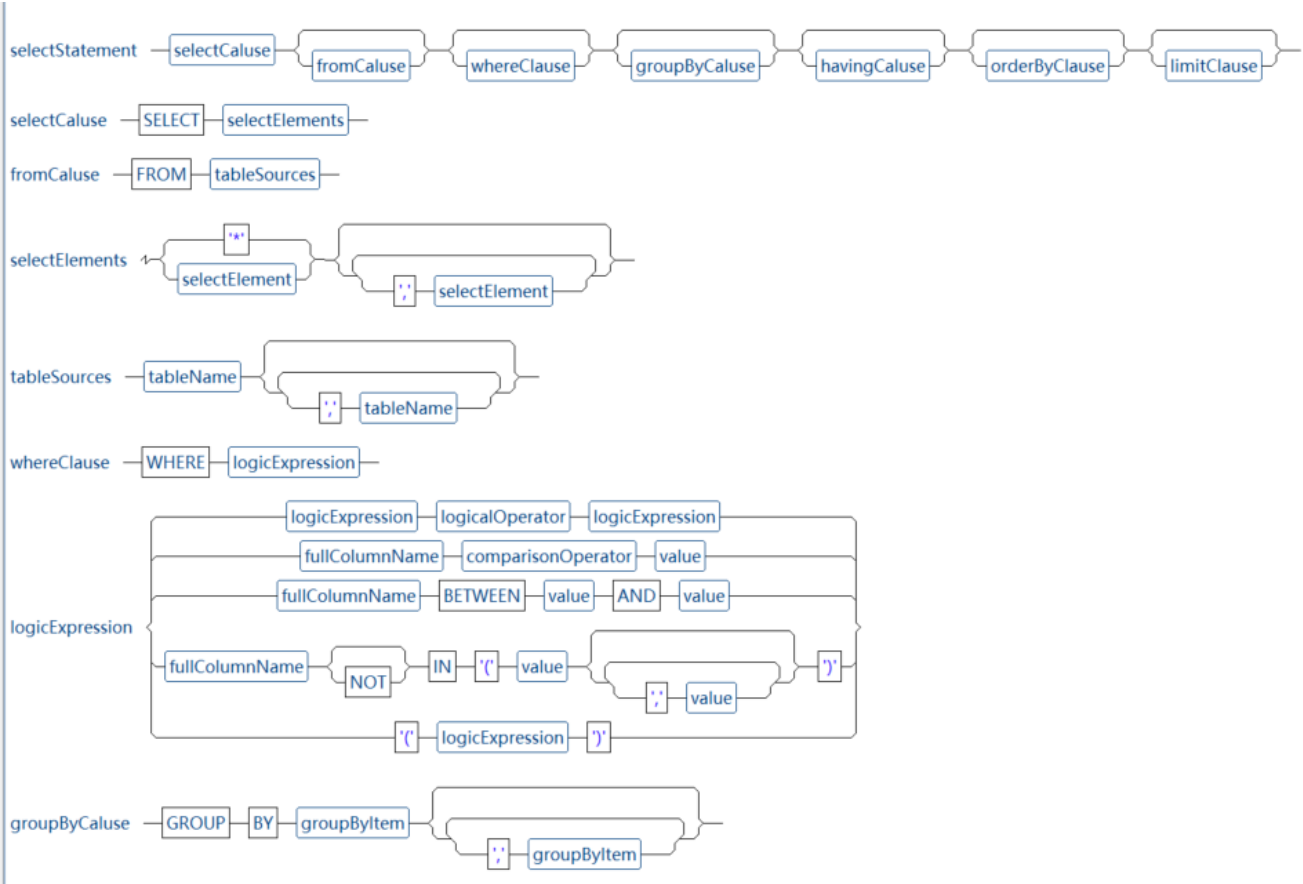
对应的分析程序叫做 **lexer**，负责将符号 (token) 分组成符号类 (token class or token type)

