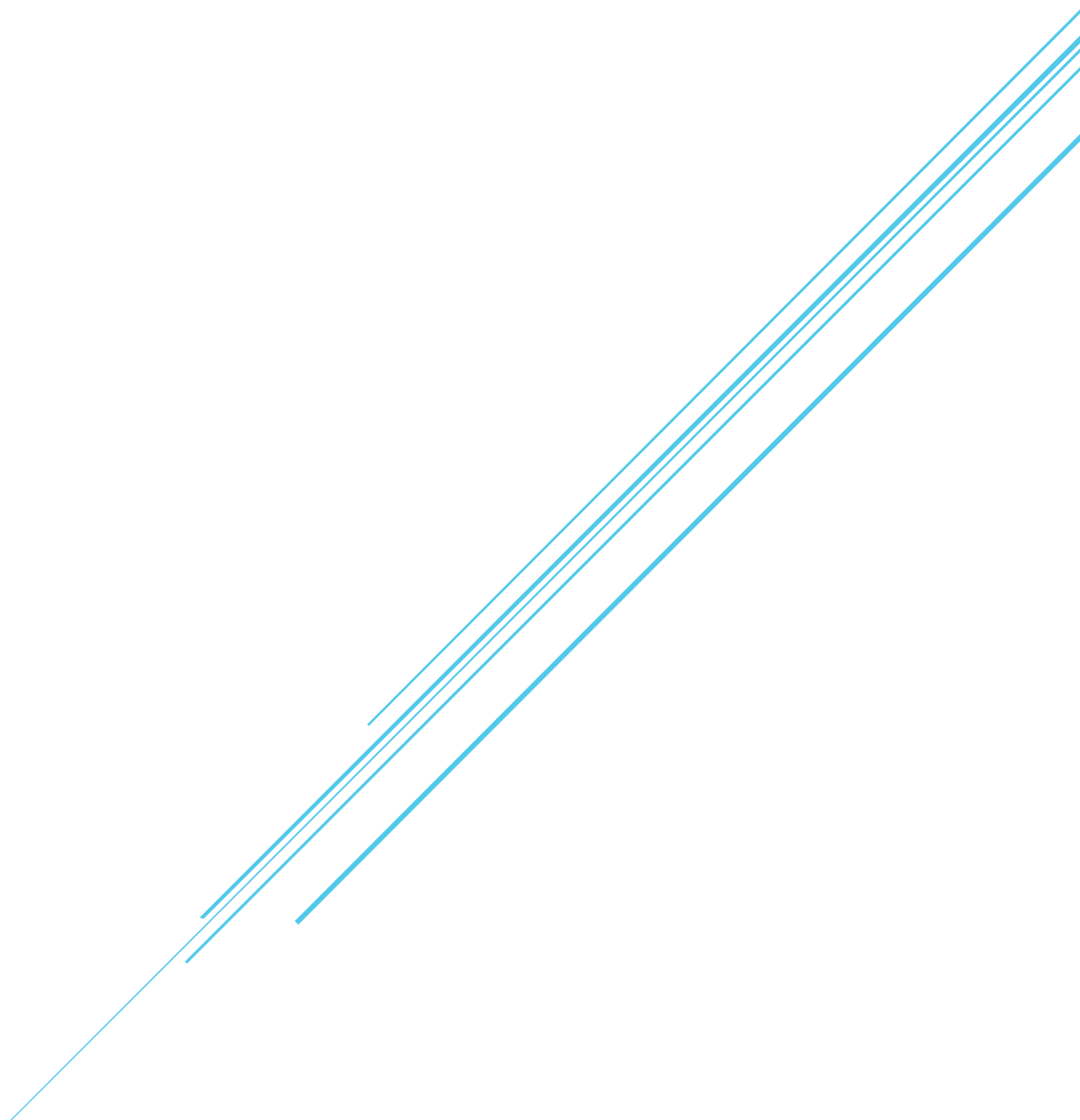


ANTLR4

طراحی کامپایلر

۱۴۰۲/۴/۶



- نام و نام خانوادگی : علی بدیعی
- نام استاد :
- گروه : ۱

خلاصه ای از ANTLR4

لینک ویدیو:

<https://drive.google.com/file/d/1VP-qQ9P3dJKZInbGjlhot5-0o8iCPhAu/view?usp=sharing>

+ برای نصب کتابخانه:

ابتدا از نصب بودن جاوا مطمئن میشویم. سپس برای نصب ANTLR دستورات زیر را به ترتیب در cmd اجرا میکنیم:

نصب ANTLR Python 3 Runtime:

```
sudo pip install antlr4-python3-runtime
```

برای اینکه نیاز به نصب جاوا نباشد و تماما با پایتون قابل اجرا باشد:

```
pip install antlr4-tools
```

اجرای دستور زیر برای ساخت فایل اجرایی:

```
antlr4
```

سپس فایل antlr-4.12.0-complete.jar را در مسیر زیر میگذاریم:

```
C:\Javalib
```

دستور زیر را برای اضافه شدن به Environment Variable وارد میکنیم:

```
SET CLASSPATH=.;C:\Javalib\antlr-4.12.0-complete.jar;%CLASSPATH%
```

ANTLR4

و برای aliases:

```
doskey antlr4=java org.antlr.v4.Tool $*
```

```
doskey grun =java org.antlr.v4.gui.TestRig $*
```

+ خروجی دستورات:

```
D:\University\6th Term\Other Projects\Hematkar\Test1>antlr4
ANTLR Parser Generator Version 4.12.0
-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
```

```
D:\University\6th Term\Other Projects\Hematkar\Test1>grun
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.
```

