ANTLR

Nardeen M.

What is ANTLR?

- ANTLR (ANother Tool for Language Recognition) is a powerful parser generator.
- ANTLR is a tool that translates your grammar to a parser/lexer in Java (or another target language) and the runtime needed by the generated parsers/lexers.
- From a grammar, ANTLR generates a parser that can build parse trees.

Identifiers

- Lexer rules names always start with a capital letter [UpperCase].
- Parser rules names always start with a lowercase letter. The initial character can be followed by uppercase and lowercase letters, digits, and underscores.
- Here are some sample names:
  - ID, ZERO   //lexer rules
  - expr, start_rule   //parser rules

Literals

- ANTLR does not distinguish between character and string literals as most languages do. All literal strings one or more characters in length are enclosed in single quotes, such as:
  - ‘0’
  - ‘Hello’

Actions

- Actions are blocks of text written in the target language [Java] 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' ;
```

ANTLR

- For more information:
- https://github.com/antlr/antlr4/blob/master/doc/index.md

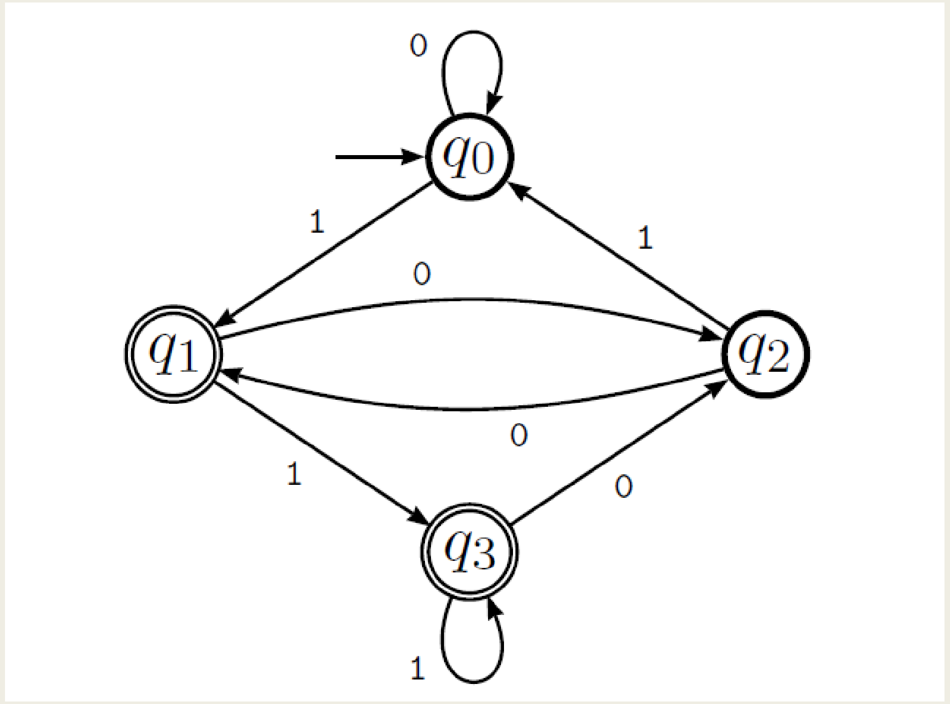
Task 5: ANTLR Lexical Analysis

Nardeen M.

Task 5: ANTLR Lexical Analysis

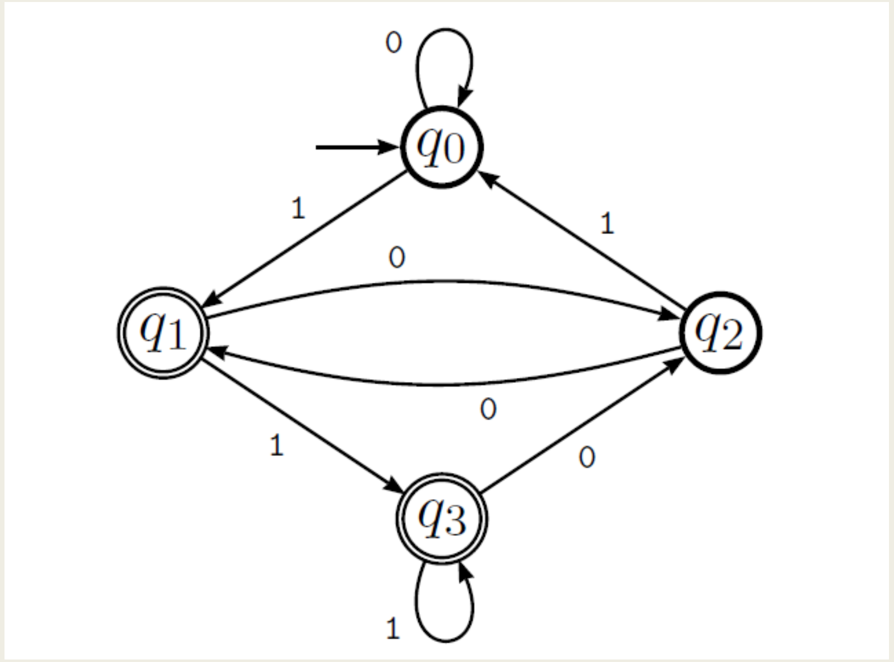
- You will implement an ANTLR lexical analyzer for the following fallback DFA:

0,0,1,;1,2,3,**00;**2,1,0,;3,2,3,**11**#1,3



Task 5: ANTLR Lexical Analysis

- Your task is to get the regular expression for this FDFA
- Then write its grammar using ANTLR
- For example, running the lexical analyzer implementing the FDFA on the string:
  - `100101` produces the output `1100`



Setup

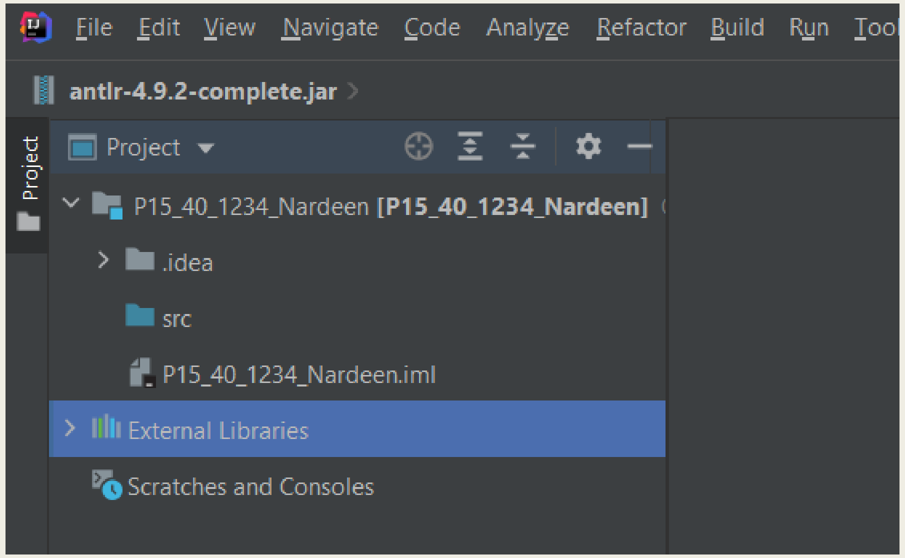
Setup

We should first:

1. Make sure you have JAVA installed and running
2. Download IntelliJ IDEA: https://www.jetbrains.com/idea/download/
    a. Please install and activate it before the session.
    b. Either Community and Ultimate will work.
3. Download ANTLR v4:
    https://www.antlr.org/download/antlr-4.9.2-complete.jar
4. Download ANTLR v4 plugin:
    https://plugins.jetbrains.com/files/7358/108410/antlr-intellij-plugin-v4-1.16.zip

Setup

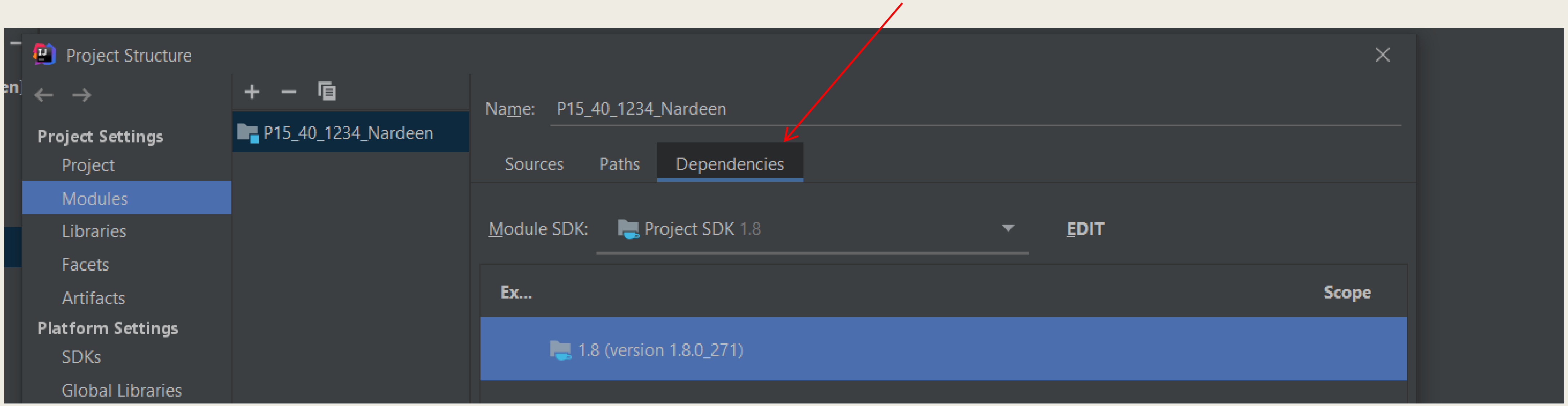
- Open intellj
- Create a new project:
  - **File** > **New** > **Project**
  - Filename: [LabNo_ID_Name], ex: P15_40_1234_Nardeen



Setup

You should add the ANTLR plugin in intellj:

