ANTLR

What is ANTLR?

- ANTLR (ANother Tool for Language Recognition) is a powerful parser generator for reading, processing, executing, or translating structured text or binary files. It's widely used to build languages, tools, and frameworks.
- From a grammar, ANTLR generates a parser that can build and walk parse trees.

Setup

- We will introduce 2 ways to use antlr:
  - First one using only cmd/PowerShell.
  - Second one using intellj plugins.
    - There are another plugins, for example for eclipse (there is a link that might help in the references).

Setup:

- We should first:
  - Download java.
  - Download ANTLR.
    - https://www.antlr.org/download/

First Method

Setup:

- Create folder (C:\Javalib).
- Create in this folder 2 files:
  - antlr4.bat
  - grun.bat
- Also put the antlr.zip in this folder.

antlr4.bat

- Write in this file:

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

- Save this file.

grun.bat

- Write in this file:

```
@ECHO OFF
SET TEST_CURRENT_DIR=%CLASSPATH:.;=%
if "%TEST_CURRENT_DIR%" == "%CLASSPATH%" ( SET CLASSPATH=.;%CLASSPATH% )
@ECHO ON
java org.antlr.v4.gui.TestRig %*
```

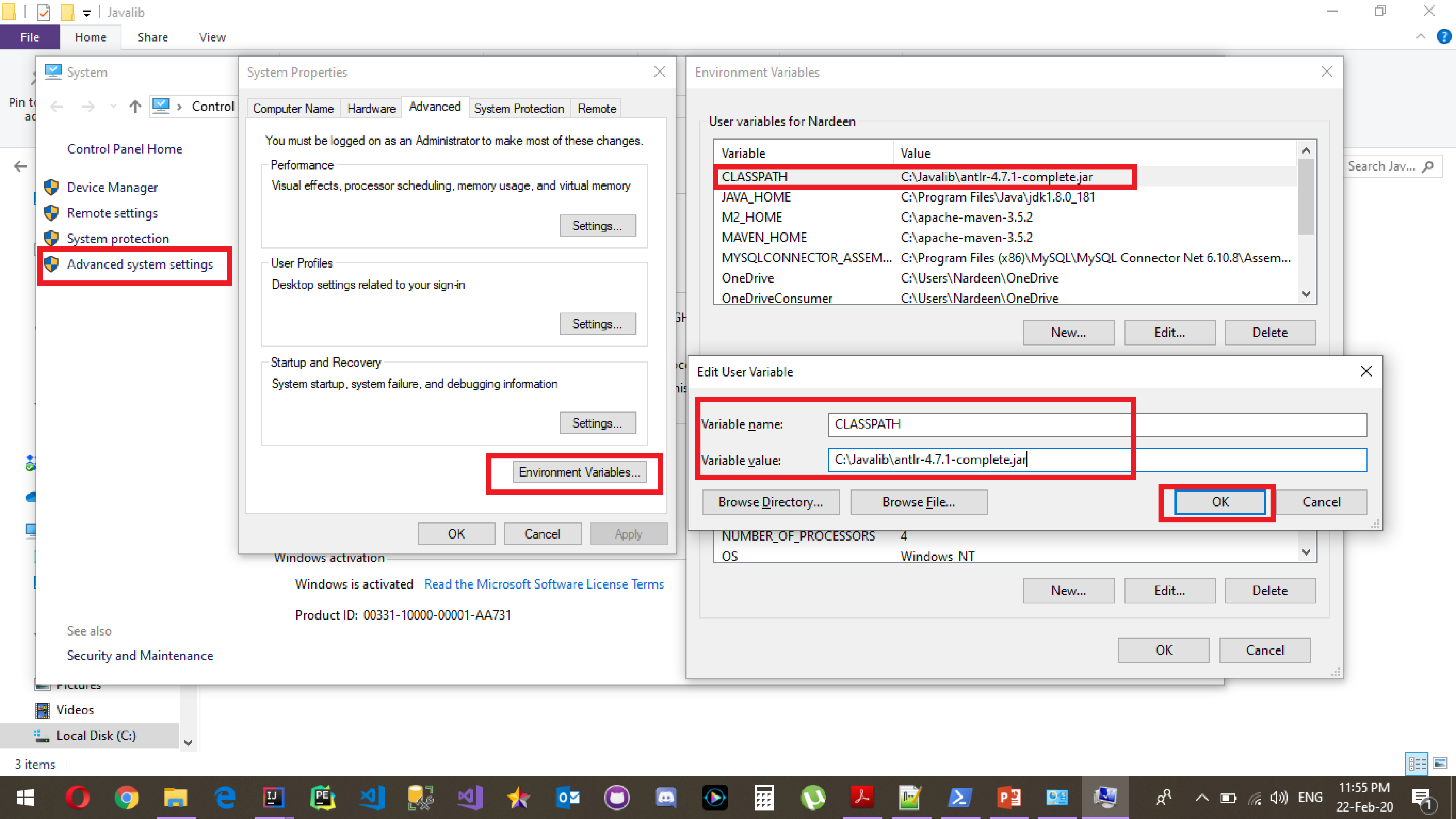
- Save this file.

Setup:

- Open File Explorer.
- Right Click on This PC.
- Select Properties.
- Select Advanced System Settings (from the left hand side).
- Select Environment Variable (at the bottom of the screen).
- At User Variable:
  - Click New.

    Variable Name: CLASSPATH

    Variable Value: C:\Javalib\antlr-4.7.1-complete.jar //antlr path
  - Click Ok



System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...

User Profiles
Desktop settings related to your sign-in
Settings...

