

컴파일러개론 2주차 실습

ANTLR

2024. 09. 20.

TA: 박정필

✉: 202350941@o.cnu.ac.kr

공지, 질문 방법

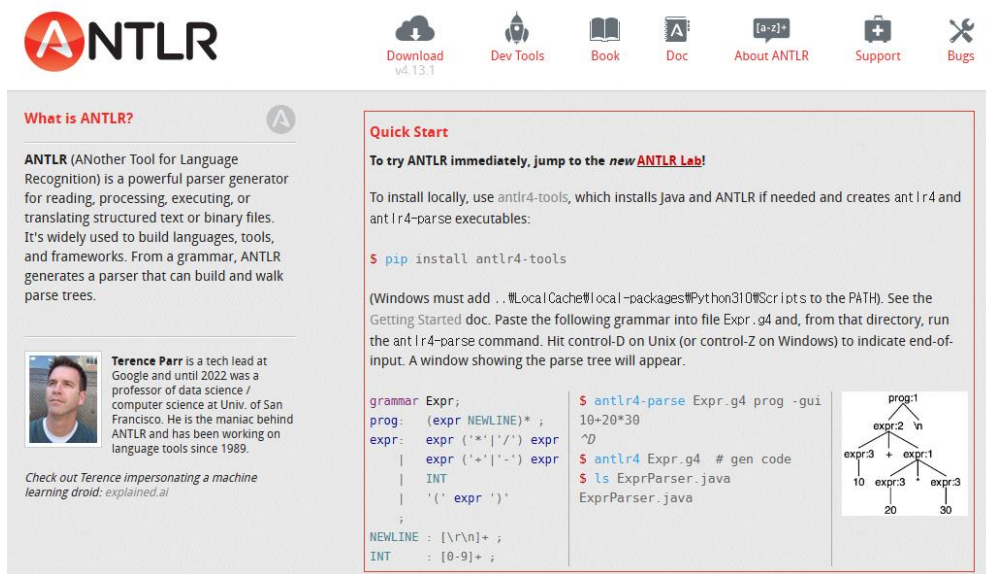
- 강의자료, 동영상
 - 사이버캠퍼스 업로드 예정
- 공지
 - 카카오톡 오픈채팅, 사이버캠퍼스
 - <https://open.kakao.com/o/gkfF7JMg>
 - 오픈프로필 사용 가능
 - 참여코드: 2024cp01
- 질문
 - 수업시간, 카카오톡 오픈채팅에 질문
 - 과제 관련 질문은 제출기한 전날까지만 가능 (당일 질문은 답변 X)
 - 이메일
 - 이메일 제목은 "[컴파일러개론][분반] ...", 제목 반드시 준수
 - 교수님 : eschough@cnu.ac.kr
 - TA : 202350941@o.cnu.ac.kr

목차

- 2주차 실습
 - ANTLR 설치 및 실행
- 2주차 과제
 - 학번과 Rule 번호 출력

ANTLR 설치 및 실행

- ANTLR? (Another Tool For Language Recognition)
 - 구문 분석 (Syntax Analysis)을 위해 LL(*) 문법을 사용하는 Parser Generator (Parser 가 아님에 주의, Parser 생성 도구)



The screenshot shows the ANTLR website homepage. At the top, there's a navigation bar with icons for Download (v4.13.1), Dev Tools, Book, Doc, About ANTLR, Support, and Bugs. The main content area is divided into two columns. The left column, titled 'What is ANTLR?', contains a brief description of ANTLR as a powerful parser generator and a bio for Terence Parr. The right column, titled 'Quick Start', provides instructions on how to try ANTLR immediately by jumping to the new ANTLR Lab, followed by a list of commands to install and run ANTLR, and a small diagram of a parse tree for the expression '10+20*30'.

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.

Quick Start

To try ANTLR immediately, jump to the [new ANTLR Lab!](#)

To install locally, use `antlr4-tools`, which installs Java and ANTLR if needed and creates `antlr4` and `antlr4-parse` executables:

