Step by step creation of a simple compiler using ANTLR4

What is ANTLR?

ANTLR (ANother Tool for Language Recognition) is a powerful framework for building programming language parsers, interpreters, and compilers. It was created by Terence Parr and first released in 1989. ANTLR provides a robust set of tools and libraries that simplify the process of creating language processors.

Why ANTLR?

At its core, ANTLR is a **parser generator** that takes a formal description of a language (specified using its own grammar syntax) and generates code that can parse input conforming to that language. It supports both lexer (tokenization) and parser (syntactic analysis) phases, making it capable of handling a wide range of language structures and grammatical rules.

One of the distinguishing features of ANTLR is its support for LL(*) parsing, which stands for “lookahead left-to-right with arbitrary lookahead.” This parsing technique allows ANTLR to handle ambiguous grammars and resolve conflicts efficiently. ANTLR-generated

parsers are typically efficient and can produce helpful error messages when encountering invalid input.

In addition to parsing, ANTLR provides features for constructing abstract syntax trees (ASTs), generating **visitors** or **listeners** for **AST traversal** (more on that later), and performing semantic analysis on the parsed input.

ANTLR supports multiple target languages, including Java, C#, Python, and more. This flexibility allows developers to choose their preferred programming language for implementing the actions associated with parsing rules which makes ANTLR a popular choice for building language processors and facilitating language-related tasks in software development.

What is an AST traversal?

AST traversal refers to the process of traversing or visiting the nodes of an Abstract Syntax Tree (AST). An Abstract Syntax Tree is a hierarchical representation of the syntactic structure of a program or a piece of code. It captures the relationships between different language constructs, such as statements, expressions, and declarations.

During AST traversal, each node of the AST is visited to perform specific operations or analyses. This allows to extract information, perform transformations, or generate code based on the structure and content of the AST.

There are generally two common approaches to AST traversal: visitor pattern and listener pattern.

1. *Visitor Pattern:* In the visitor pattern, visitor classes are defined to encapsulate operations on different nodes of the AST. Each visitor class typically has methods corresponding to different node types. By traversing the AST, the appropriate visitor methods are invoked for each visited node, allowing to perform specific operations or gather information.
2. *Listener Pattern*: In the listener pattern, listener classes are defined to respond to events triggered during the traversal of the AST. Listeners are associated with specific node types and are notified when those node types are encountered

during traversal. Listeners can react to these events by executing custom logic or modifying the AST.

Both patterns offer flexibility in performing operations during AST traversal, and the choice between them depends on the requirements of each use case.

AST traversal is commonly used in various language processing tasks, including static analysis, code generation, optimization, and refactoring. By traversing the AST, the structure and content of a program can be accessed and analyzed in a systematic and organized manner.

Overall, AST traversal is an essential technique for extracting information, performing analyses, and manipulating code during the language processing pipeline.

ANTLR4 for Windows

To install ANTLR on Windows, we need to follow these steps:

Step 1: Set up Java

ANTLR requires Java to run. Ensure that you have Java Development Kit (JDK) installed on your Windows system. You can download the JDK from the Oracle website at https://www.oracle.com/java/technologies/javase-jdk11-downloads.html. Follow the installation instructions provided by Oracle for your specific version of the JDK.

//Side notes:

Do I need also to download JRE? Actually, in the case of ANTLR, we only need to install the JDK on Windows. The JDK includes both JRE and the necessary tools for Java development, so it is sufficient for running ANTLR.

What is the difference between JDK and JRE? The JDK (Java Development Kit) and JRE (Java Runtime Environment) are both components of the Java platform, but they serve different purposes:

1.JDK (Java Development Kit) is a software development kit that provides tools and libraries necessary for Java development. It includes the JRE, along with additional tools like the Java compiler (javac), debugger (jdb), and other utilities. The JDK is

used by developers to write, compile, and debug Java programs. It also includes development-related documentation and examples.

2.JRE (Java Runtime Environment): The JRE is an environment that allows to run Java applications on computer. It includes the Java Virtual Machine (JVM), which is responsible for executing Java bytecode, as well as core libraries and other runtime components required to run Java applications. The JRE does not include the development tools and utilities found in the JDK.

In summary, the JDK is used for Java development and includes the JRE, while the JRE is used to run Java applications but does not include the development tools. If you are developing Java applications, you need the JDK installed on your system. If you only need to run Java applications, the JRE is sufficient. It's worth noting that starting from Java 11, Oracle has changed the distribution of Java, where they only provide the JDK and no longer separate the JRE. The JDK now includes everything needed to run Java applications as well.

//End of side note

Step 2: Download ANTLR

Visit the official ANTLR website at https://www.antlr.org/download.html and navigate to the download page. Look for the latest release of ANTLR and download the ANTLR binary jar file (antlr-4.x-complete.jar). At the time of writing this tutorial, the link for download is ANTLR tool itself. At the time of writing this tutorial the version: ***antlr-4.12.0-complete.jar*** is the one available for download and the one used in this tutorial.

