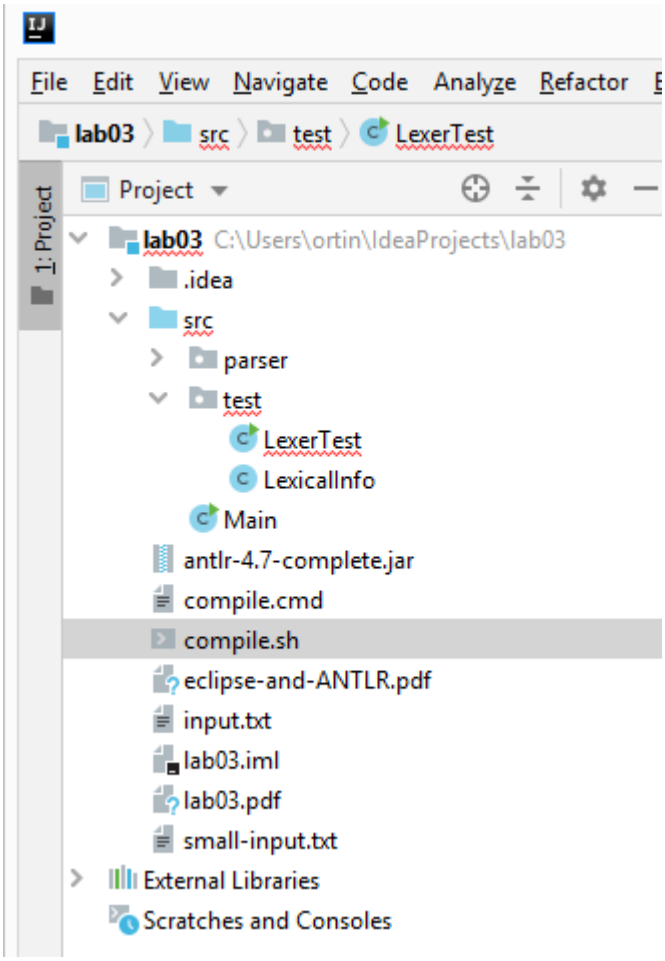
IntelliJ configuration to run ANTLR

ANTLR (ANother Tool for Language Recognition) is a lexer and parser generator for reading, processing and translating source code and binary files. This document shows you how to configure IntelliJ to run ANTLR. Any other Java IDE could be used.

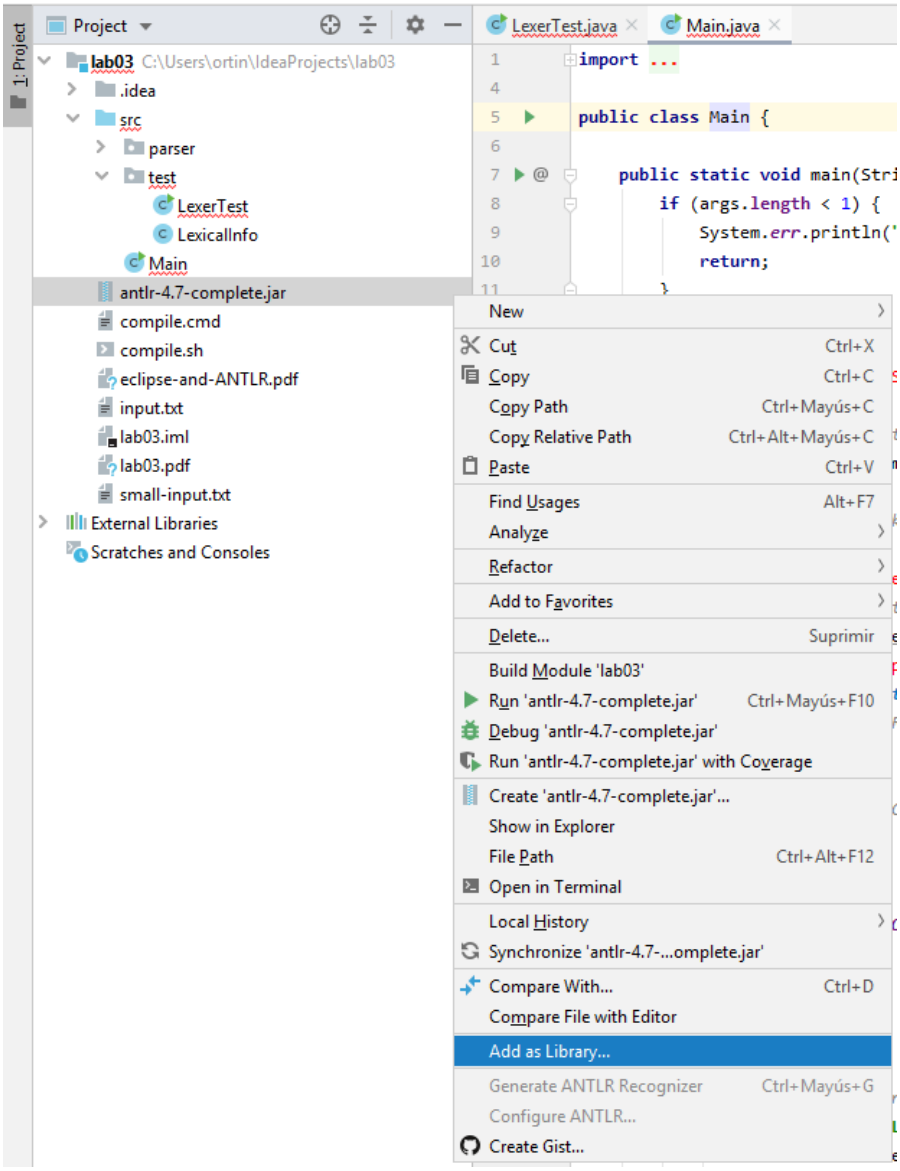
How to create our first lexer / parser in IntelliJ using ANTLR