Startup and Recovery
System startup, system failure, and debugging information
Settings...

Environment Variables...

OK | Cancel | Apply

Windows activation
Windows is activated Read the Microsoft Software License Terms
Product ID: 00331-10000-00001-AA731

Environment Variables

User variables for Nardeen

Variable	Value
CLASSPATH	C:\Javalib\antlr-4.7.1-complete.jar
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_181
M2_HOME	C:\apache-maven-3.5.2
MAVEN_HOME	C:\apache-maven-3.5.2
MYSQLCONNECTOR_ASSEM...	C:\Program Files (x86)\MySQL\MySQL Connector Net 6.10.8\Assem...
OneDrive	C:\Users\Nardeen\OneDrive
OneDriveConsumer	C:\Users\Nardeen\OneDrive

New... | Edit... | Delete

Edit User Variable

Variable name: CLASSPATH
Variable value: C:\Javalib\antlr-4.7.1-complete.jar

Browse Directory... | Browse File... | OK | Cancel

NUMBER_OF_PROCESSORS	4
OS	Windows_NT

New... | Edit... | Delete

OK | Cancel

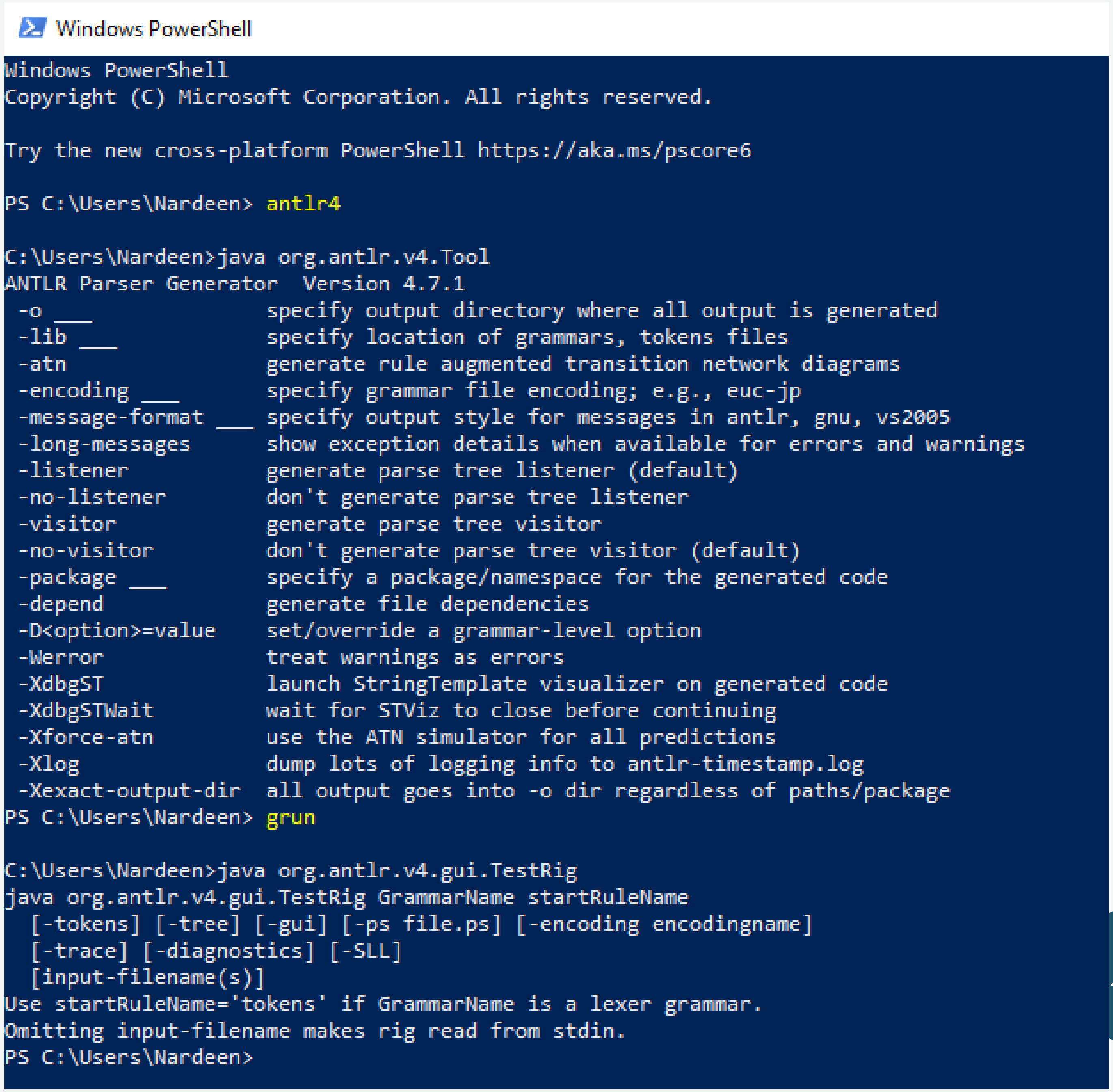
Setup:

- At User Variable: (Continue)
  - Choose Variable : Path
  - Then Click Edit.
  - Then Click New:

  C:\Javalib
  - Then Click New:
  - Also make sure that there is a variable for your java jdk in user variable, if not: then add it.