- **File** > **Project Structure** > **Modules** > **Dependencies**


Project Structure

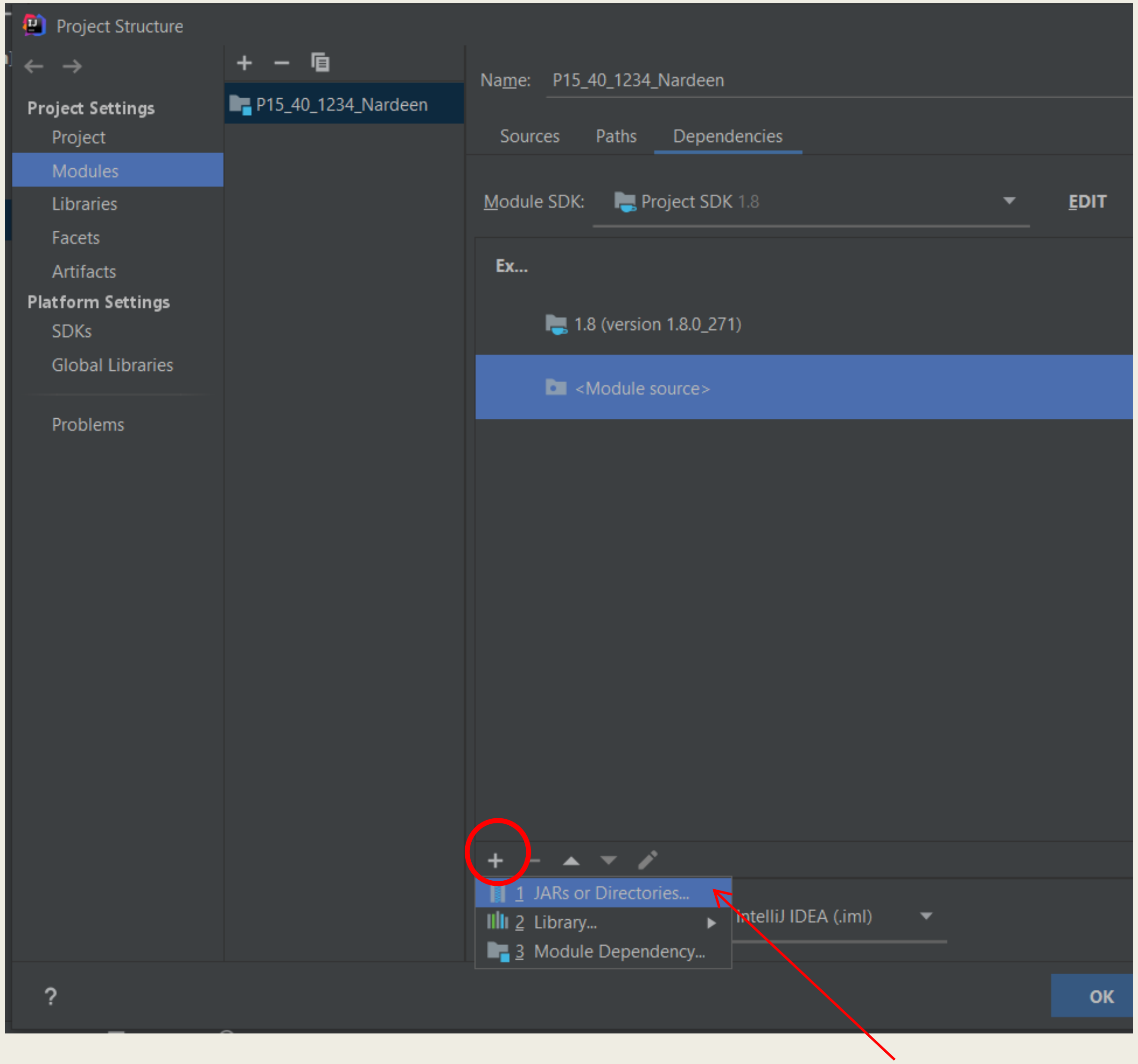
Name: P15_40_1234_Nardeen

Module SDK: Project SDK 1.8

Ex...	Scope
1.8 (version 1.8.0_271)	

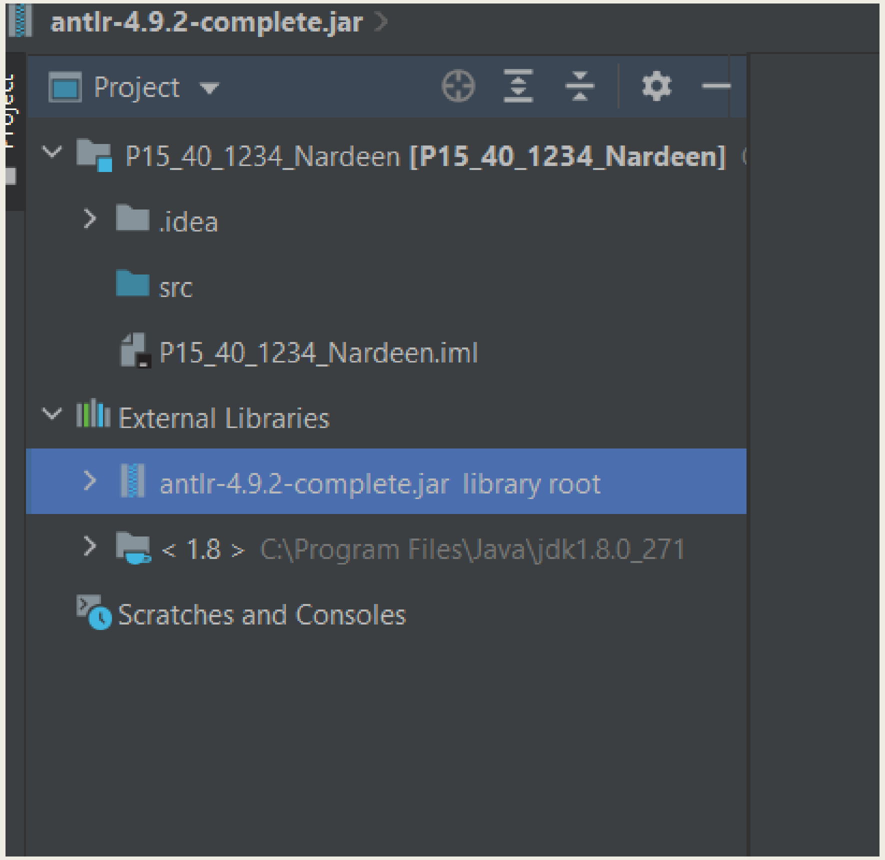
Setup

- Then Add ‘**+**’ > **1 JARs or Directories**
- Choose ANTLR (.jar) > **Ok** > **OK**



Setup

- You will find the plugin appears in the **External Libraries**



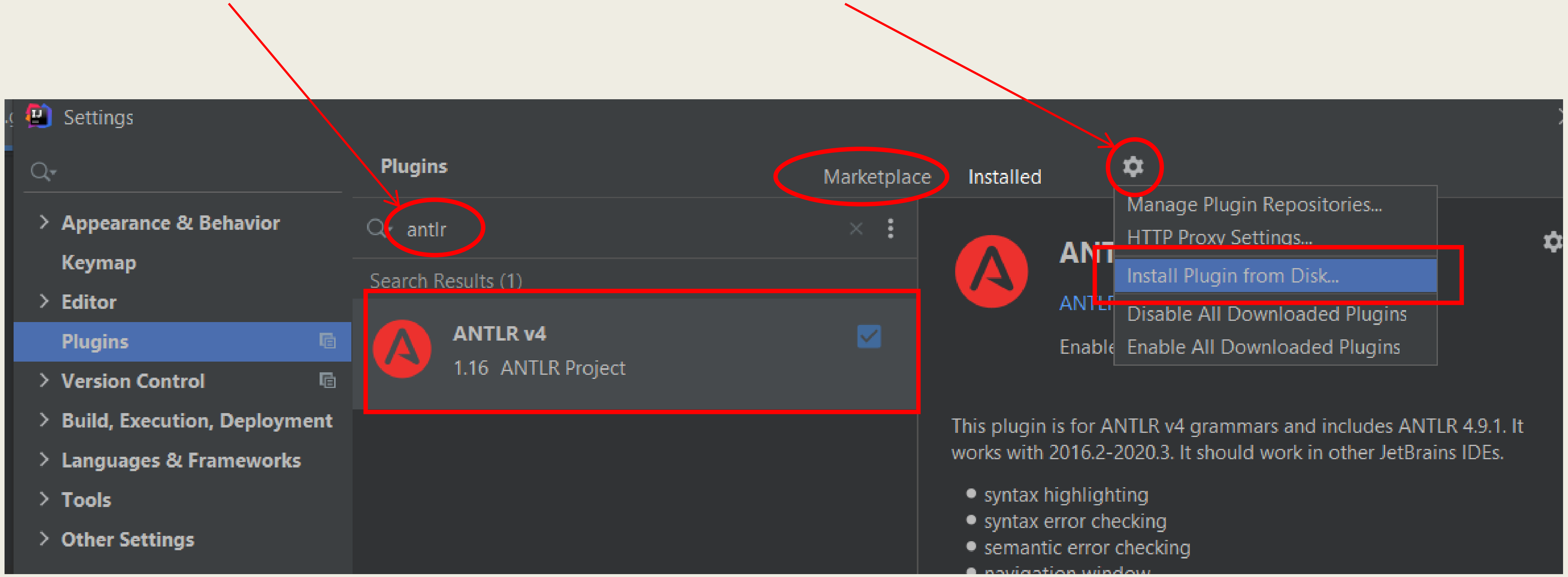
antlr-4.9.2-complete.jar

Project

P15_40_1234_Nardeen [P15_40_1234_Nardeen]
.idea
src
P15_40_1234_Nardeen.iml
External Libraries
antlr-4.9.2-complete.jar library root
< 1.8 > C:\Program Files\Java\jdk1.8.0_271
Scratches and Consoles

Setup

- ➢ File > Settings > Plugins > Add the plugin either from:
  - ➢ Markerplace > Write ANTLR OR Install Plugin from disk (.zip)

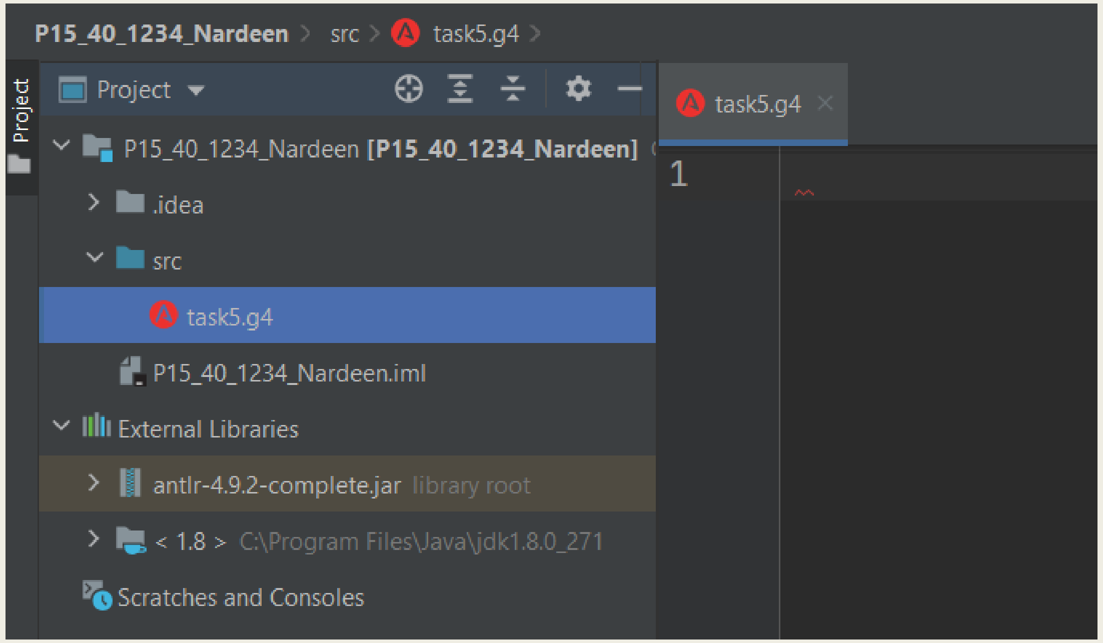


This plugin is for ANTLR v4 grammars and includes ANTLR 4.9.1. It works with 2016.2-2020.3. It should work in other JetBrains IDEs.

- syntax highlighting
- syntax error checking
- semantic error checking

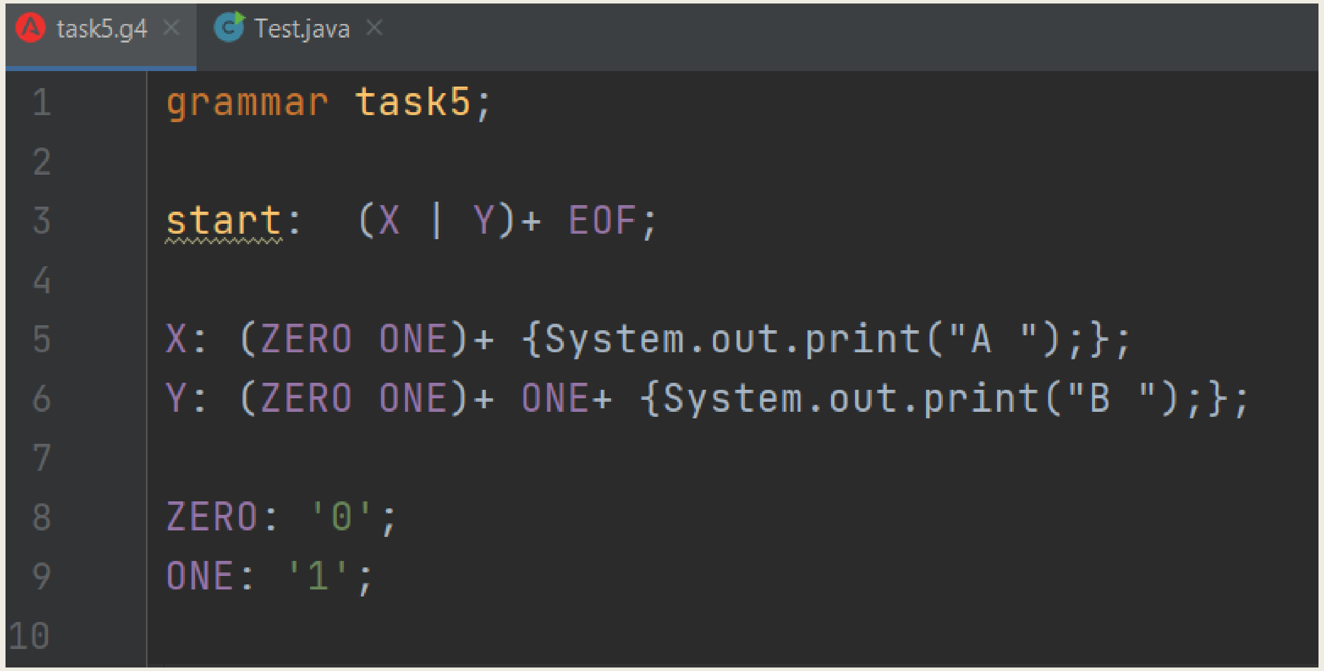
Setup

- ➢ From **scr** > Right Click > **New** > **File** > **task5.g4**


P15_40_1234_Nardeen > src > task5.g4
Project
task5.g4
P15_40_1234_Nardeen [P15_40_1234_Nardeen]
.idea
src
task5.g4
P15_40_1234_Nardeen.iml
External Libraries
antlr-4.9.2-complete.jar library root
< 1.8 > C:\Program Files\Java\jdk1.8.0_271
Scratches and Consoles

Setup

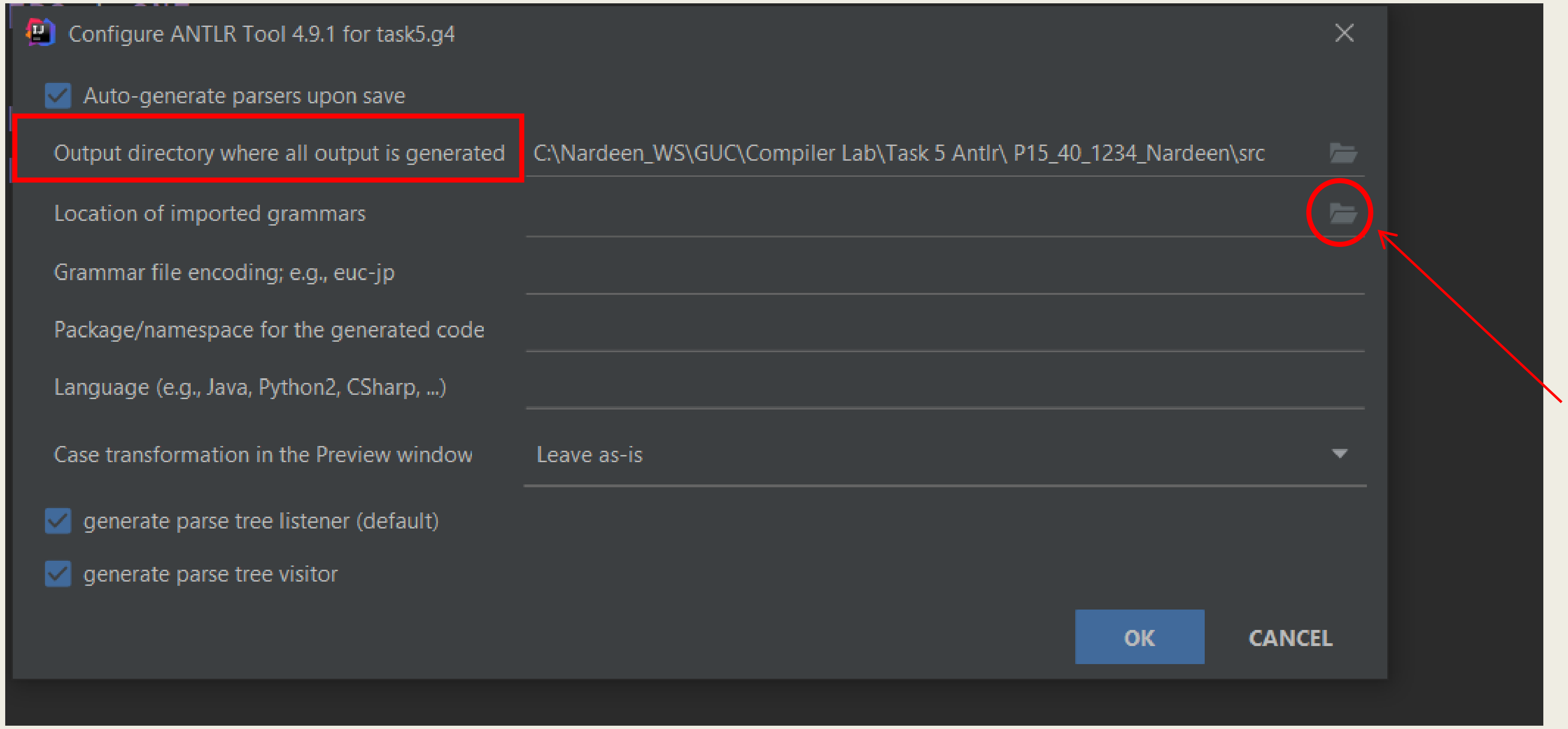
- ➢ Start write your Grammar
  - ➢ *The filename is the same name of the grammar*