Create a new empty Java project in IntelliJ.

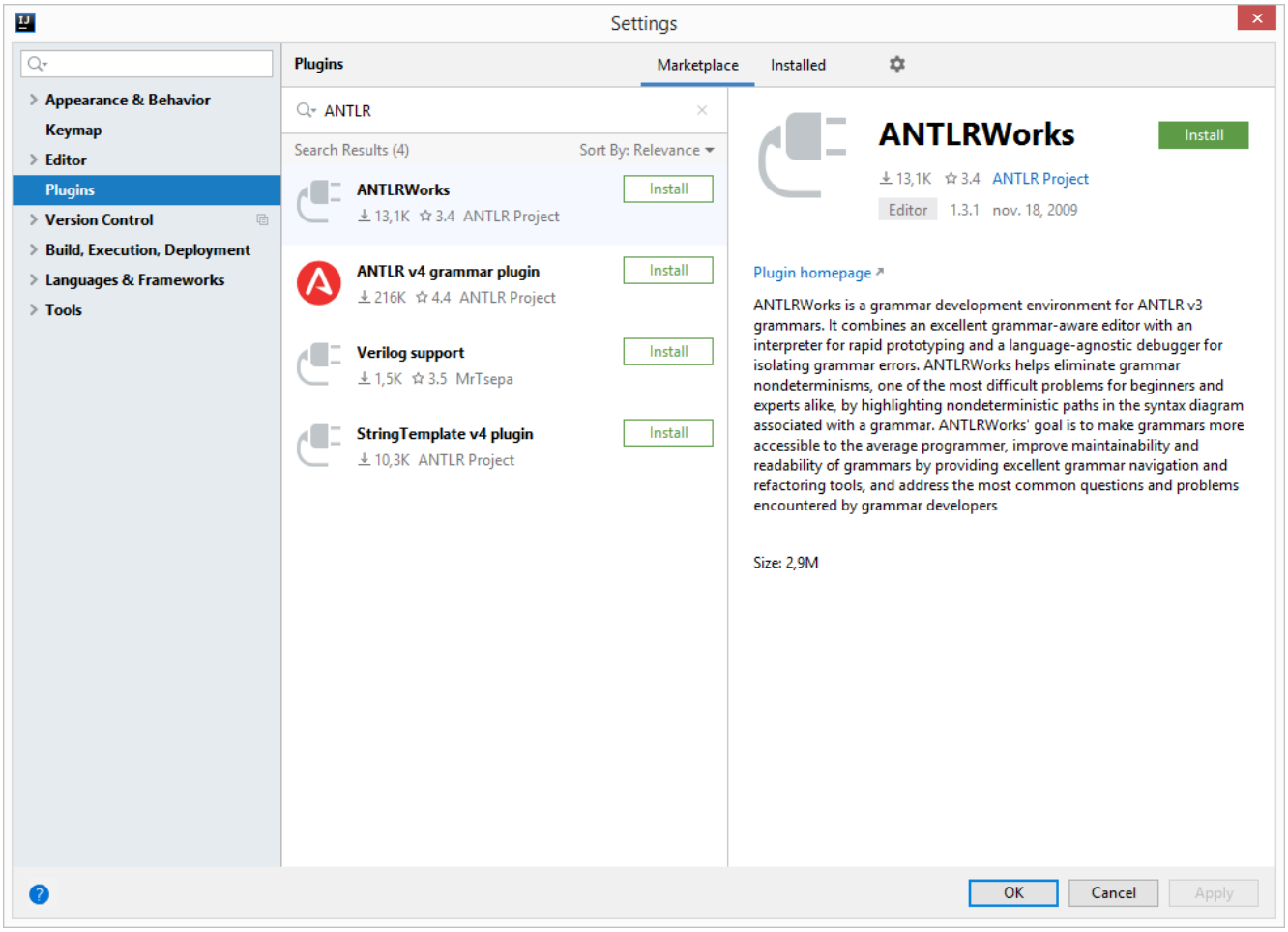
Copy the given files in the project directory (copy and paste, or drag and drop).