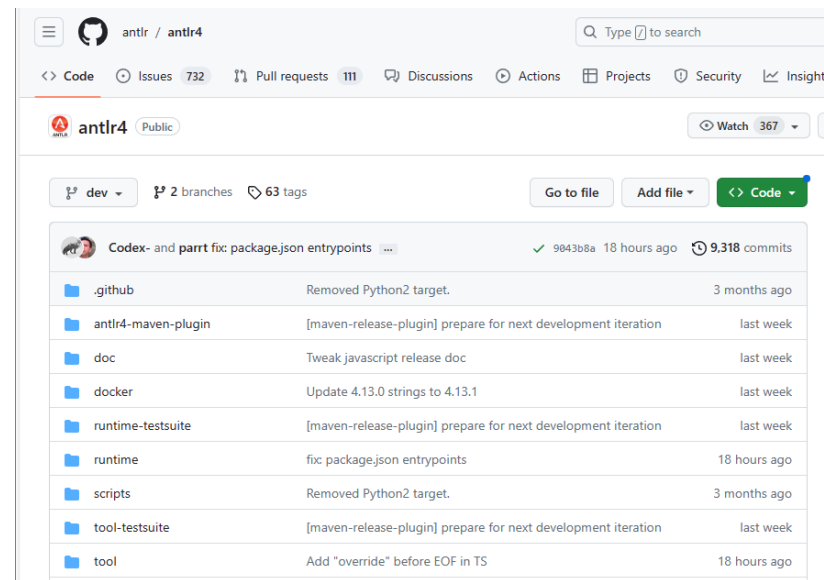
```
$ pip install antlr4-tools
```

(Windows must add `..\\LocalCache\\local-packages\\Python310\\Scripts` to the PATH). See the Getting Started doc. Paste the following grammar into file `Expr.g4` and, from that directory, run the `antlr4-parse` command. Hit control-D on Unix (or control-Z on Windows) to indicate end-of-input. A window showing the parse tree will appear.

```
grammar Expr;
prog: (expr NEWLINE)* ;
expr: expr ('*' '/' '^') expr
    | INT
    | '(' expr ')';
NEWLINE: [\r\n]+ ;
INT: [0-9]+ ;
```

```
$ antlr4-parse Expr.g4 prog -gui
10+20*30
^D
$ antlr4 Expr.g4 # gen code
$ ls ExprParser.java
ExprParser.java
```

The parse tree diagram shows the structure of the expression `10+20*30`. The root node is `prog:1`, which has a single child `expr:2`. `expr:2` has three children: `expr:3` (10), `+`, and `expr:1`. `expr:1` has three children: `expr:3` (20), `*`, and `expr:3` (30).



The screenshot shows the GitHub repository for ANTLR. The repository is named 'antlr4' and is public. It has 732 issues, 111 pull requests, and 63 tags. The repository is managed by 'antlr' and has 367 watchers. The commit history shows a recent commit by 'Codex-' titled 'and parrr fix: package.json endpoints' with 9,318 commits. The repository contains several files and folders, including .github, antlr4-maven-plugin, doc, docker, runtime-testsuite, runtime, scripts, tool-testsuite, and tool.

antlr / antlr4

Code Issues 732 Pull requests 111 Discussions Actions Projects Security Insights

antlr4 Public Watch 367

dev 2 branches 63 tags Go to file Add file Code

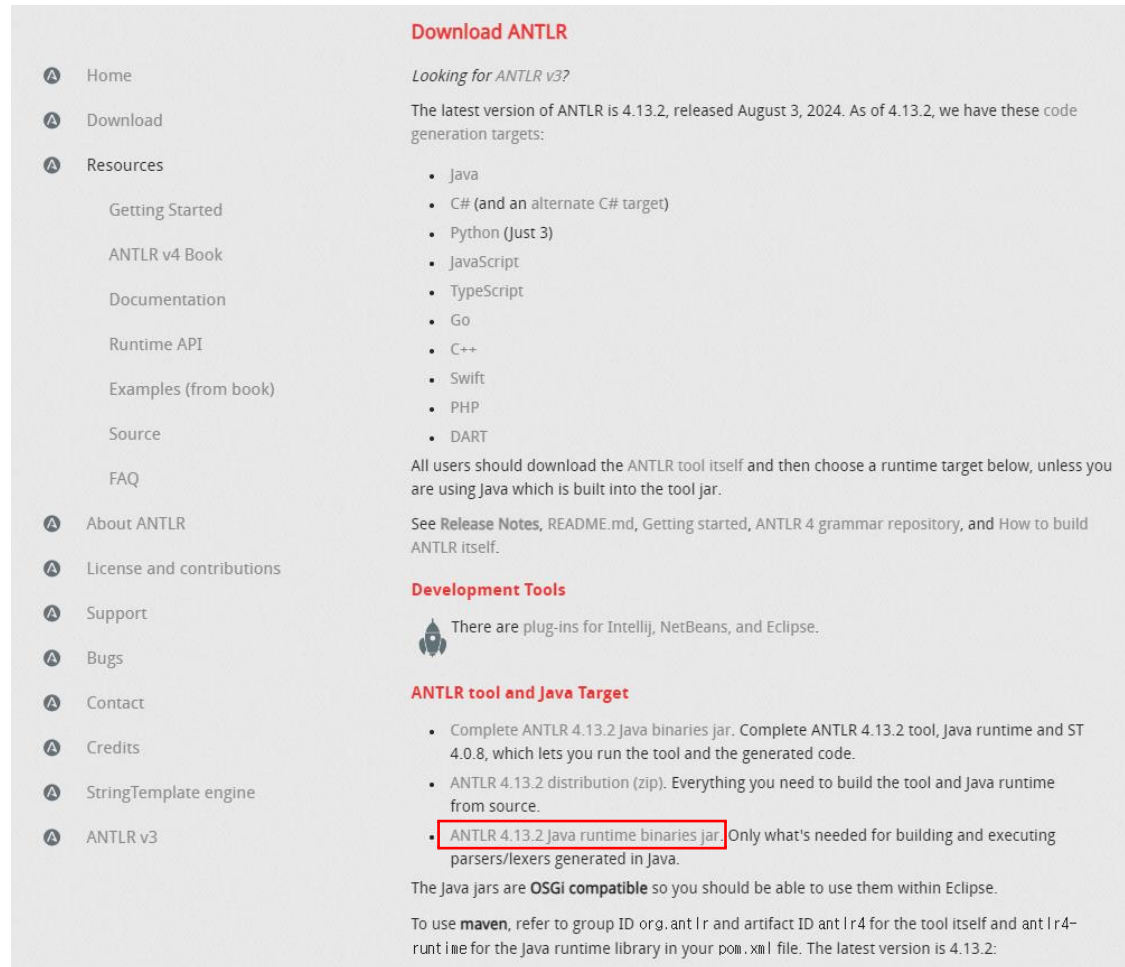
Codex- and parrr fix: package.json endpoints 9043b8a 18 hours ago 9,318 commits