```
grammar task5;

start:  (X | Y)+ EOF;

X: (ZERO ONE)+ {System.out.print("A ");};
Y: (ZERO ONE)+ ONE+ {System.out.print("B ");};

ZERO: '0';
ONE: '1';

```

Setup

- Right click in the g4 file **> Configure ANTLR** > **Output directory** > Change to the src directory



Configure ANTLR Tool 4.9.1 for task5.g4

✓ Auto-generate parsers upon save

Output directory where all output is generated: C:\Nardeen_WS\GUC\Compiler Lab\Task 5 Antlr\ P15_40_1234_Nardeen\src

Location of imported grammars

Grammar file encoding; e.g., euc-jp

Package/namespace for the generated code

Language (e.g., Java, Python2, CSharp, ...)

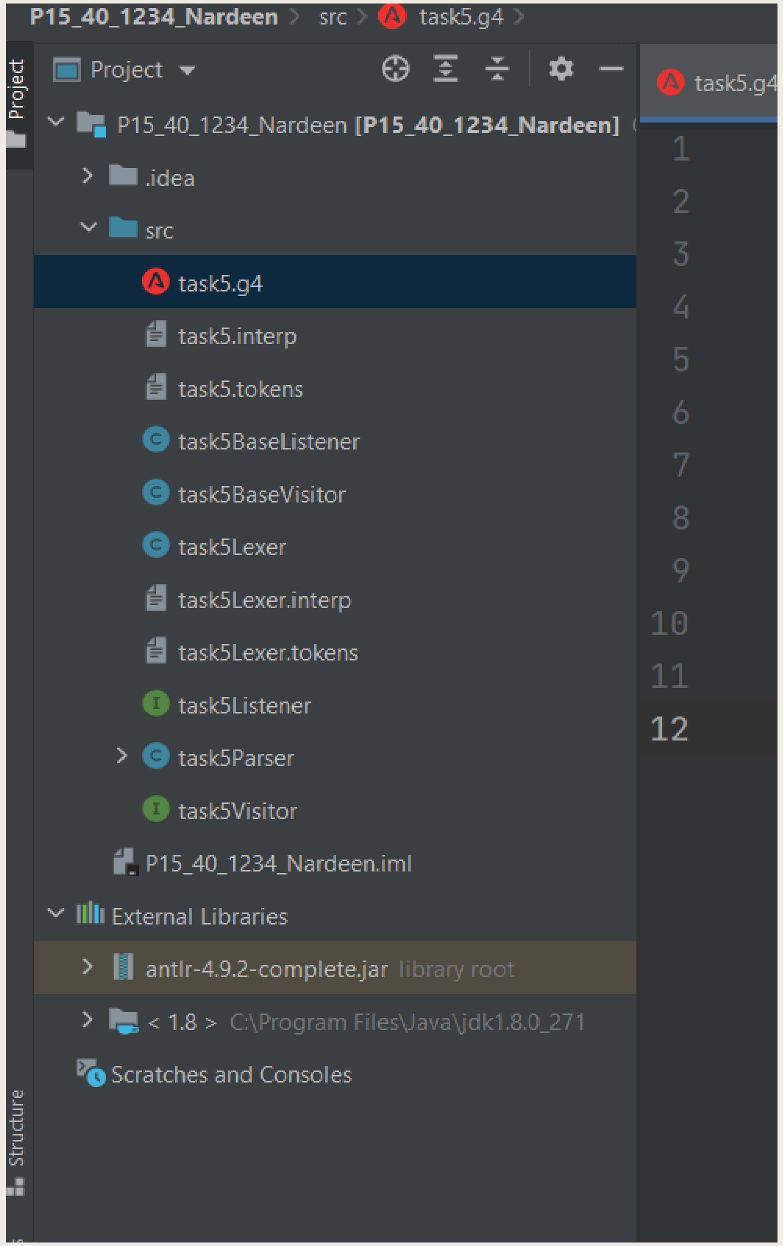
Case transformation in the Preview window: Leave as-is

✓ generate parse tree listener (default)

✓ generate parse tree visitor

Setup