Make sure that everything works correctly

- Open PowerShell/cmd.
- Write these commands:
  - antlr4
  - grun



Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Nardeen> antlr4

C:\Users\Nardeen>java org.antlr.v4.Tool
ANTLR Parser Generator  Version 4.7.1
 -o ___              specify output directory where all output is generated
 -lib ___            specify location of grammars, tokens files
 -atn                generate rule augmented transition network diagrams
 -encoding ___       specify grammar file encoding; e.g., euc-jp
 -message-format ___ specify output style for messages in antlr, gnu, vs2005
 -long-messages      show exception details when available for errors and warnings
 -listener           generate parse tree listener (default)
 -no-listener        don't generate parse tree listener
 -visitor            generate parse tree visitor
 -no-visitor         don't generate parse tree visitor (default)
 -package ___        specify a package/namespace for the generated code
 -depend             generate file dependencies
 -D<option>=value    set/override a grammar-level option
 -Werror             treat warnings as errors
 -XdbgST             launch StringTemplate visualizer on generated code
 -XdbgSTWait         wait for STViz to close before continuing
 -Xforce-atn         use the ATN simulator for all predictions
 -Xlog               dump lots of logging info to antlr-timestamp.log
 -Xexact-output-dir  all output goes into -o dir regardless of paths/package
PS C:\Users\Nardeen> grun

C:\Users\Nardeen>java org.antlr.v4.gui.TestRig
java org.antlr.v4.gui.TestRig GrammarName startRuleName
  [-tokens] [-tree] [-gui] [-ps file.ps] [-encoding encodingname]
  [-trace] [-diagnostics] [-SLL]
  [input-filename(s)]
Use startRuleName='tokens' if GrammarName is a lexer grammar.
Omitting input-filename makes rig read from stdin.
PS C:\Users\Nardeen>
```

Example 1

- Create a new file: Hello.g4 in your code folder.
- Add your grammar in it, then save it.

```
grammar Hello;
r : 'hello' ID ;   // match keyword hello followed by an identifier
ID : [a-z]+ ;    // match lower-case identifiers
WS : [ \t\r\n]+ -> skip ;   // skip spaces, tabs, newlines
```

- Note: <u>The File name mush be the same as your grammar</u>

Return to the cmd/PoweShell

- Cd to your code folder where you placed the grammar, for example:

    cd F:\GUC\5.2\Example_1

- Then write the command :

    antlr4 Hello.g4

- Then Compile your grammar:

    javac Hello*.java

- Then Write:

    grun Hello r –tokens

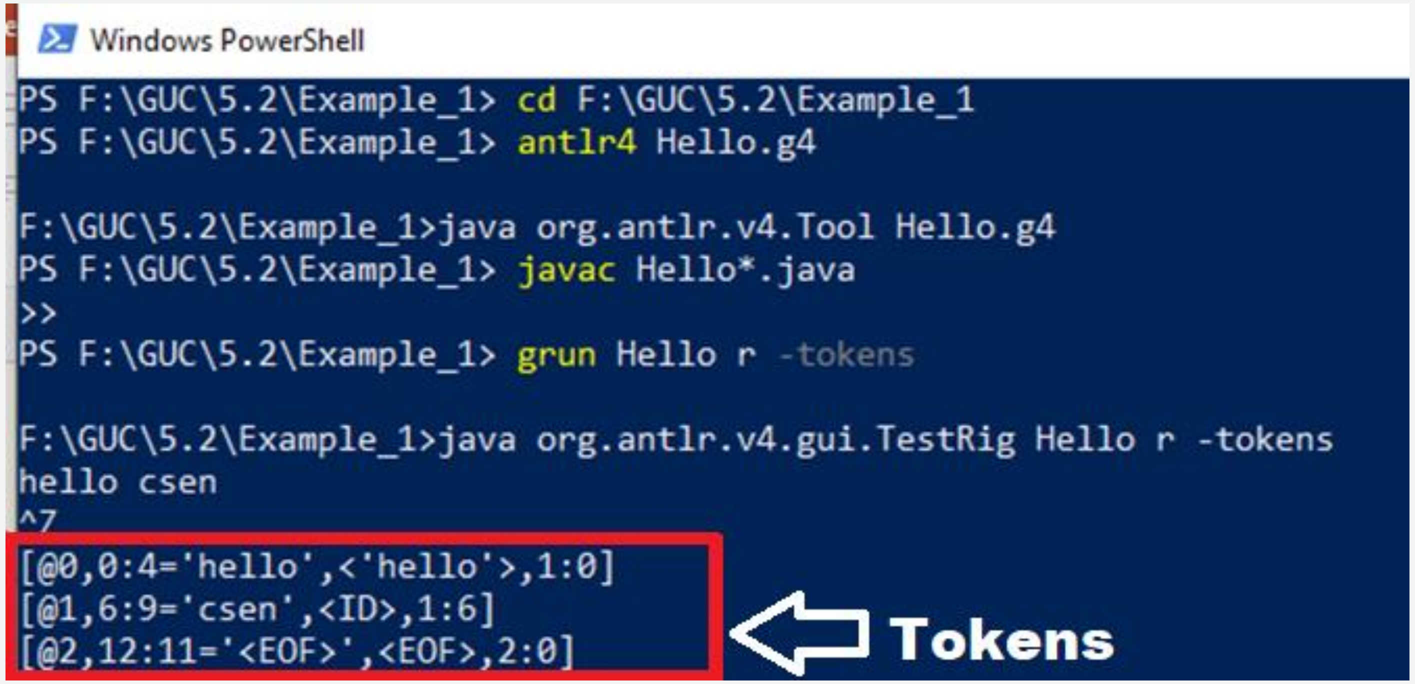
    // r is the start of your grammar

    //token for tokenize the input

- Then you write any string, for example:

    hello csen   // any string

    ^Z   //Press Ctril+Z (EOF) in Windows