2.解析阶段

根据词法，构建出一棵分析树 (parse tree) 或叫语法树 (syntax tree)



ANTLR 的直观印象，就像是在走迷宫，或者说是电路板更准确，最终只有一条最优路可通达开始与结束，中间的各种叉路与开关，就是我们所编写的规则，下面是我编写的一个SQL查询的简单实现，截取一部分图示：



ANTLR 官方网址 <http://www.antlr.org/>

ANTLR 官方 Github <https://github.com/antlr/antlr4>

大量语法文件例子 <https://github.com/antlr/grammars-v4>

二.主要应用场景

1.定制特定领域语言（DSL）

类似hibernate中的HQL，用DSL来定义要执行操作的高层语法，这种语法接近人可理解的语言，由DSL到计算机语言的翻译则通过ANTLR来做，可在ANTLR的结构语言中定义DSL命令具体要执行何种操作。

2.文本解析 可利用ANTLR解析JSON，HTML，XML，EDIFACT，或自定义的报文格式。解析出来的信息需要做什么处理也可以在结构文件中定义。

3.数学计算 加减乘除，线性方程，几何运算，微积分等等

三.ANTRL 语法

1.结构

```
/** Optional javadoc style comment */
grammar Name;
options {...}
import ... ;

tokens {...}
channels {...} // lexer only
@actionName {...}

rule1 // parser and lexer rules, possibly intermingled
...
ruleN
```

grammar

声明语法头，类似于java类的定义

```
grammar SPL;
```

options

选项，如语言选项，输出选项，回溯选项，记忆选项等等

```
options { output=AST; language=Java; }

options { tokenVocab=MySqlLexer; }
```

@actionName

动作（Actions）实际上是用目标语言写成的、嵌入到规则中的代码（以花括号包裹）。它们通常直接操作输入的标号，但是他们也可以用来调用相应的外部代码。属性，到目前为止我的理解还不多，感觉像是C++中类里面的成员。常用属性或动作说明：

- @header { package com.zetyun.aiops antlr.test; }
这个动作很有用，即在运行脚本后，生成的类中自动带上这个包路径，避免了手动加入的麻烦。
- @members { int i; public TParser(TokenStream input, int foo) { this(input); i = foo; }}
- @after {System.out.println("after matching rule; before finally");}

rule

这是核心，表示规则，以“:”开始，“;”结束，多规则以“|”分隔。

```
ID : [a-zA-Z0-9|'_' ]+ ; //数字
STR: '\'' ( '\\' | ~ ( '\\') ) * '\'' ;
WS: [ \t\n\r ]+ -> skip ; // 系统级规则，即忽略换行与空格
```

```
sqlStatement
: ddlStatement
| dmlStatement      | transactionStatement
| replicationStatement | preparedStatement
| administrationStatement | utilityStatement
;
```

2.注释

- 单行、多行、javadoc风格
- javadoc风格只能在开头使用

```
/**
 * This grammar is an example illustrating the three kinds
 * of comments.
 */
grammar T;

/* a multi-line
   comment
*/

/** This rule matches a declarator for my language */

decl : ID ; // match a variable name
```

3.标识符

- 符号(Token)名大写开头
- 解析规则(Parser rule)名小写开头,后面可以跟字母、数字、下划线

```
ID, LPAREN, RIGHT_CURLY // token names
expr, simpleDeclarator, d2, header_file // rule names
```

四.遍历模式

1、Listener (观察者模式，通过结点监听，触发处理方法)

- 程序员不需要显示定义遍历语法树的顺序，实现简单
- 缺点，不能显示控制遍历语法树的顺序
- 动作代码与文法产生式解耦，利于文法产生式的重用
- 没有返回值，需要使用map、栈等结构在节点间传值

2、Visitor (访问者模式，主动遍历)