Once donwloaded, we need to put the ANTLR jar file (***antlr-4.12.0-complete.jar***) into a folder we will need later on. In our case, we will create a folder named JavaLib put it into the C:\ root and copy the ANTLR jar file inside of it.

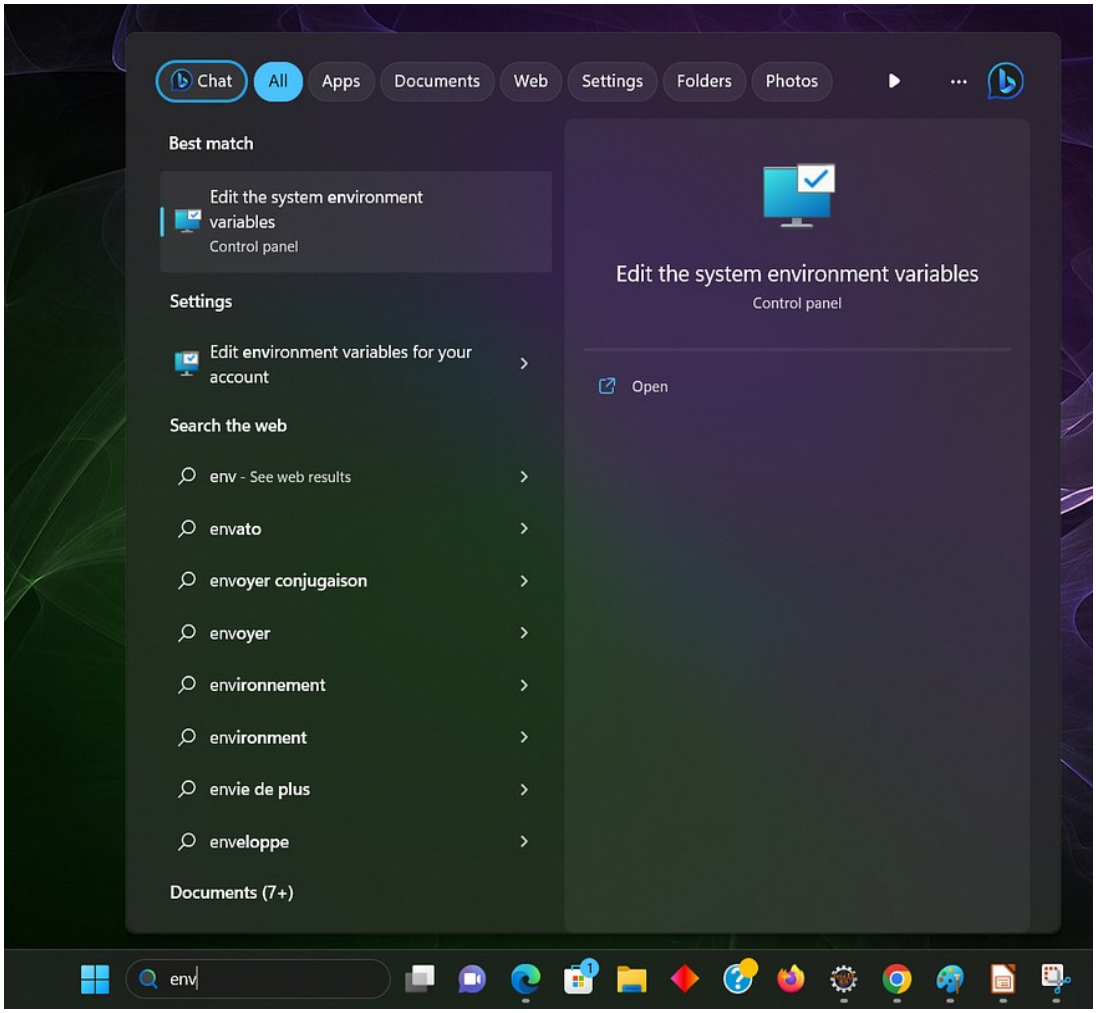
Step 3: Set up the CLASSPATH

Once the ANTLR binary jar file downloaded, we need to set up the CLASSPATH environment variable to include the location of the jar file.