```
PS F:\GUC\5.2\Example_1> cd F:\GUC\5.2\Example_1
PS F:\GUC\5.2\Example_1> antlr4 Hello.g4

F:\GUC\5.2\Example_1>java org.antlr.v4.Tool Hello.g4
PS F:\GUC\5.2\Example_1> javac Hello*.java
>>
PS F:\GUC\5.2\Example_1> grun Hello r -tokens

F:\GUC\5.2\Example_1>java org.antlr.v4.gui.TestRig Hello r -tokens
hello csen
^Z
[@0,0:4='hello',<'hello'>,1:0]
[@1,6:9='csen',<ID>,1:6]
[@2,12:11='<EOF>',<EOF>,2:0]
```

Actions

- Actions are blocks of text written in the target language and enclosed in curly braces. The recognizer triggers them according to their locations within the grammar.
- For example, the following rule emits "decl" after the parser has seen a valid declaration:

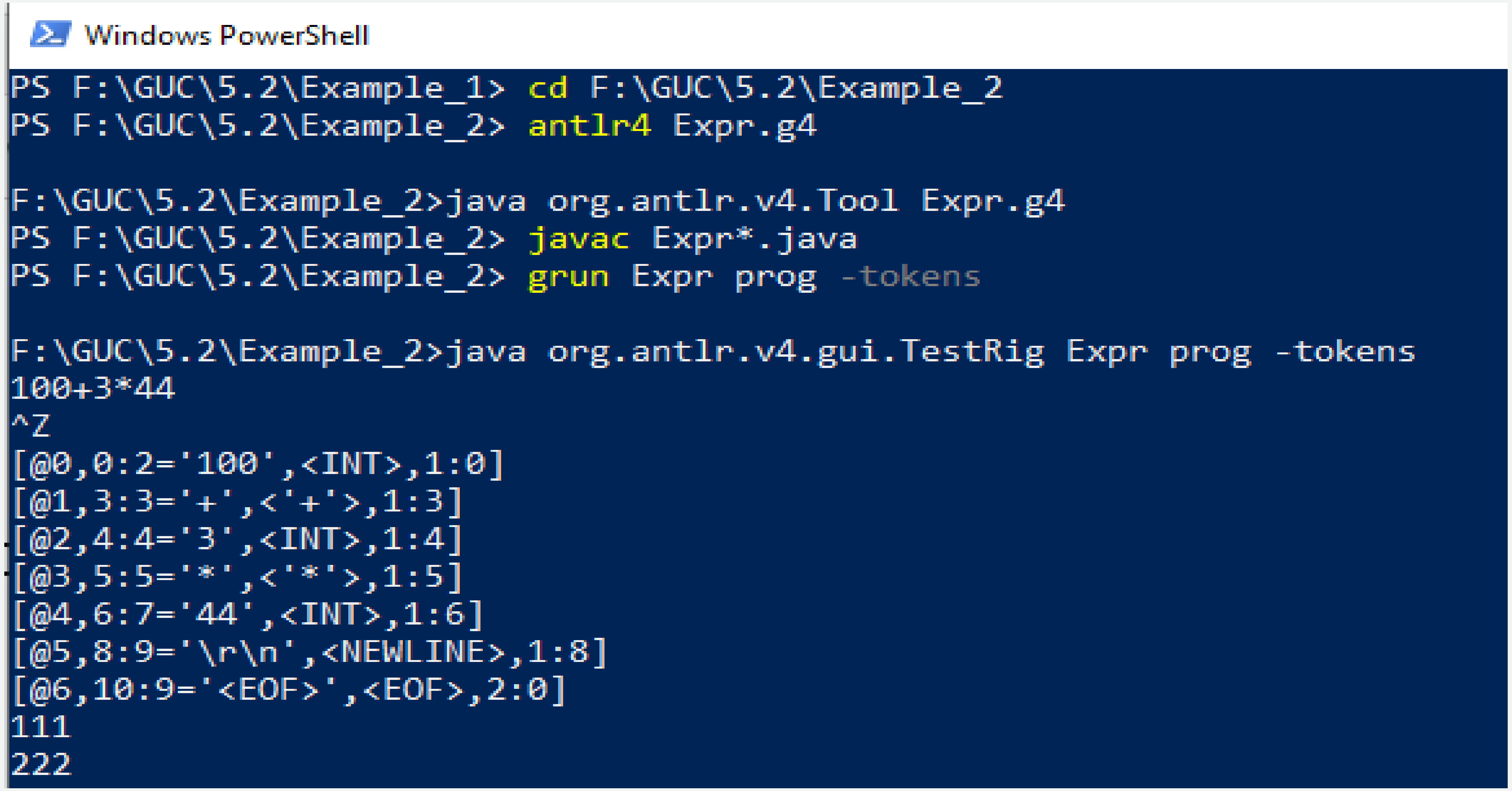
```
decl: type ID ';' {System.out.println("decl");} ;
type: 'int' | 'float' ;
```

- Only actions within the outermost token rule are executed.

Example 2