- 程序员可以显示定义遍历语法树的顺序
- 不需要与antlr遍历类ParseTreeWalker一起使用，直接对tree操作
- 动作代码与文法产生式解耦，利于文法产生式的重用
- visitor方法可以直接返回值，返回值的类型必须一致，不需要使用map这种节点间传值方式，效率高

五.示例

计算器之四则运算（官方例子照抄）

1、新建g4文件，如Math.g4

```
grammar Math;
```

```
@header{package com.zetyun.aiops.core.math;}

prog : stat+;

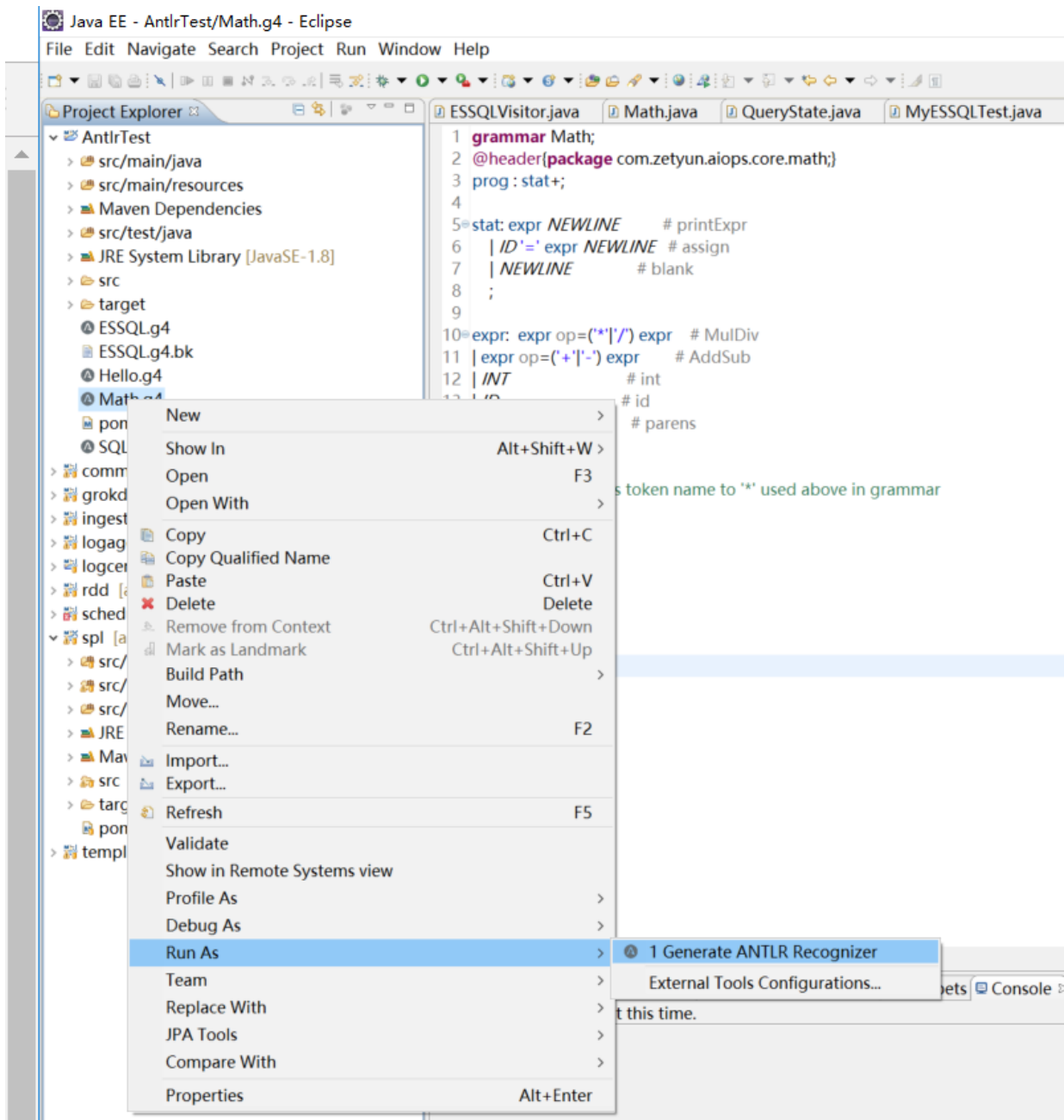
stat: expr NEWLINE      # printExpr
    | ID '=' expr NEWLINE # assign
    | NEWLINE           # blank
    ;

expr: expr op=('*' | '/') expr # MulDiv
    | expr op=('+' | '-') expr # AddSub
    | INT                     # int
    | ID                     # id
    | '(' expr ')'           # parens
    ;

MUL : '*' ; // assigns token name to '*' used above in grammar
DIV : '/' ;
ADD : '+' ;
SUB : '-' ;
ID  : [a-zA-Z]+ ;
INT : [0-9]+ ;
NEWLINE: '\r'? '\n' ;
WS  : [ \t]+ -> skip;
```