- ➢ Right click in the g4 file > **Generate ANTLR Recognizer**



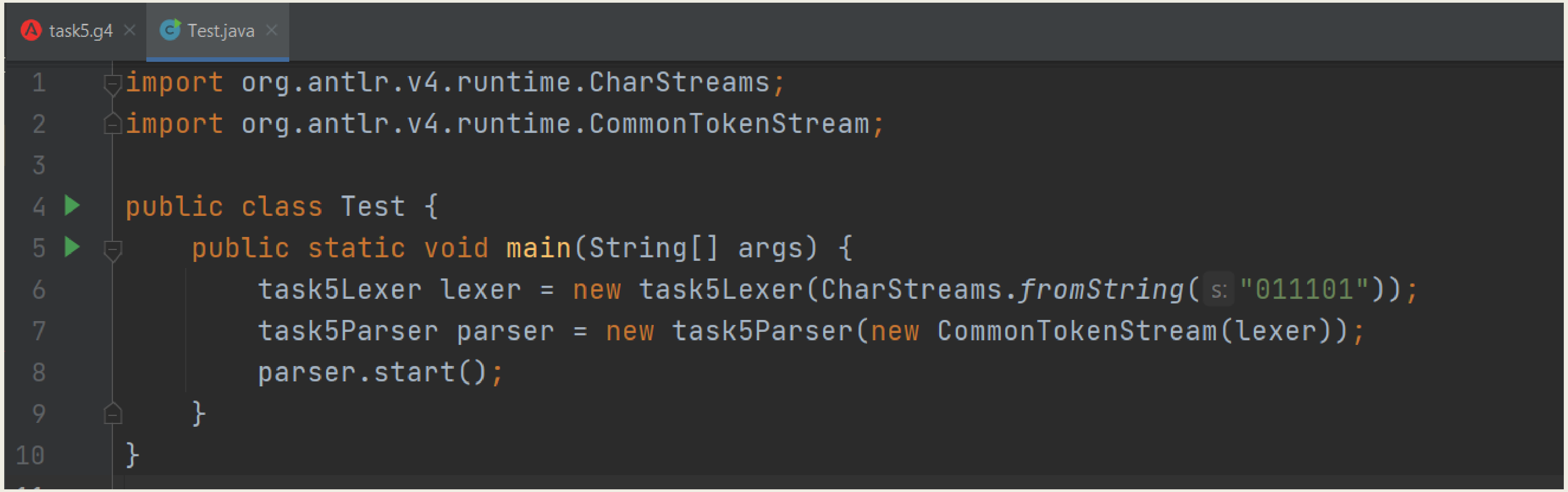
P15_40_1234_Nardeen > src > task5.g4 >

Project

- P15_40_1234_Nardeen [P15_40_1234_Nardeen]
  - .idea
  - src
    - task5.g4
    - task5.interp
    - task5.tokens
    - task5BaseListener
    - task5BaseVisitor
    - task5Lexer
    - task5Lexer.interp
    - task5Lexer.tokens
    - task5Listener
    - task5Parser
    - task5Visitor
  - P15_40_1234_Nardeen.iml
- External Libraries
  - antlr-4.9.2-complete.jar library root
  - < 1.8 > C:\Program Files\Java\jdk1.8.0_271
- Scratches and Consoles

Setup

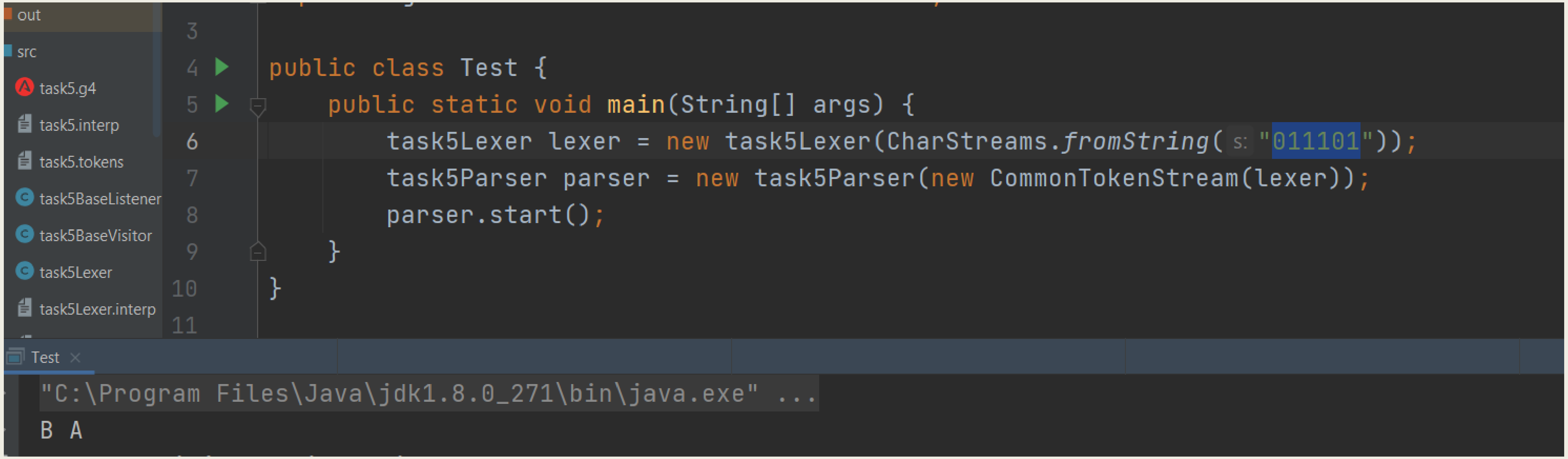
- Create a new java class
  - **New > Java Class > Test**
  - You will find this class on MET website
  - **parser.start()**
    - **start** is the name of the start rule of your grammar


task5.g4 Test.java