Add antlr-4.8-complete.jar as a library.

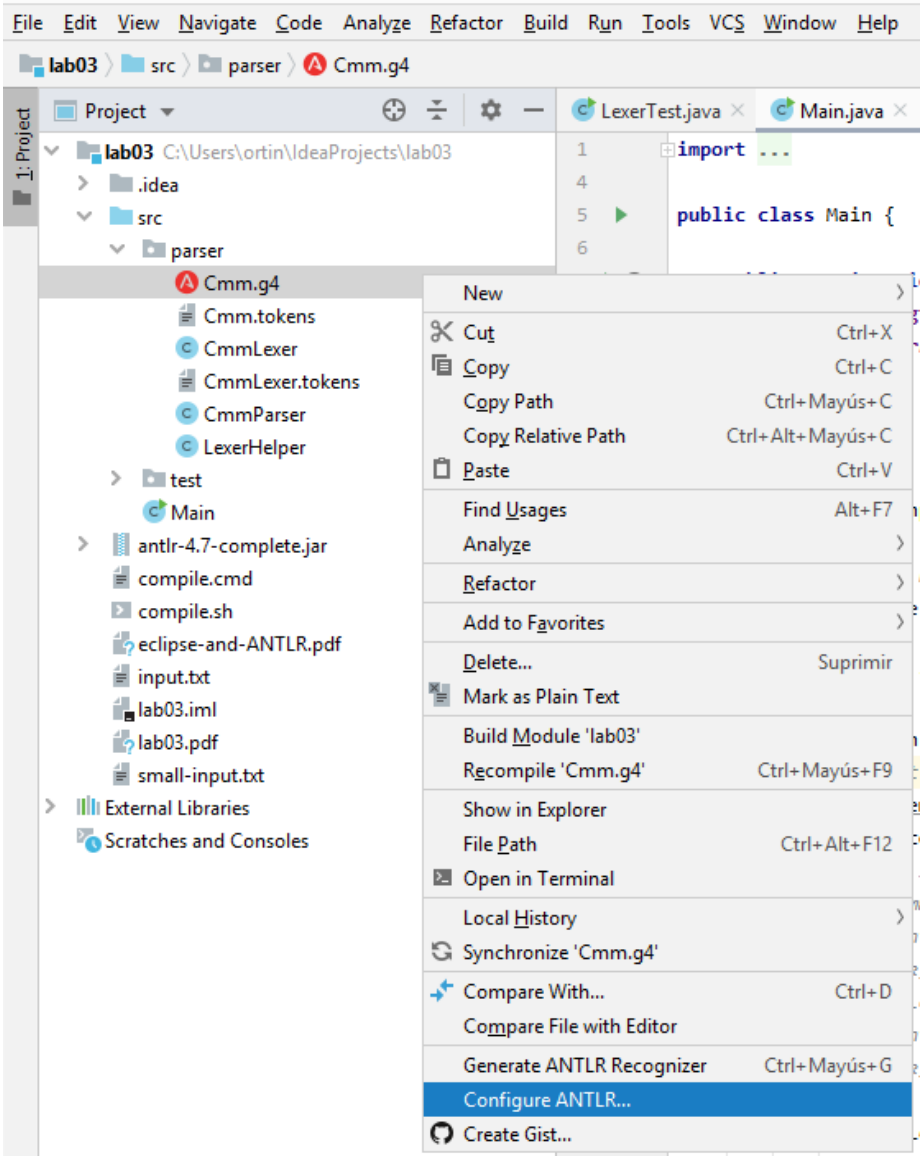


Let's install the IntelliJ plug-in for ANTLR 4. Select the "File" | "Settings" option in the menu. Then, select "Plugins" and, in the textbox under the "Marketplace" tab, write "ANTLR".