File/Folder	Description	Last Update
.github	Removed Python2 target.	3 months ago
antlr4-maven-plugin	[maven-release-plugin] prepare for next development iteration	last week
doc	Tweak javascript release doc	last week
docker	Update 4.13.0 strings to 4.13.1	last week
runtime-testsuite	[maven-release-plugin] prepare for next development iteration	last week
runtime	fix package.json endpoints	18 hours ago
scripts	Removed Python2 target.	3 months ago
tool-testsuite	[maven-release-plugin] prepare for next development iteration	last week
tool	Add "override" before EOF in TS	18 hours ago

ANTLR Jar 다운로드

■ ANTLR Jar 다운로드

- <https://www.antlr.org/download.html>
- ANTLR 4.13.2 java runtime binaries jar 다운로드



The screenshot shows the ANTLR website's download page. On the left is a navigation menu with links: Home, Download, Resources (Getting Started, ANTLR v4 Book, Documentation, Runtime API, Examples (from book), Source, FAQ), About ANTLR, License and contributions, Support, Bugs, Contact, Credits, StringTemplate engine, and ANTLR v3. The main content area is titled 'Download ANTLR' and includes a section 'Looking for ANTLR v3?' stating the latest version is 4.13.2. It lists code generation targets: Java, C# (and an alternate C# target), Python (Just 3), JavaScript, TypeScript, Go, C++, Swift, PHP, and DART. A note states that users should download the ANTLR tool itself and then choose a runtime target, unless using Java which is built into the tool jar. It references Release Notes, README.md, Getting started, ANTLR 4 grammar repository, and How to build ANTLR itself. A 'Development Tools' section mentions plug-ins for IntelliJ, NetBeans, and Eclipse. The 'ANTLR tool and Java Target' section lists three options: Complete ANTLR 4.13.2 Java binaries jar, ANTLR 4.13.2 distribution (zip), and ANTLR 4.13.2 Java runtime binaries jar (highlighted with a red box). The highlighted option is described as 'Only what's needed for building and executing parsers/lexers generated in Java.' A note mentions that the Java jars are OSGI compatible. At the bottom, it explains how to use Maven to refer to the group ID, artifact ID, and version for the tool and runtime libraries.

Download ANTLR

Looking for ANTLR v3?

The latest version of ANTLR is 4.13.2, released August 3, 2024. As of 4.13.2, we have these code generation targets:

- Java
- C# (and an alternate C# target)
- Python (Just 3)
- JavaScript
- TypeScript
- Go
- C++
- Swift
- PHP
- DART

All users should download the ANTLR tool itself and then choose a runtime target below, unless you are using Java which is built into the tool jar.

See **Release Notes**, **README.md**, **Getting started**, **ANTLR 4 grammar repository**, and **How to build ANTLR itself**.

Development Tools

There are plug-ins for IntelliJ, NetBeans, and Eclipse.

ANTLR tool and Java Target

- Complete ANTLR 4.13.2 Java binaries jar. Complete ANTLR 4.13.2 tool, Java runtime and ST 4.0.8, which lets you run the tool and the generated code.
- ANTLR 4.13.2 distribution (zip). Everything you need to build the tool and Java runtime from source.
- **ANTLR 4.13.2 Java runtime binaries jar** Only what's needed for building and executing parsers/lexers generated in Java.