```java
import org.antlr.v4.runtime.CharStreams;
import org.antlr.v4.runtime.CommonTokenStream;

public class Test {
    public static void main(String[] args) {
        task5Lexer lexer = new task5Lexer(CharStreams.fromString( s: "011101"));
        task5Parser parser = new task5Parser(new CommonTokenStream(lexer));
        parser.start();
    }
}
```

Setup

- Now you can run your grammar on any binary string
- For example: “0111101” > B A



out
src
task5.g4
task5.interp
task5.tokens
task5BaseListener
task5BaseVisitor
task5Lexer
task5Lexer.interp

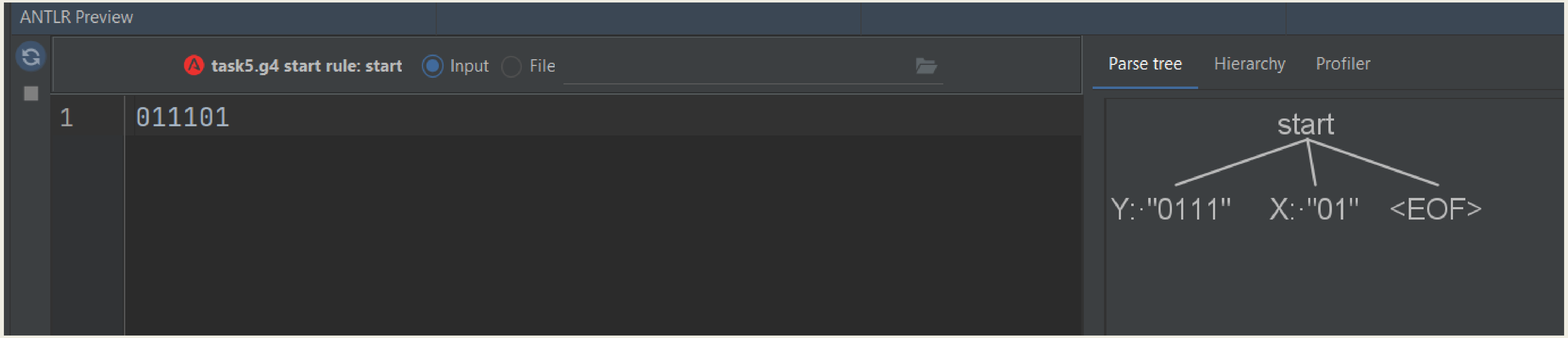
```java
public class Test {
    public static void main(String[] args) {
        task5Lexer lexer = new task5Lexer(CharStreams.fromString( s: "011101"));
        task5Parser parser = new task5Parser(new CommonTokenStream(lexer));
        parser.start();
    }
}
```

Test

```
"C:\Program Files\Java\jdk1.8.0_271\bin\java.exe" ...
B A
```

Setup

- 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 start**
- 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.



```
011101
```