```
grammar Expr;
prog: (expr NEWLINE)*;
expr: expr('*'|'/')expr{System.out.println("111");}
| expr('+'|'-')expr{System.out.println("222");}
| INT
|'('expr')';
NEWLINE: [\r\n]+;
INT: [0-9]+;
```

Example 2 Test:

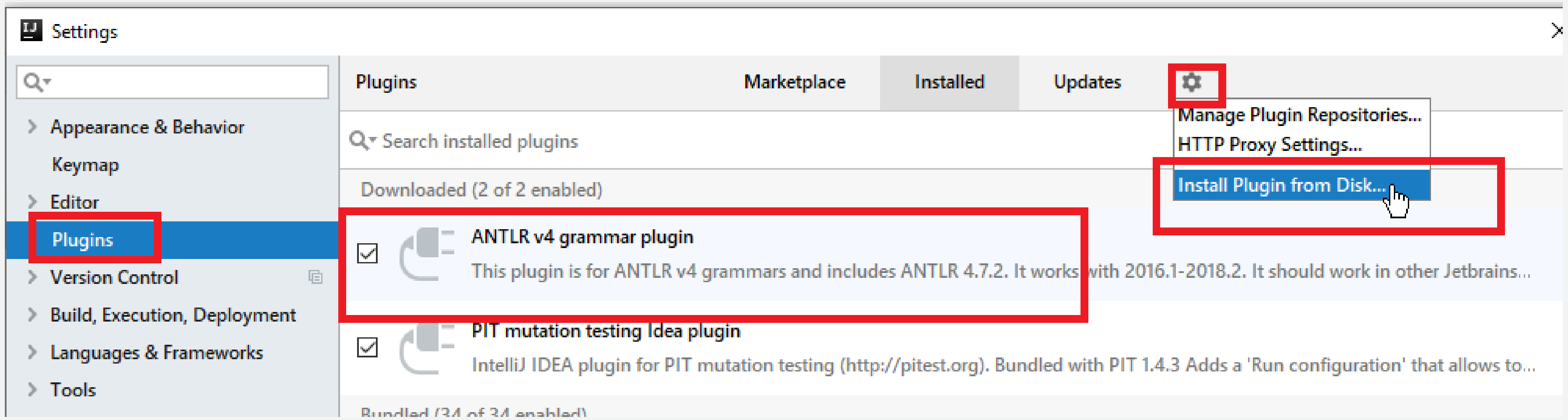


```
PS F:\GUC\5.2\Example_1> cd F:\GUC\5.2\Example_2
PS F:\GUC\5.2\Example_2> antlr4 Expr.g4

F:\GUC\5.2\Example_2>java org.antlr.v4.Tool Expr.g4
PS F:\GUC\5.2\Example_2> javac Expr*.java
PS F:\GUC\5.2\Example_2> grun Expr prog -tokens

F:\GUC\5.2\Example_2>java org.antlr.v4.gui.TestRig Expr prog -tokens
100+3*44
^Z
[@0,0:2='100',<INT>,1:0]
[@1,3:3='+',<'+'>,1:3]
[@2,4:4='3',<INT>,1:4]
[@3,5:5='*',<'*'>,1:5]
[@4,6:7='44',<INT>,1:6]
[@5,8:9='\r\n',<NEWLINE>,1:8]
[@6,10:9='<EOF>',<EOF>,2:0]
111
222
```

Second Method

Setup:

- We should first:
  - Download antlr intellj plugins:
    - https://plugins.jetbrains.com/plugin/7358-antlr-v4-grammar-plugin/versions
- Open a new project, then you should add the plugin in intellj:
  - File > Settings > Plugins > Install Plugins form disk (at the top right), the choose the file.



ANTLR v4 grammar plugin
This plugin is for ANTLR v4 grammars and includes ANTLR 4.7.2. It works with 2016.1-2018.2. It should work in other Jetbrains...

PIT mutation testing Idea plugin
IntelliJ IDEA plugin for PIT mutation testing (http://pitest.org). Bundled with PIT 1.4.3 Adds a 'Run configuration' that allows to...

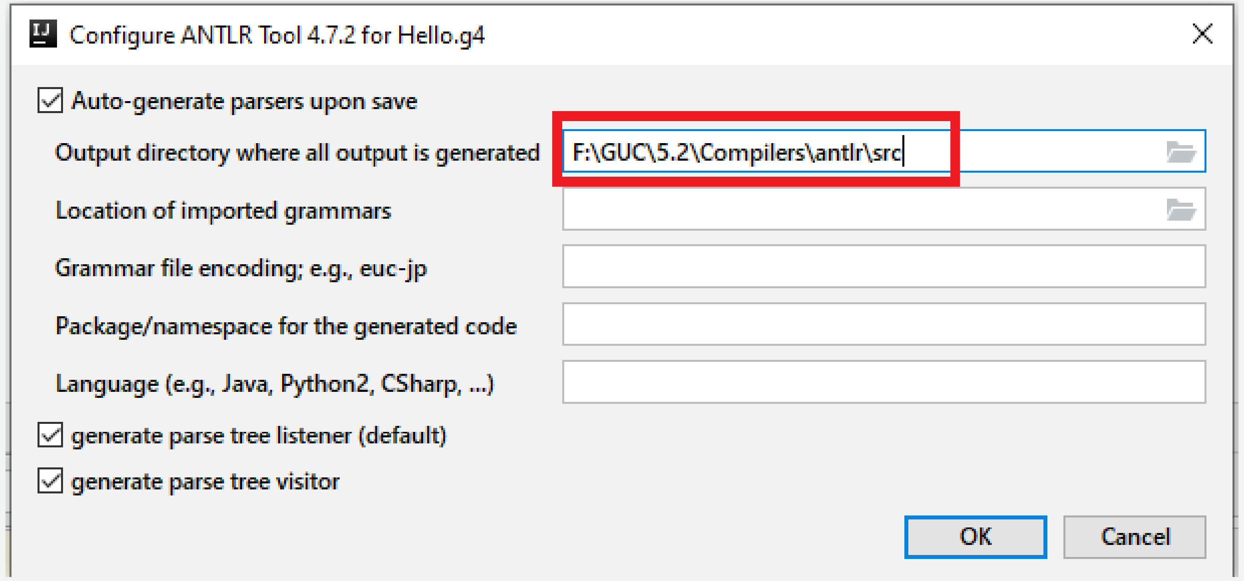
Setup:

- In src folder:
  - Create your grammar file: Hello.g4
- Add your grammar in it: (Like example 1)