2、运行Math.g4文件，生成.java文件



如果没有安装eclipse antlr插件的话，根据如下指示操作即可：

help -> Eclipse Marketplace -> 搜索 Antlr -> 选中合适版本，安装即可。


 Eclipse Marketplace

Eclipse Marketplace

Select solutions to install. Press Finish to proceed with installation.
Press the information button to see a detailed overview and a link to more information.

SearchRecentPopularInstalledEclipse Newsletter: Boot Build Eclips...


Find: antlrAll MarketsAll CategoriesGo

**ANTLR 4 IDE 0.3.6**

ANTLR 4.x Advanced Syntax Highlighting (even for target language) Automatic Code Generation (on save) Manual Code Generation (through External Tools menu) Code... [more info](#)

by Edgar Espina, EPL
[antlr](#) [antlr4](#) [antlr4 v4](#)

★ 20Installs: **38.0K** (698 last month)UpdateUninstall


**AntlrDT 4.3.1**

Contributes an ANTLR V4 grammar editor and builder to the Eclipse platform. Features Antlr Editor and Outline View – full syntax-directed editor Grammar... [more info](#)

by Certiv Analytics, EPL
[antlr](#) [parser](#) [editor](#) [DSL](#)

★ 3Installs: **3.20K** (120 last month)Install

Marketplaces



?

< Back

Install Now >

Finish

Cancel

3、编写测试文件，检验语法，验证结果等

```
package com.zetyun.aiops.test;

import org.antlr.v4.runtime.CharStream;
import org.antlr.v4.runtime.CharStreams;
import org.antlr.v4.runtime.CommonTokenStream;
import org.antlr.v4.runtime.tree.ParseTree;

import com.zetyun.aiops.core.math.MathLexer;
import com.zetyun.aiops.core.math.MathParser;

public class Math {
```

```
public static void main(String[] args) {  
  
    CharStream input = CharStreams.fromString("12*2+12\r\n");  
    MathLexer lexer=new MathLexer(input);  
    CommonTokenStream tokens = new CommonTokenStream(lexer);  
    MathParser parser = new MathParser(tokens);  
    ParseTree tree = parser.prog(); // parse  
    MathVisitorTest vt=new MathVisitorTest();  
    vt.visit(tree);  
  
}  
  
}
```



4、语法树分析

- 1) 下载antlr4运行包, 这里我选择的版本是 antlr-4.7-complete.jar
<http://www.antlr.org/download.html>
- 2) 新建运行脚本 antlr4.bat 和 grun.bat, 放置于任意目录, 如 E:/tools/antlr4
antlr4.bat 内容:

```
java org.antlr.v4.Tool %*
```

grun.bat 内容:

```
java org.antlr.v4.gui.TestRig %*
```

注: antlr依赖于java, 如果java环境变量没有设置, 请先行设置好。

- 3) 设置antlr4的系统环境变量(classpath 和 path)
classpath:

环境变量

Zetyun 的用户变量(U)

变量	值
OneDrive	C:\Users\Zetyun\OneDrive
Path	C:\Users\Zetyun\AppData\Local\Microsoft\WindowsApps;
TEMP	C:\Users\Zetyun\AppData\Local\Temp
TMP	C:\Users\Zetyun\AppData\Local\Temp

编辑系统变量

变量名(N):

CLASSPATH

变量值(V):

%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar;E:\tools\antlr4\antlr-4.7-complete.jar;

浏览目录(D)...