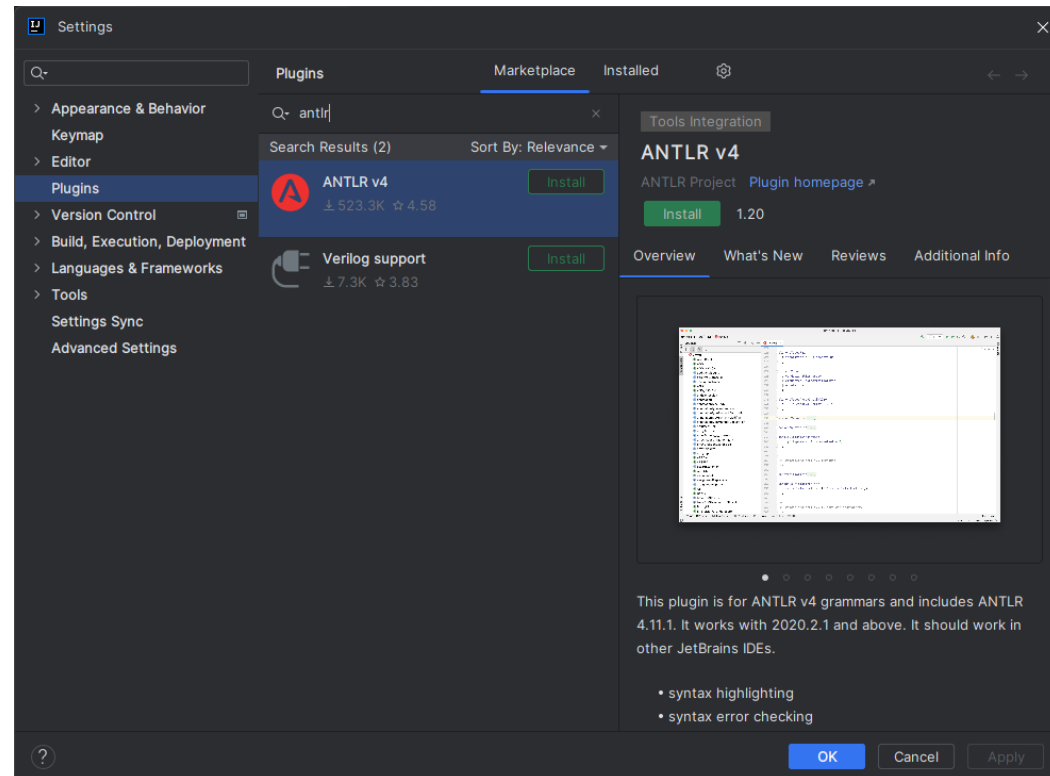
The Java jars are **OSGI compatible** so you should be able to use them within Eclipse.

To use **maven**, refer to group ID `org.antlr.v4` and artifact ID `antlr4-runtime` for the tool itself and `antlr4-runtime` for the Java runtime library in your `pom.xml` file. The latest version is 4.13.2:

ANTLR 설치 및 실행 (1/5)

■ IntelliJ IDEA 에 ANTLR 설치 및 실행

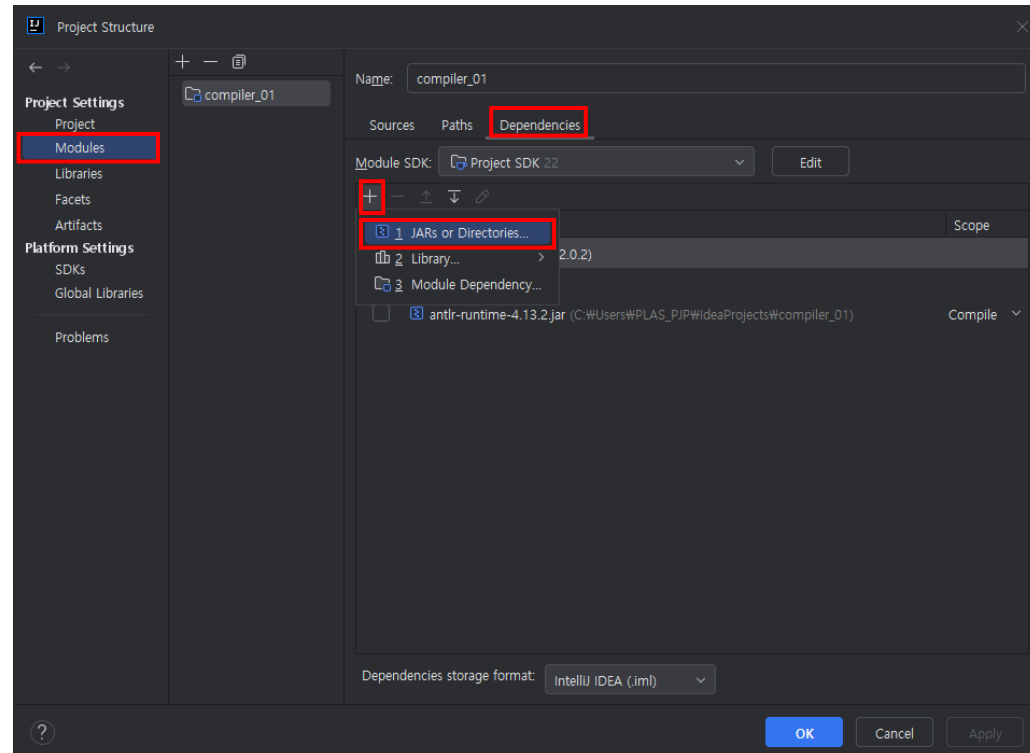
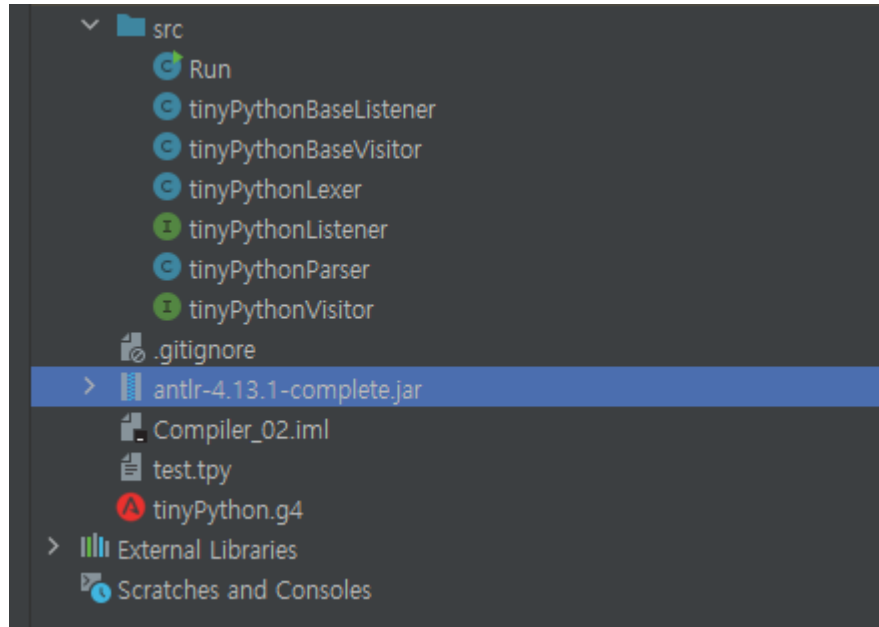
- File -> Settings...
- Plugins
- ANTLR 검색
- Install