```
grammar Hello;
r : 'hello' ID ;   // match keyword hello followed by an identifier
ID : [a-z]+ ;    // match lower-case identifiers
WS : [ \t\r\n]+ -> skip ;   // skip spaces, tabs, newlines
```

Setup:

- On the grammar, Right Click then choose Configure Antlr.
- Write the dir of the src (as shown in the figure).
- Then Right Click then choose Generate Antlr Regonizer.


Configure ANTLR Tool 4.7.2 for Hello.g4

☑ Auto-generate parsers upon save

Output directory where all output is generated	F:\GUC\5.2\Compilers\antlr\src
Location of imported grammars	
Grammar file encoding; e.g., euc-jp	
Package/namespace for the generated code	
Language (e.g., Java, Python2, CSharp, ...)	

☑ generate parse tree listener (default)

☑ generate parse tree visitor

Go to the cmd/PoweShell

- Cd to your code folder where you placed the grammar, for example:

  cd F:\GUC\5.2\Example_1
- Then write the command :

  antlr4 Hello.g4
- Then Compile your grammar:

  javac Hello*.java
- Then Write:

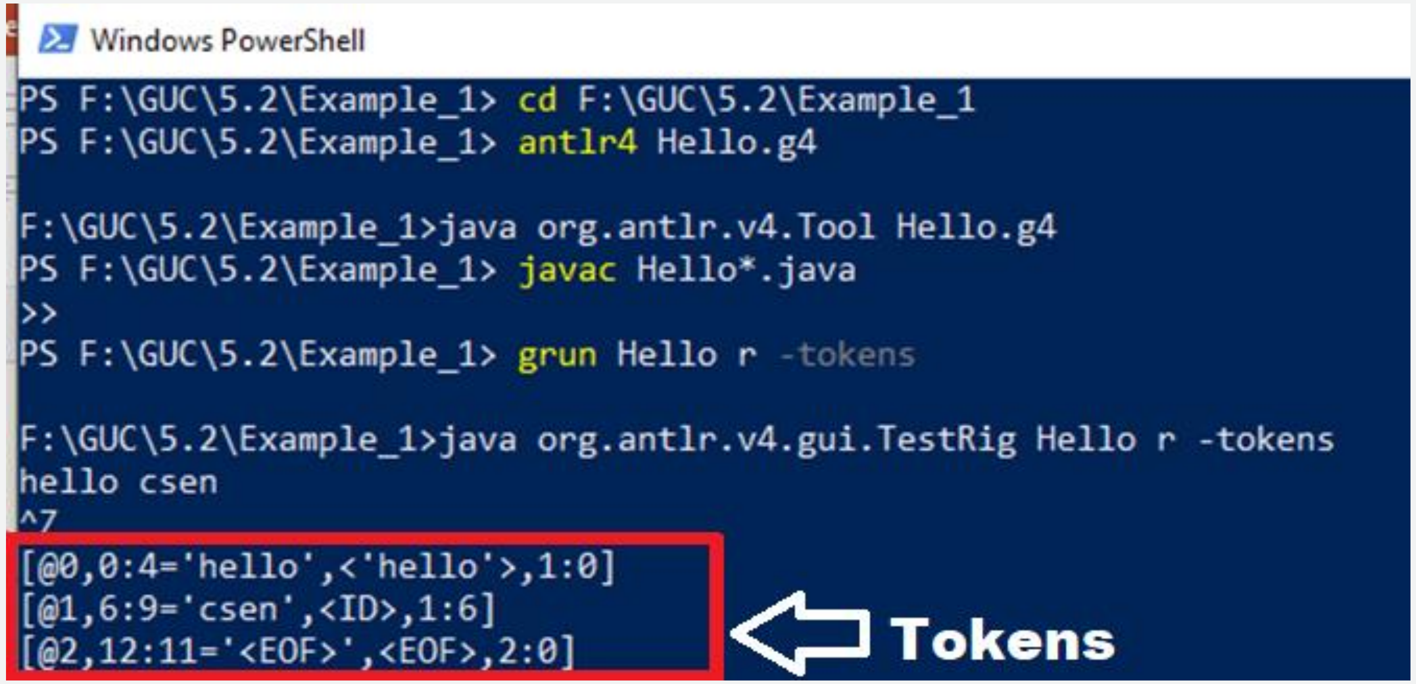
  grun Hello r –tokens

  // r is the start of your grammar

  //token for tokenize the input
- Then you write any string, for example:

  hello csen // any string

  ^Z //Press Ctril+Z (EOF) in Windows

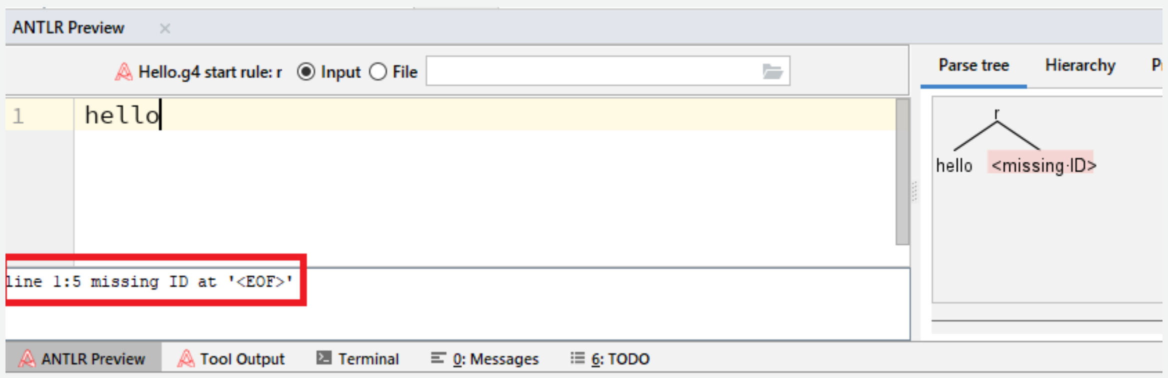


Windows PowerShell