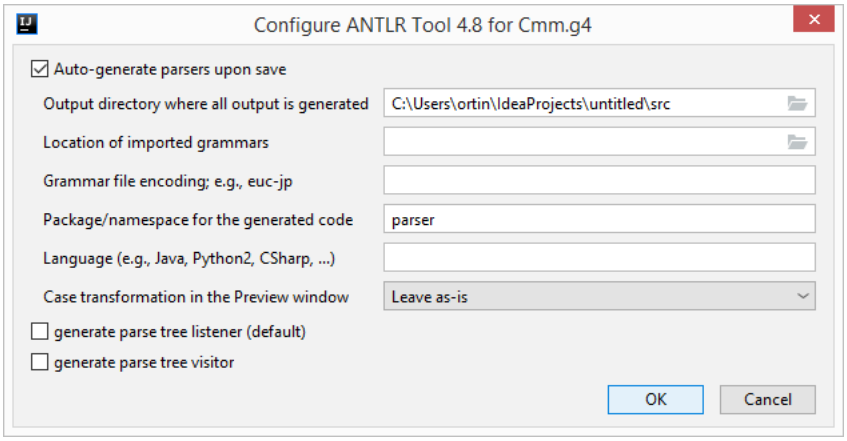


Install the "ANTLR v4 grammar plugin" and restart IntelliJ.

To configure how to generate code for the src/parser/Cmm.g4 grammar specification file, right-click over Cmm.g4 and select “Configure ANTLR...”



Select the output directory as "src", the package for the generated code must be "parser", and disable the "generate parse tree listener" option.



Then, to generate code for Cmm.g4, open the Cmm.g4 file and press "Ctrl + Shift + G" (Generate) or right-click anywhere in the file to select "Generate ANTLR Recognizer".

Remember, every time you modify the g4 grammar specification file, press "Ctrl + Shift + G" to re-generate the lexical and syntactic recognizers.

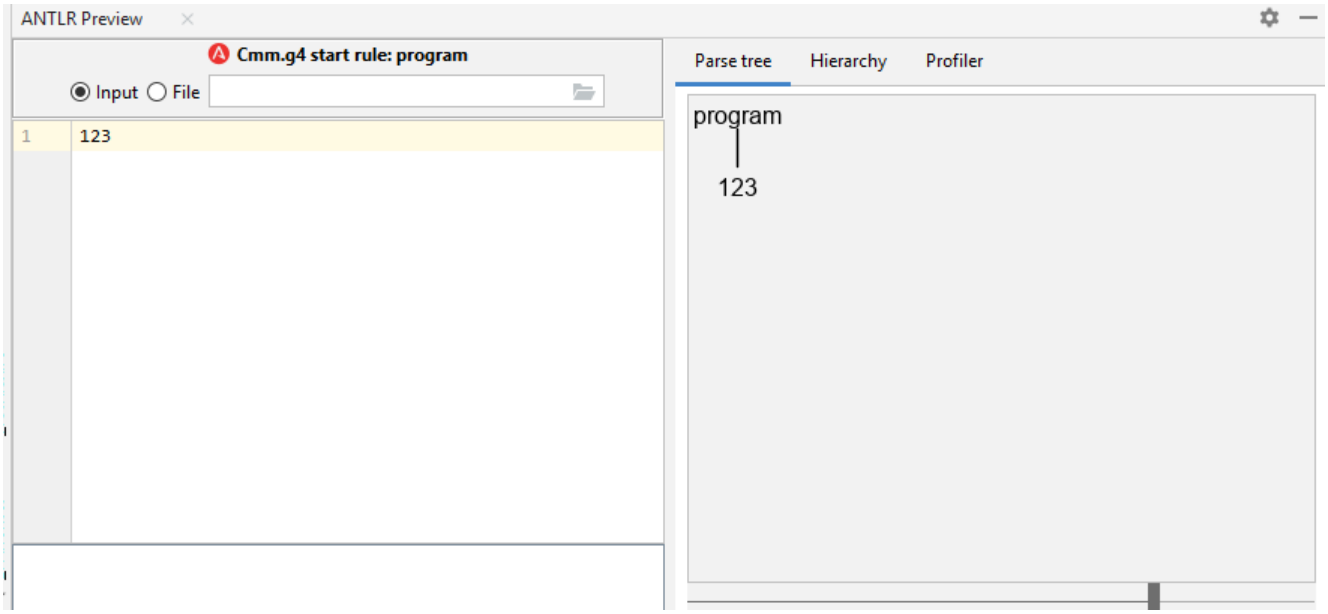
Run the Main Java class, passing small-input.txt as a parameter to see how the lexer works (you main need to comment the whole LexerTest.java file, since it is not complete and has errors).