ANTLR 설치 및 실행 (2/5)

■ ANTLR Jar 파일 추가

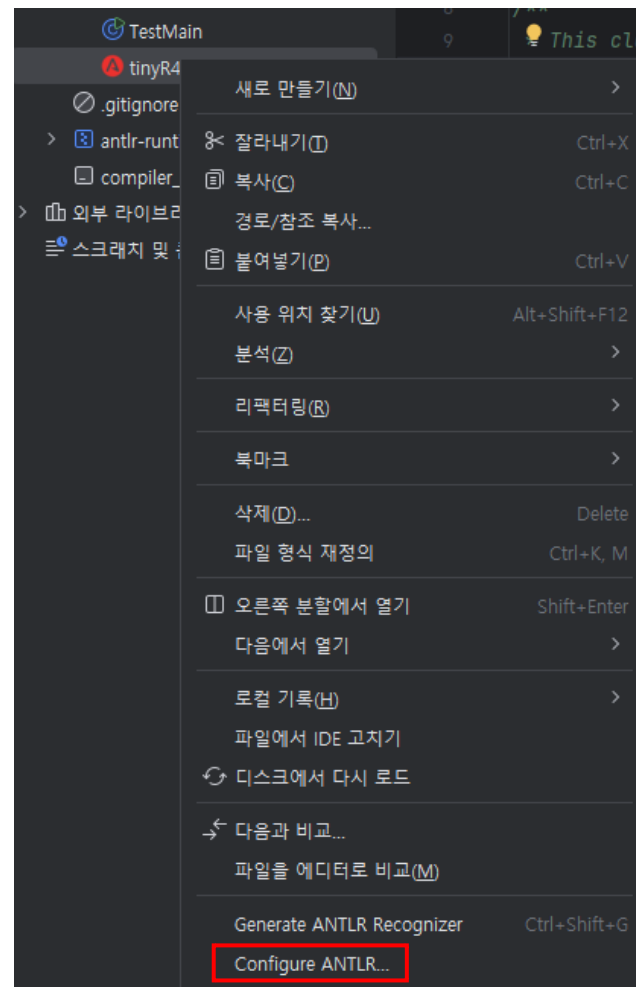
- 다운 받은 jar파일 프로젝트 폴더에 추가
- File > Project Structure > Modules > Dependencies 에서 ANTLR Jar 추가



ANTLR 설치 및 실행 (3/5)