```
PS F:\GUC\5.2\Example_1> cd F:\GUC\5.2\Example_1
PS F:\GUC\5.2\Example_1> antlr4 Hello.g4

F:\GUC\5.2\Example_1>java org.antlr.v4.Tool Hello.g4
PS F:\GUC\5.2\Example_1> javac Hello*.java
>>
PS F:\GUC\5.2\Example_1> grun Hello r -tokens

F:\GUC\5.2\Example_1>java org.antlr.v4.gui.TestRig Hello r -tokens
hello csen
^Z
[@0,0:4='hello',<'hello'>,1:0]
[@1,6:9='csen',<ID>,1:6]
[@2,12:11='<EOF>',<EOF>,2:0]
```

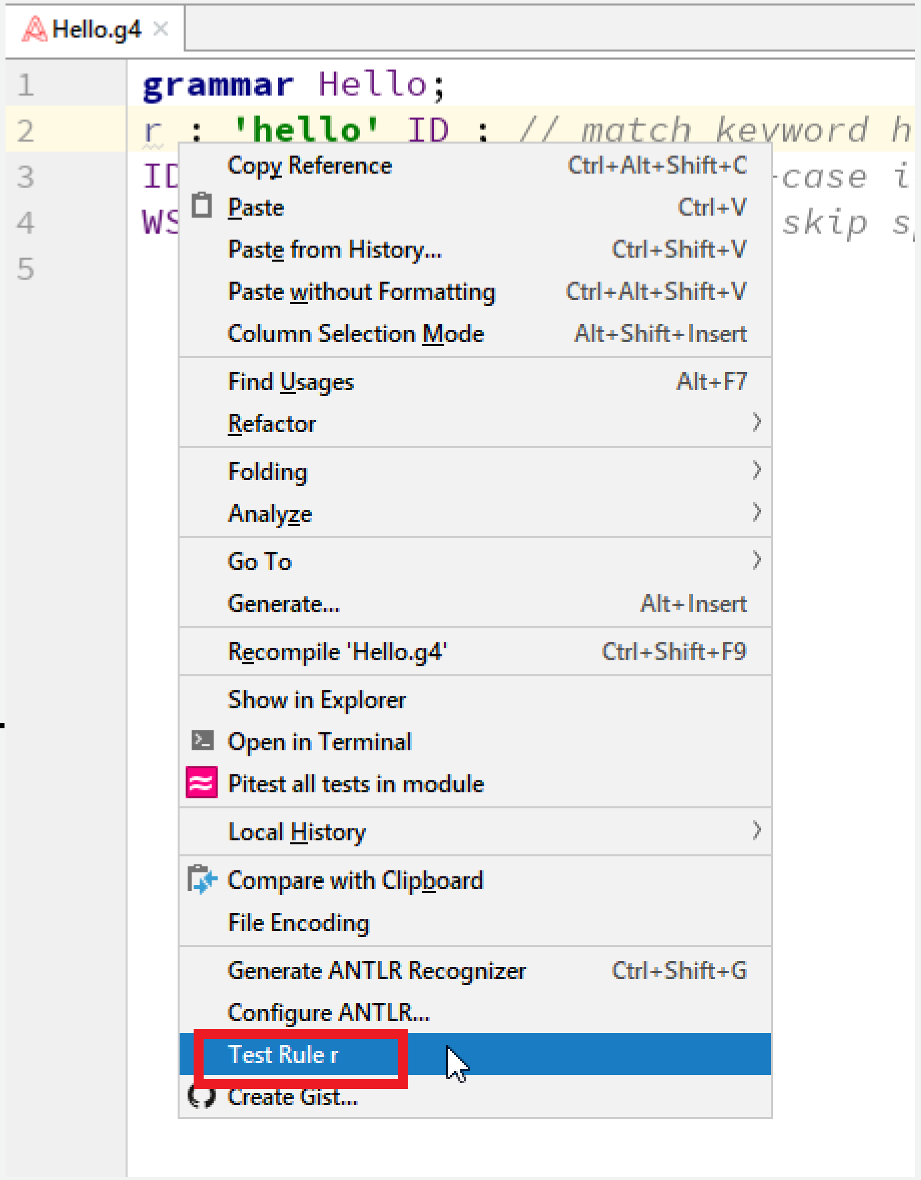
Back to intellj

- We can also run the grammar and see the parse tree:
  - Right click on the start of your regular expression in the grammar, then choose Test Rule r.
- Now Antlr Prview is opened at the bottom, you can write any string, and see its parse tree.
- Also if there is an error in tokens, it will appear at the bottom.