浏览文件(F)...

确定

取消

configsetroot	C:\WINDOWS\ConfigSetRoot
DriverData	C:\Windows\System32\Drivers\DriverData
JAVA_HOME	E:\tools\jdk8
MAVEN_HOME	E:\tools\maven3
NUMBER_OF_PROCESSORS	4
OS	Windows NT

新建(W)...

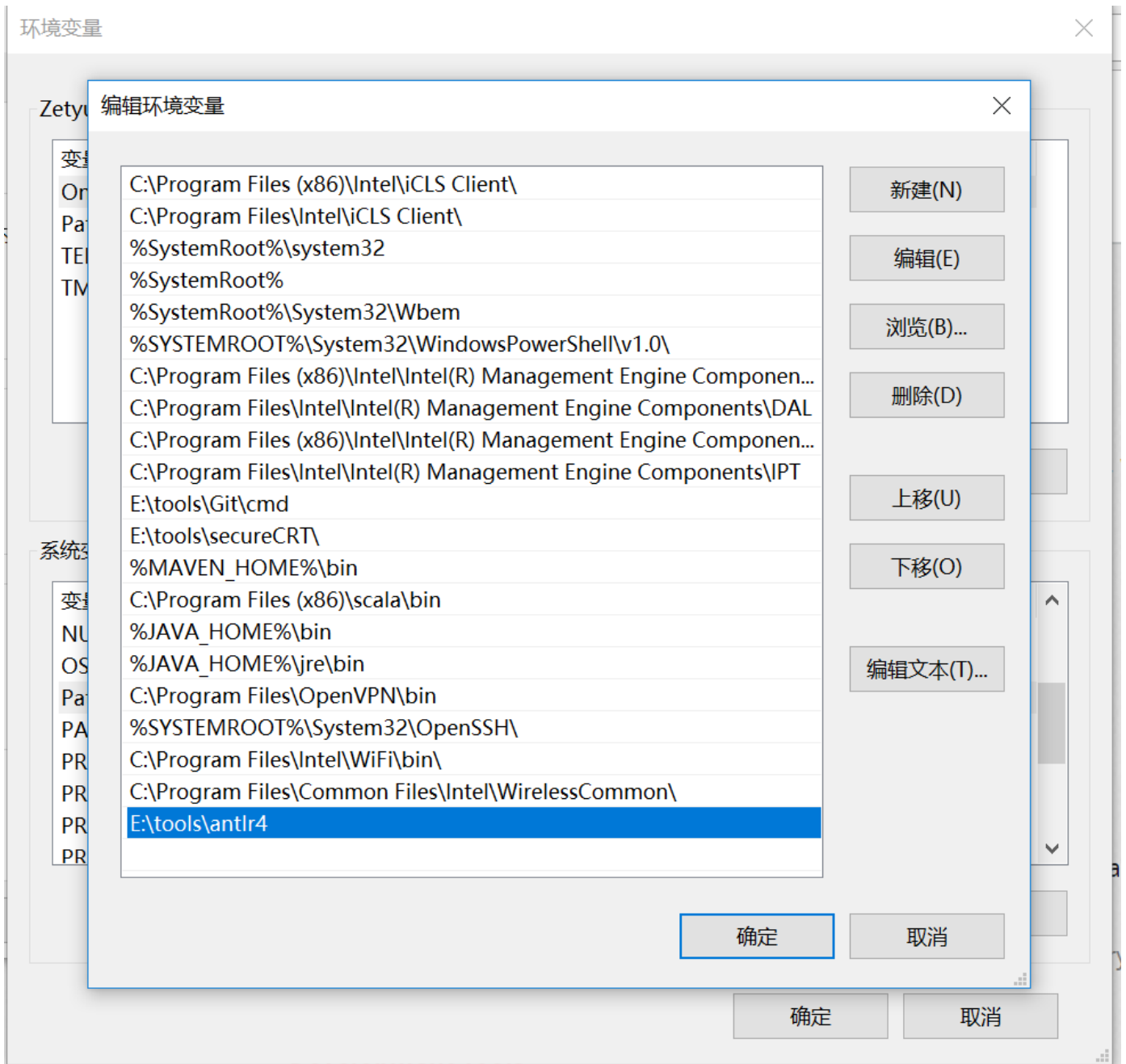
编辑(I)...