- tinyR4.g4

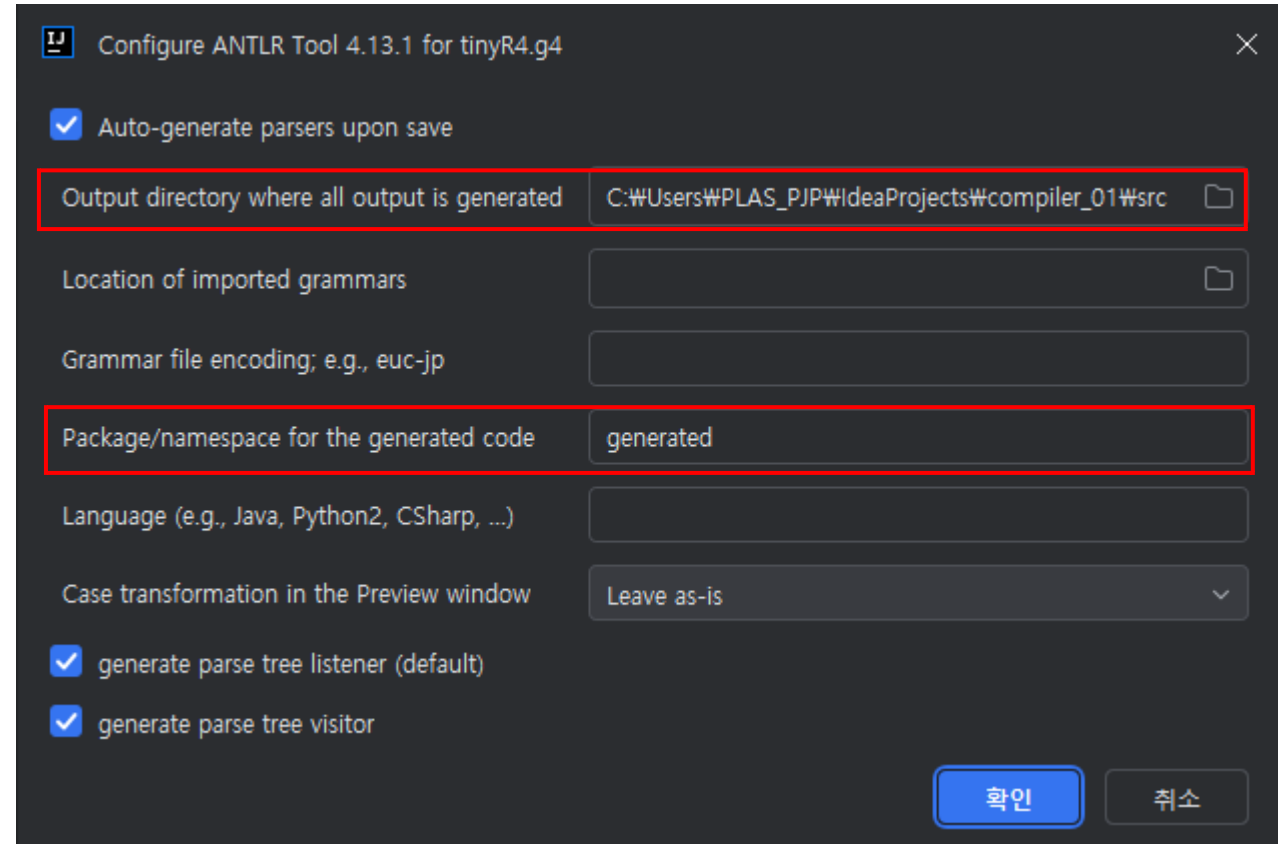
- 사이버캠퍼스에서
파일 다운로드
- 프로젝트에 tinyR4.g4 추가
- tinyR4.g4 우클릭 후,
Configure ANTLR... 클릭



ANTLR 설치 및 실행 (4/5)

■ Configure ANTLR

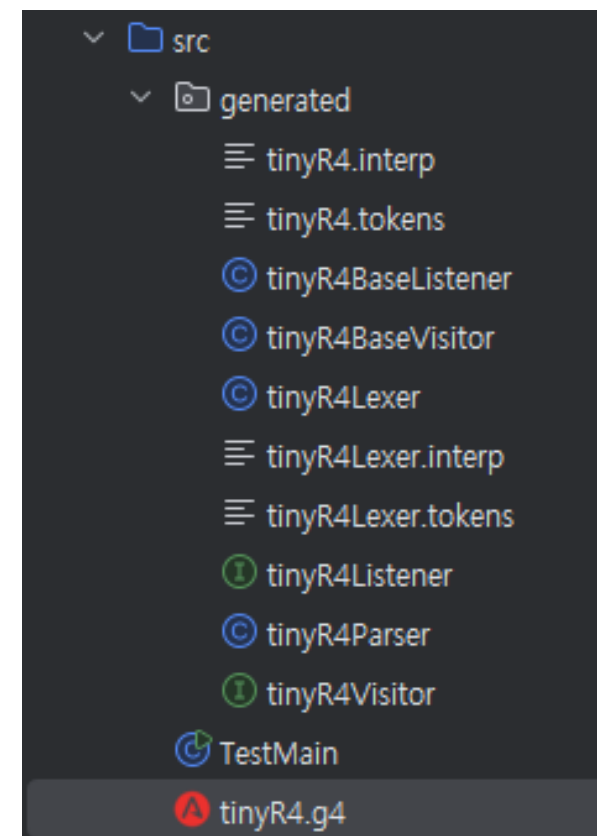
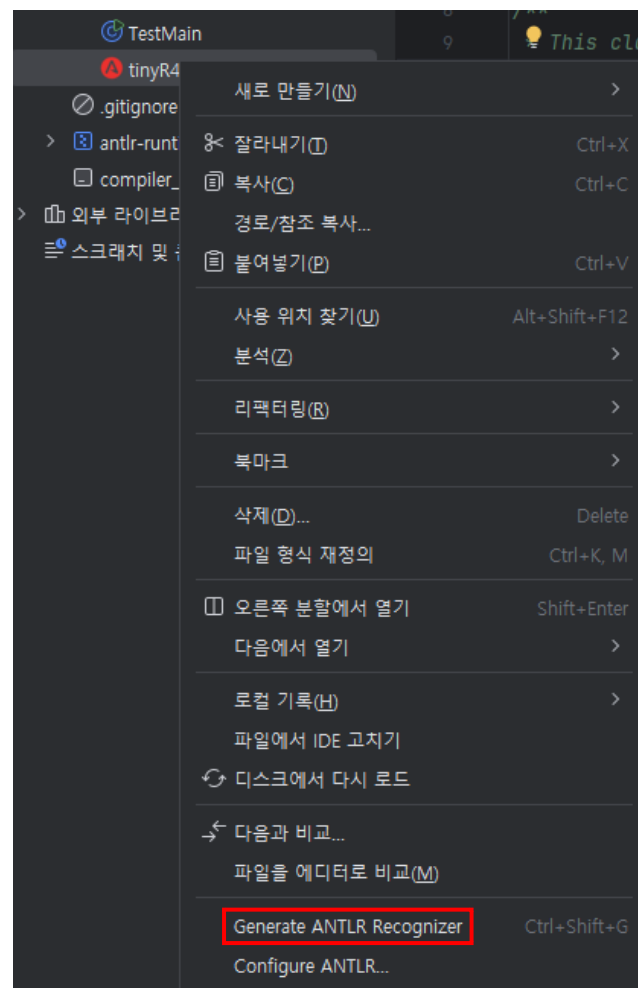
- Lexer, Parser 등이 생성될 경로 및 기타 옵션을 설정할 수 있음
- Output directory ...는 자신의 src 경로로 설정
- Package/namespace는 그림과 동일하게 generated 입력