```
hello
```

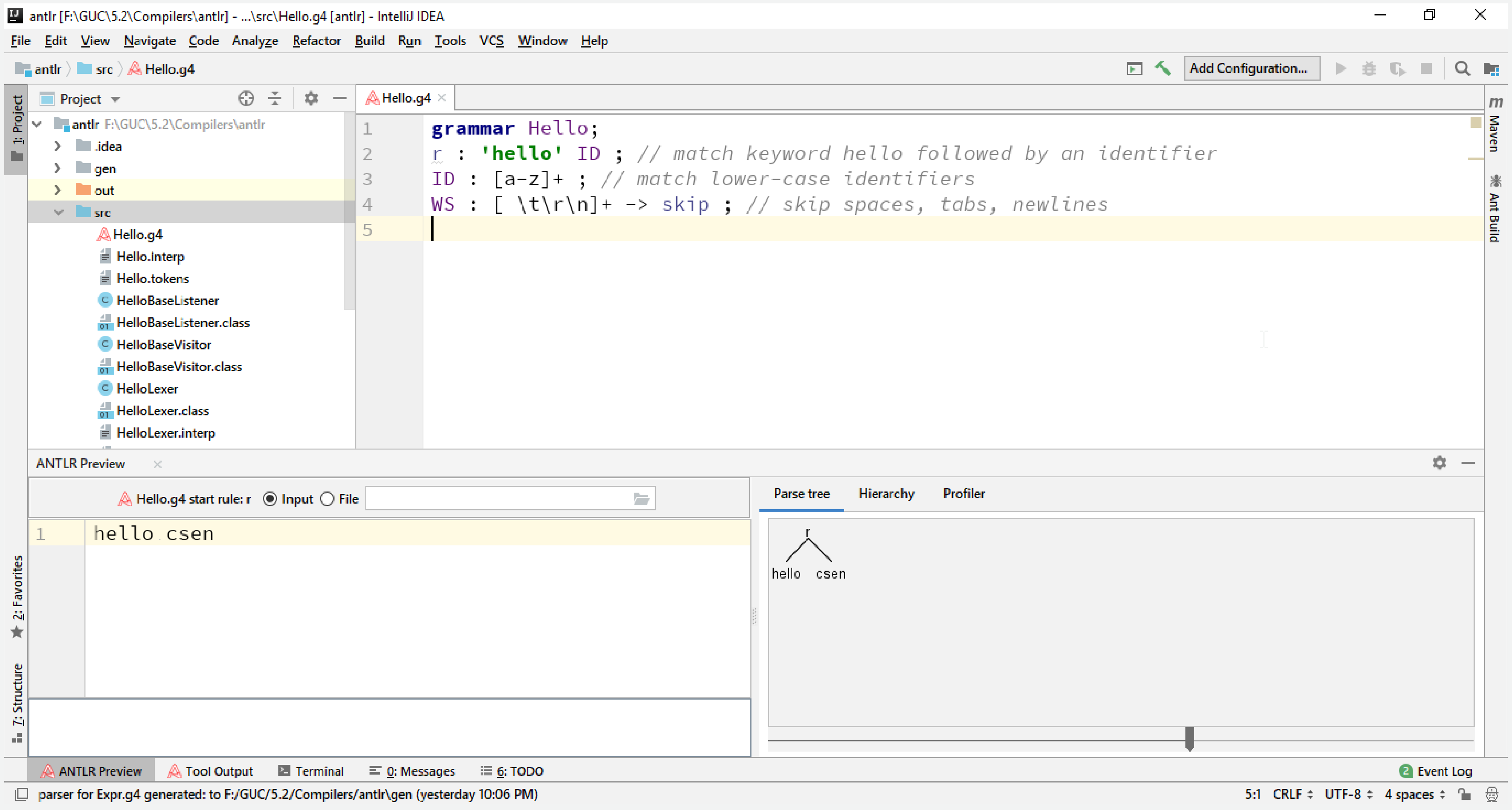
```
line 1:5 missing ID at '<EOF>'
```



```
grammar Hello;
r : 'hello' ID ; // match keyword h
ID ...case i
WS ...skip s
```

Hello.g4
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Paste from History... Ctrl+Shift+V
Paste without Formatting Ctrl+Alt+Shift+V
Column Selection Mode Alt+Shift+Insert
Find Usages Alt+F7
Refactor
Folding
Analyze
Go To
Generate... Alt+Insert
Recompile 'Hello.g4' Ctrl+Shift+F9
Show in Explorer
Open in Terminal
Pitest all tests in module
Local History
Compare with Clipboard
File Encoding
Generate ANTLR Recognizer Ctrl+Shift+G
Configure ANTLR...
Test Rule r
Create Gist...

Example 1 Test:



antlr [F:\GUC\5.2\Compilers\antlr] - ...\src\Hello.g4 [antlr] - IntelliJ IDEA

```
grammar Hello;
r : 'hello' ID ; // match keyword hello followed by an identifier
ID : [a-z]+ ; // match lower-case identifiers
WS : [ \t\r\n]+ -> skip ; // skip spaces, tabs, newlines
```

Hello.g4 start rule: r [✓ Input] File

```
hello csen
```

parser for Expr.g4 generated: to F:/GUC/5.2/Compilers/antlr\gen (yesterday 10:06 PM)

References:

- Antlr intellj plugins:
  - Download:
    - https://plugins.jetbrains.com/plugin/7358-antlr-v4-grammar-plugin/versions
  - Steps:
    - https://www.youtube.com/watch?time_continue=121&v=svEZtRjVBTY&feature=emb_logo
- Antlr tools
  - https://www.antlr.org/tools.html
- Guide:
  - https://github.com/antlr/antlr4/blob/master/doc/getting-started.md