删除(L)

确定

取消

path (.bat所在目录):



- 4) 选择要分析的g4文件，运行命令生成相关java文件与token文件

```
D:\workspace\AntlrTest\test\math> antlr4 Math.g4
```

此电脑 > develop (D:) > workspace > AntlrTest > test > math					搜索"math"
名称	修改日期	类型	大小		
Math.g4	2018/5/25 11:14	G4 文件	1 KB		
Math.tokens	2018/5/25 11:35	TOKENS 文件	1 KB		
MathBaseListener.java	2018/5/25 11:35	JAVA 文件	4 KB		
MathLexer.java	2018/5/25 11:35	JAVA 文件	5 KB		
MathLexer.tokens	2018/5/25 11:35	TOKENS 文件	1 KB		
MathListener.java	2018/5/25 11:35	JAVA 文件	4 KB		
MathParser.java	2018/5/25 11:35	JAVA 文件	16 KB		

- 5) 编译java文件

```
D:\workspace\AntlrTest\test\math> javac ./*.java
```

此电脑 > develop (D:) > workspace > AntlrTest > test > math					▼ ↺ 搜
名称	修改日期	类型	大小		
Math.tokens	2018/5/25 11:35	TOKENS 文件	1 KB		
MathBaseListener.class	2018/5/25 11:38	CLASS 文件	3 KB		
MathBaseListener.java	2018/5/25 11:35	JAVA 文件	4 KB		
MathLexer.class	2018/5/25 11:38	CLASS 文件	5 KB		
MathLexer.java	2018/5/25 11:35	JAVA 文件	5 KB		
MathLexer.tokens	2018/5/25 11:35	TOKENS 文件	1 KB		
MathListener.class	2018/5/25 11:38	CLASS 文件	2 KB		
MathListener.java	2018/5/25 11:35	JAVA 文件	4 KB		
MathParser\$AddSubContext.class	2018/5/25 11:38	CLASS 文件	2 KB		
MathParser\$AssignContext.class	2018/5/25 11:38	CLASS 文件	2 KB		
MathParser\$BlankContext.class	2018/5/25 11:38	CLASS 文件	1 KB		
MathParser\$ExprContext.class	2018/5/25 11:38	CLASS 文件	1 KB		
MathParser\$IdContext.class	2018/5/25 11:38	CLASS 文件	1 KB		
MathParser\$IntContext.class	2018/5/25 11:38	CLASS 文件	1 KB		
MathParser\$MulDivContext.class	2018/5/25 11:38	CLASS 文件	2 KB		
MathParser\$ParensContext.class	2018/5/25 11:38	CLASS 文件	1 KB		
MathParser\$PrintExprContext.class	2018/5/25 11:38	CLASS 文件	2 KB		
MathParser\$ProgContext.class	2018/5/25 11:38	CLASS 文件	2 KB		
MathParser\$StatContext.class	2018/5/25 11:38	CLASS 文件	1 KB		
MathParser.class	2018/5/25 11:38	CLASS 文件	10 KB		
MathParser.java	2018/5/25 11:35	JAVA 文件	16 KB		