ANTLR 설치 및 실행 (5/5)

■ Generate ANTLR Recognizer

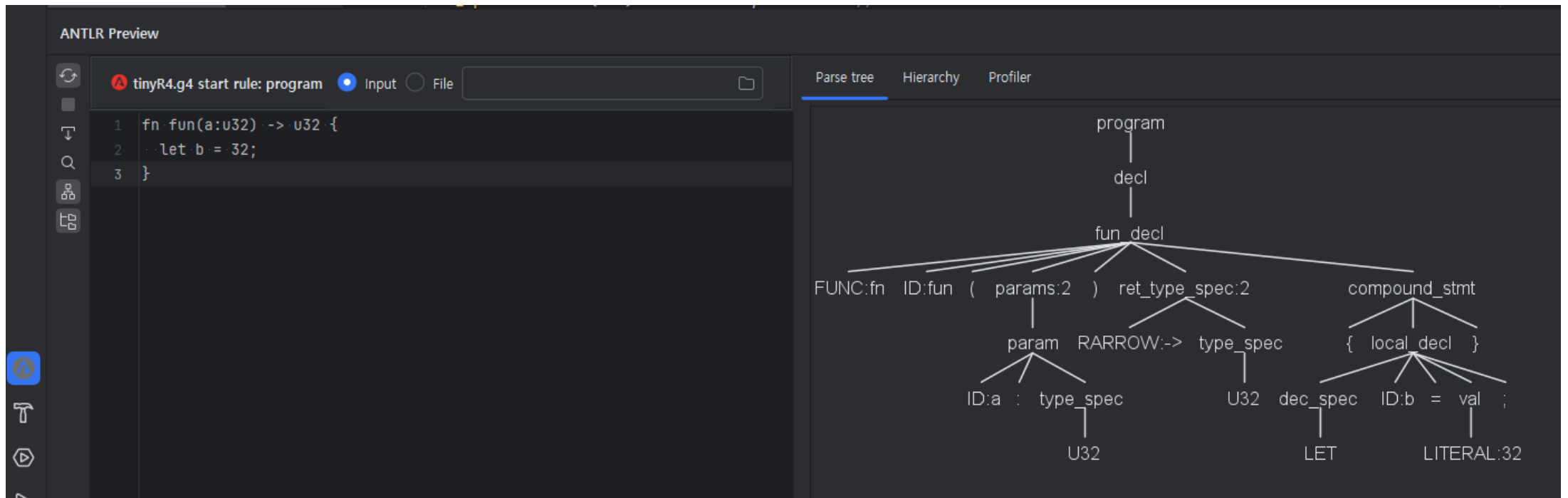
- 그 후, 다시 tinyR4.g4 우클릭하여 Generate ANTLR Recognizer 수행
- 그러면 오른쪽 그림과 같이 generated 패키지에 Lexer, Parser 등이 생성됨



ANTLR Preview (1/2)

■ ANTLR Preview

- tinyR4.g4 코드에서 program을 우클릭하여 Test Rule program 클릭
- 아래와 같이 입력한 코드에 대한 Parse tree를 보여줌



ANTLR Preview (2/2)

■ ANTLR Preview

- 입력한 예시에 틀린 문법이 있으면 어디가 틀렸는지 보여줌

The screenshot displays the ANTLR Preview interface. On the left, a code editor shows the following code:

```
1 fn fun(a:u32) -> u32 {  
2   let b = 32;  
3   let k;  
4 }
```

On the right, the 'Parse tree' tab is active, showing a hierarchical tree structure. The root node is 'program', which branches into 'decl', then 'fun_decl'. 'fun_decl' branches into 'FUNC:fn', 'ID:fun', '(', 'params:2', ')', 'ret_type_spec:2', and 'compound_stmt'. 'params:2' branches into 'param', which further branches into 'ID:a' and 'type_spec' (which branches into 'U32'). 'ret_type_spec:2' branches into 'R_ARROW:->' and 'type_spec' (which branches into 'U32'). 'compound_stmt' branches into '{', 'local_decl', '=', 'val', ';', 'local_decl', and '}'. The first 'local_decl' branches into 'dec_spec' (which branches into 'LET') and 'ID:b'. The second 'local_decl' branches into 'dec_spec' (which branches into 'LET') and 'ID:k'. A red highlight is visible under 'ID:k'. At the bottom of the code editor, a message states: 'line 3:7 mismatched input ';' expecting {';', '='}'.