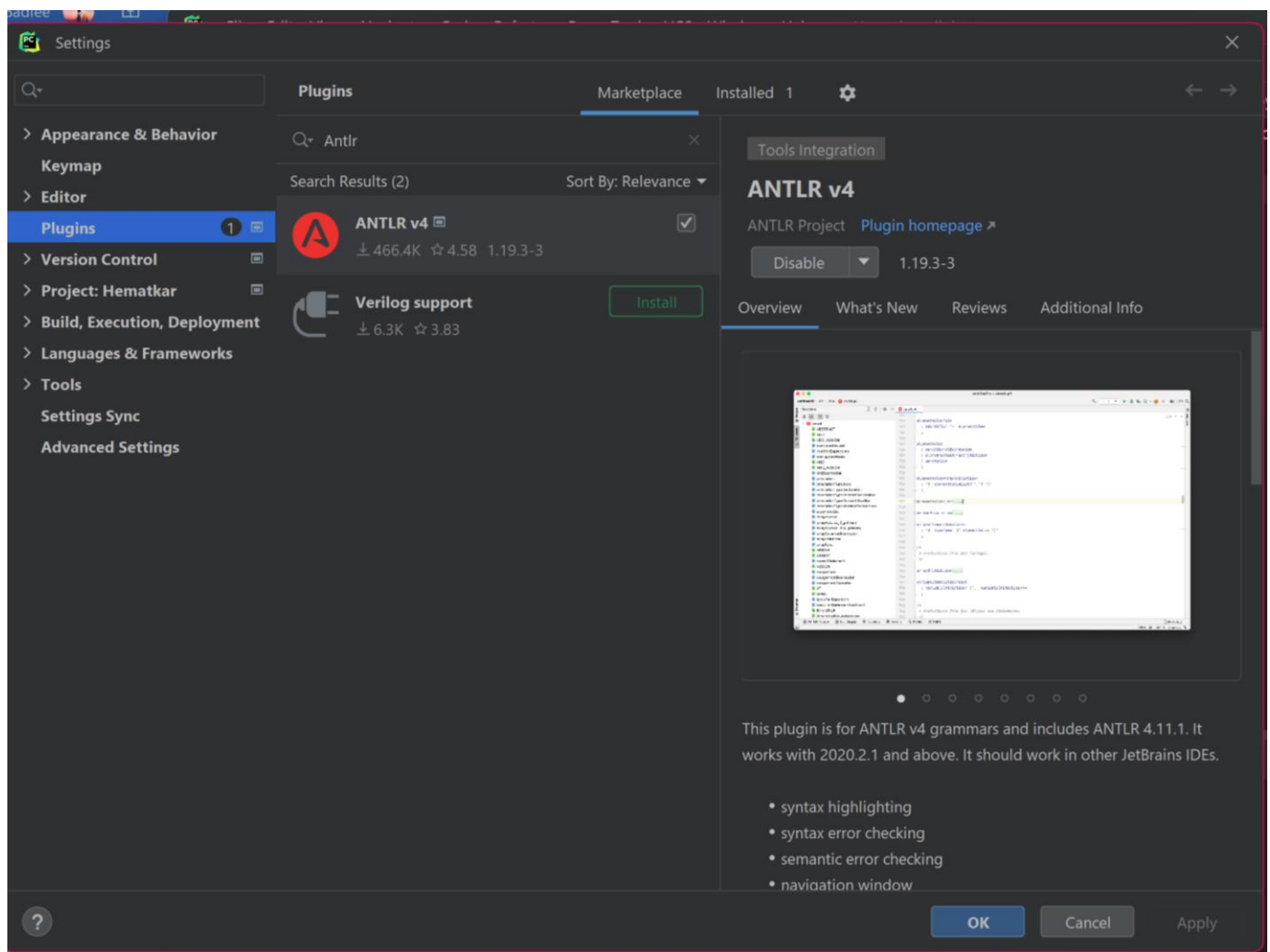
ANTLR4

+ میتوان از پلاگین برای نرم افزار های JetBrains استفاده کرد:

برای نصب ANTLR وارد قسمت زیر میشویم:

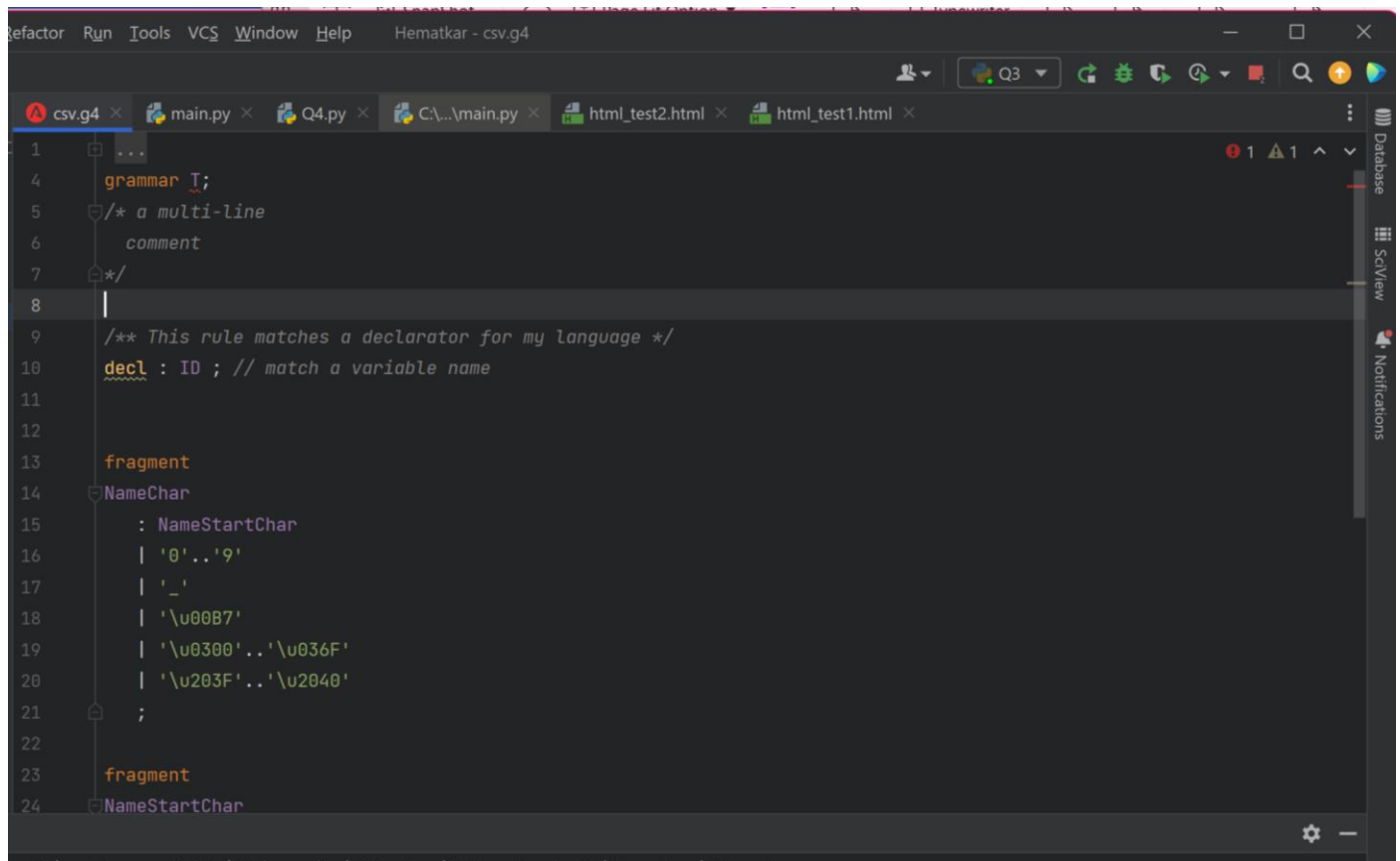
Settings > Plugins > Marketplace

سپس نام ANTLR را در قسمت سرچ وارد میکنیم و سپس روی Install کلیک میکنیم. پس از reset کردن IDE با تصویر زیر مواجه میشویم:



ANTLR4

و فایل های g4 را شناسایی میکند:



```
1  ...
4  grammar I;
5  /* a multi-line
6     comment
7  */
8
9  /** This rule matches a declarator for my language */
10 decl : ID ; // match a variable name
11
12
13 fragment
14 NameChar
15     : NameStartChar
16     | '0'..'9'
17     | '_'
18     | '\u00B7'
19     | '\u00300'..'036F'
20     | '\u003F'..'02040'
21     ;
22
23 fragment
24 NameStartChar
```

ANTLR4

+ برای مثال ، از خود ابزار ANTLR4 استفاده میکنیم که به صورت گرافیکی نشان دهد:

Hello.g4:

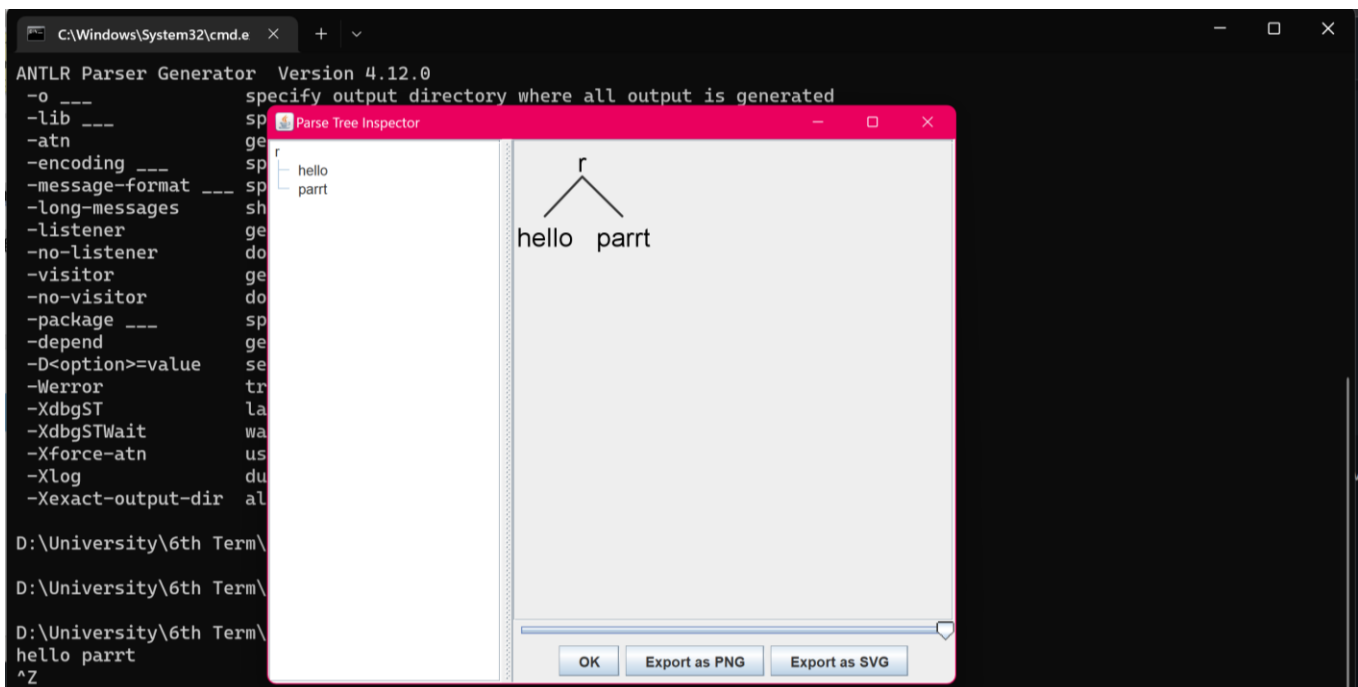
```
grammar Hello; //Define a grammar called Hello
r : 'hello' ID ;    // match keyword hello followed by an identifier
ID : [a-z]+ ;      // match lower-case identifiers
WS : [ \t\r\n]+ -> skip ; // skip spaces, tabs, newlines
```

antlr4 Hello.g4

javac Hello*.java

grun Hello r -gui
hello parrt
^D

9



برای ساخت درخت دو راهکار وجود دارد:

۱. فایل XMLParser.g4 را در قسمت Editor باز میکنیم. سپس در پایین ترین قسمت Pycharm ،

قسمت ANTLR Preview را باز میکنیم. در قسمت سمت چپ کد برنامه ها را وارد کرده یا خود

فایل را باز میکنیم. در قسمت سمت راست ، درخت آن مشاهده میشود.

۲. میتوان از دستورات زیر استفاده کرد که از خود antlr4 (به صورت پیش فرض جاوا) استفاده کرده و

درخت آن را رسم کند (قابل ذخیره سازی است).

```
antlr4 XMLLexer.g4
```

```
antlr4 XMLParser.g4
```

و سپس در دایرکتوری ای که فایل های جاوا ساخته شده است ، دستور زیر را وارد میکنیم:

```
javac *.java
```

پس از آن میتوان دستور زیر را وارد کرد و درخت را در یک پنجره مشاهده کرد و حتی آن را سیو کرد.

```
grun XML toplevel -gui P7.ui
```

+ برای ساخت فایل های **header** یا کتابخانه ای برای اضافه کردن در زبان های مختلف ، میتوان از دستور زیر استفاده کرد و روبروی **DLanguage** ، زبان مورد نظر را قرار داد.

ابتدا فایل های **PythonLexer.g4** و **PythonParser.g4** را در دایرکتوری کنار دیگر فایل ها قرار میدهیم. سپس دستور زیر را برای ساخت **token** ها وارد میکنیم:

```
antlr4 -Dlanguage=Python3 PythonLexer.g4
```

و سپس:

```
antlr4 -Dlanguage=Python3 PythonParser.g4
```

با اینکار تعدادی فایل در داخل آن دایرکتوری ساخته میشود.

فایل **PythonParser.g4** را در قسمت **Editor** باز میکنیم. سپس در پایین ترین قسمت **Pycharm** ، قسمت **ANTLR Preview** را باز میکنیم. در قسمت سمت چپ کد برنامه ها را وارد کرده یا خود فایل را باز میکنیم. در قسمت سمت راست ، درخت آن مشاهده میشود.