An important feature of the ANTLR plugin for IntelliJ is that it allows testing productions in the IDE itself. To try it out, modify the "program" production to:

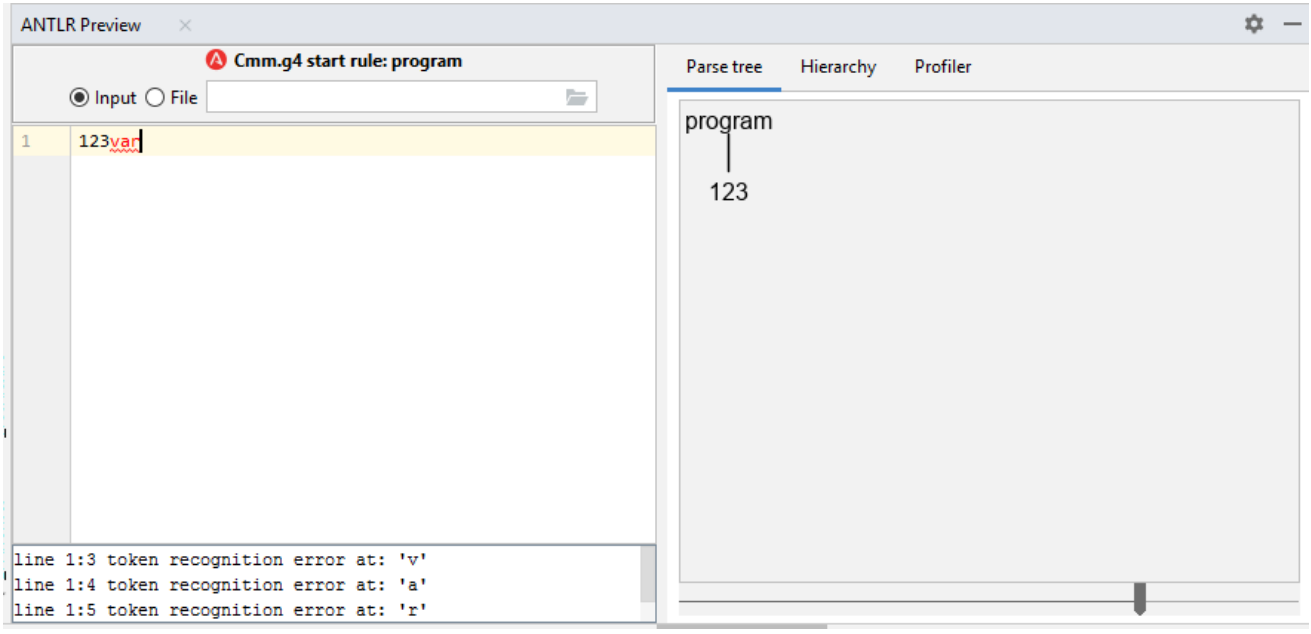
```
program: INT_CONSTANT*
       ;
```

Then, right-click over the "program" production, and select "Test Rule program".

In the left hand side window of the ANTLR preview, we can input text for that production (a file can also be analyzed). On the right hand side, the parse tree for that input program is shown.



If we write an erroneous input, it is checked by ANTLR, showing the appropriate message.



This feature is very powerful to test productions (and whole language specifications) of syntax analyzers. It is also quite helpful for token recognition: just add the sequence of tokens you want to test as the right-hand-side of the "program" non-terminal, and you can see if it works as expected.