참고 – ANTLR 설치 동영상

- 설치 동영상 링크: <https://youtu.be/34SfLpsyCDo?si=OQud-U38ED2S8TFV>
- ANTLR와 관련된 동영상(영어): <https://www.youtube.com/watch?v=itajbtWKPGQ>

과제 소개

■ 과제 소개 및 일정

- 과제: tinyR4.g4에 자신의 학번과 Rule 번호를 추가하여 출력하기
- 간단한 테스트 케이스 5개 제공
- 마감 기한: 09.20.(금) ~ 09.27.(금) 23:59
- 제출방법: 자신이 직접 작성한 테스트 케이스와 보고서, 총 2개를 압축하여 제출
- 테스트케이스, 보고서, 그리고 압축 파일의 파일명은 자신의 학번으로 바꿔서 제출
- e.g. 2022000000.zip 파일 목록
 - 2022000000.tr
 - 2022000000.pdf

과제 설명

■ 학번과 Rule 번호 추가

- tinyR4.g4 파일의 초기 상태는 아래와 같이 program 문법에 {System.out.println("202200000 Rule 0");} 가 추가되어있음
- 아래 문법들도 {System.out.println("202200000 Rule 0");} 를 추가하면서 Rule 번호를 증가 및 정의가 2개인 문법은 -1, -2를 붙이기
- **Terminal은 제외.** NonTerminal만 학번과 Rule 번호를 추가하면 됨

```
tinyR4.g4 x
1 grammar tinyR4;
2
3 program      : decl+      {System.out.println("202200000 Rule 0");} ;
4
5 decl         : fun_decl {System.out.println("202200000 Rule 1");};
6
7 fun_decl     : FUNC ID '(' params ')' ret_type_spec compound_stmt {System.out.println("202200000 Rule 2");} ;
8
9 params       : {System.out.println("202200000 Rule 3-1");}
10             | param '(' param)* {System.out.println("202200000 Rule 3-2");}
11             ;
```

NonTerminal

```
74 FUNC: 'fn';
75 U32: 'u32';
76
77 IF: 'if';
78 ELSE: 'else';
79 RETURN: 'return';
80 LET: 'let';
81 MUT: 'mut';
```

Terminal

과제 설명

■ TestMain.java

- 학번과 Rule 번호가 잘 출력되는지 확인하기 위한 TestMain.java
- 자신의 테스트 케이스를 경로에 추가하면 실행하면 됨
- 테스트 케이스의 확장자는 **.tr**

```
TestMain.java x
1  import org.antlr.v4.runtime.*;
2  import org.antlr.v4.runtime.tree.*;
3  import generated.tinyR4Lexer;
4  import generated.tinyR4Parser;
5  ▶ public class TestMain {
6  ▶      public static void main(String[] args) throws Exception {
7          tinyR4Lexer lexer = new tinyR4Lexer(CharStreams.fromFileName("./src/rust_main.tr"));
8          CommonTokenStream tokens = new CommonTokenStream(lexer);
9          tinyR4Parser parser = new tinyR4Parser(tokens);
10         ParseTree tree = parser.program();
11     }
12 }
```


과제 설명

■ 출력 예시

```
1 fn main() {  
2     println("hello?");  
3 }
```



```
202200000 Rule 3-1  
202200000 Rule 6-1  
202200000 Rule 17-1  
202200000 Rule 16-2  
202200000 Rule 15-2  
202200000 Rule 14-2  
202200000 Rule 13-1  
202200000 Rule 22-2  
202200000 Rule 17-3  
202200000 Rule 16-2  
202200000 Rule 15-2  
202200000 Rule 14-2  
202200000 Rule 13-1  
202200000 Rule 12  
202200000 Rule 11-1  
202200000 Rule 7  
202200000 Rule 2  
202200000 Rule 1  
202200000 Rule 0
```

종료 코드 0(으)로 완료된 프로세스

보고서 내용

- 아래의 내용이 보고서에 꼭 들어가야 되는 내용
 - 5개의 테스트 케이스의 출력 결과 사진
 - 5개의 테스트 케이스에서 **출력되지 않은 Rule 번호들**
 - 출력되지 않은 Rule 번호들이 들어있는 **직접 작성한** 테스트 케이스와 출력 결과 사진
(직접 작성하는 테스트 케이스는 최소 1개)
- 출력 결과 사진은 **slide 17에 있는 출력 예시와 비슷하게 삽입**해주시면 됩니다.

주의 사항

- Rule 번호 출력할 때, 자신의 학번으로 바꾸는 것 잊지 마세요.
- 출력 결과 화면에서 출력되지 않은 Rule 번호 개수에 따라 감점이 들어갑니다.
- 테스트 케이스는 main, 할당, 사칙연산, 불린연산, if문입니다.
이 외의 문법에 대한 테스트 케이스는 고려하지 않습니다.
- 보고서 작성을 위해 제공된 5개의 테스트 케이스뿐만 아니라,
출력되지 않은 Rule 번호를 찾고,
그 Rule 번호가 출력되게끔 직접 테스트 케이스를 작성하셔야 됩니다.
- tinyR4.g4 파일 수정 후, 저장하고 Generate ANTLR Recognizer를 하셔야 수정된 코드가 반영이 됩니다.