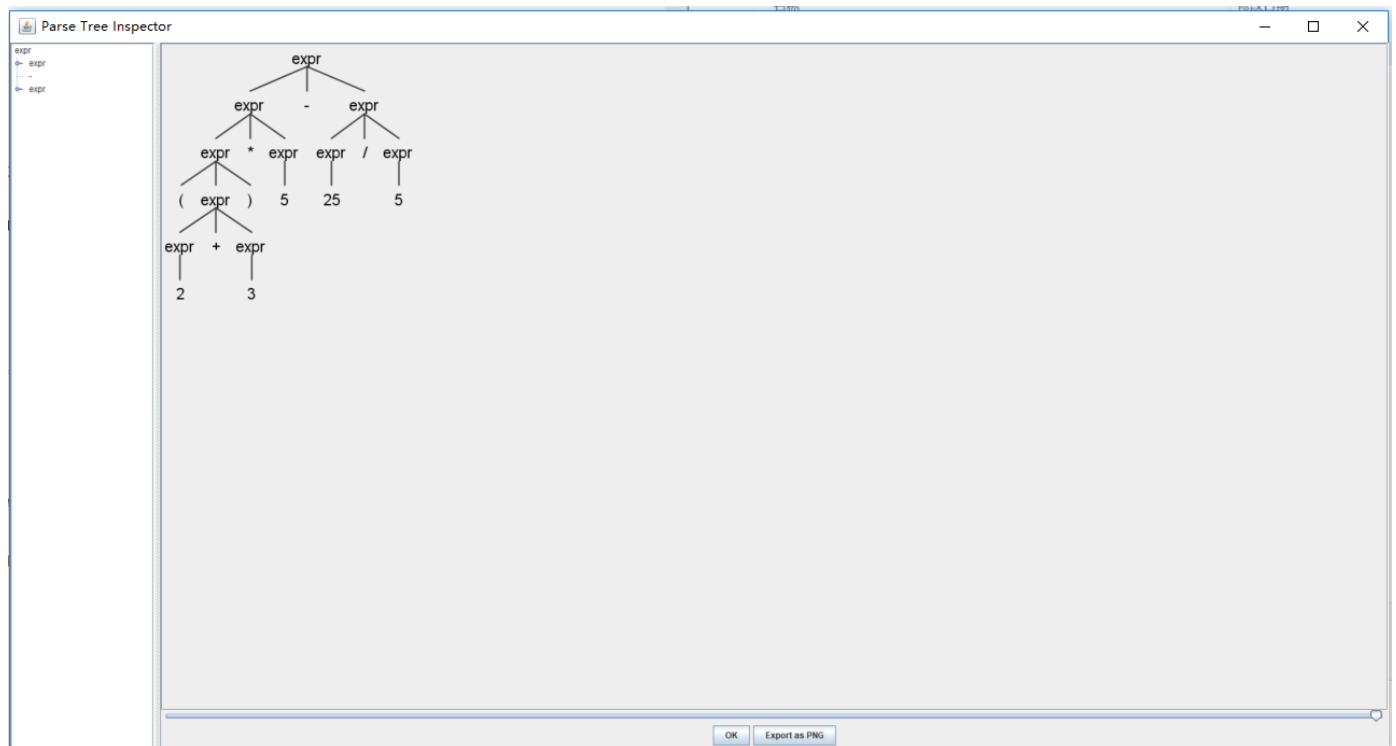
6) 分析语法树

输入grun命令回车，在命令行输入你要测试的语法，再回车，按Ctrl+z 后回车。

```
D:\workspace\AntlrTest\test\math>grun Math expr -gui

D:\workspace\AntlrTest\test\math>java org.antlr.v4.gui.TestRig Math expr -gui
(2+3)*5-25/5
^Z
```

执行完命令后，会出现GUI窗口：



通过语法树，我们可以直观的知道语法是否正确，以便随时调整。