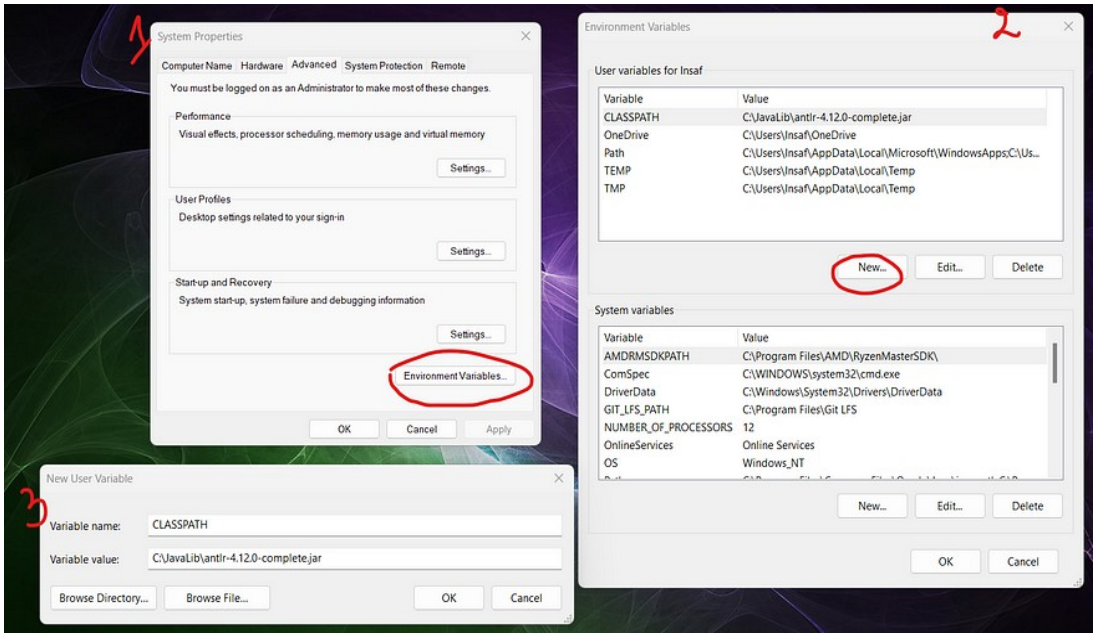
To access the environment variables, open "Control Panel." In the Control Panel, change the view to either "Large icons" or "Small icons" so that you can see all the available

options. Look for and click on the “System” or “System and Security” option. In the System or System and Security section, you should find an option called “System.” Click on it. In the System window, click on the “Advanced system settings” link on the left side. The System Properties window will open with the “Advanced” tab selected. In the “Advanced” tab, click on the “Environment Variables” button near the bottom of the window.

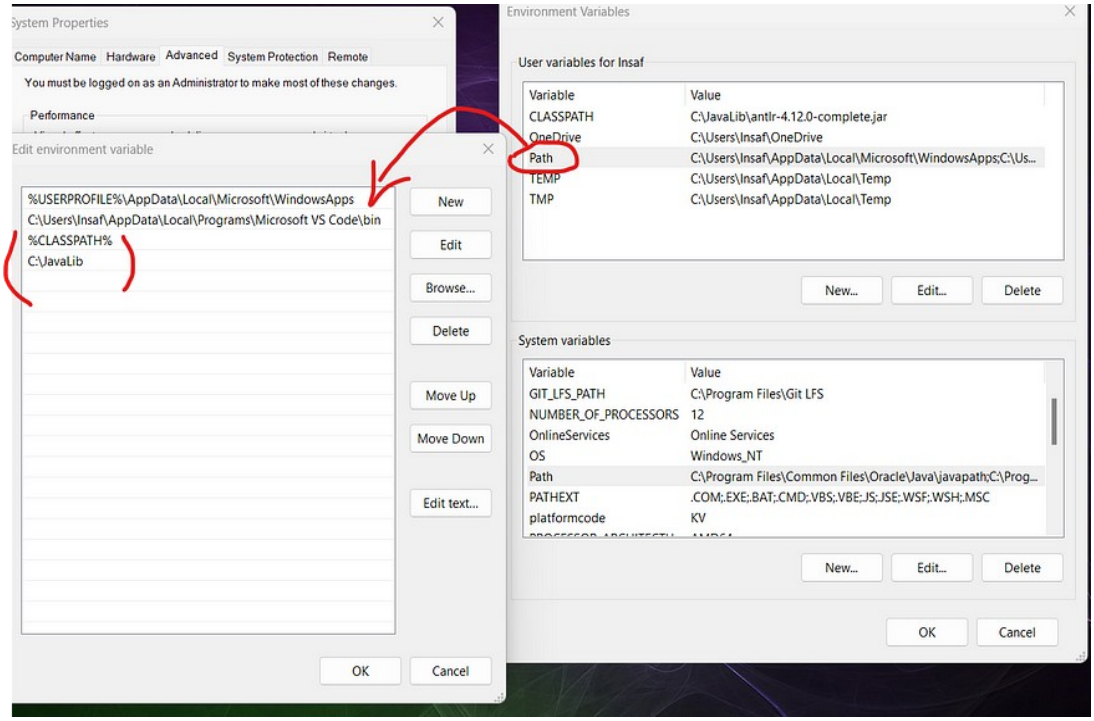
You can also do that by going to ytstem and Security, click on System, and then click on “Advanced system settings” on the left sidebar. We can also do that by typing env in the search bar of windows 11:



In the Environment Variables window, under the “System variables” section, click on “New” to add a new variable. Set the variable name as “CLASSPATH” and the variable value as the full path to the downloaded ANTLR jar file.



In the use variable path, add the path of the ANTLR jar file as well as the %CLASSPATH% variable we just created. To do so, double click on the oath of the User variables:



//Side note

What is %CLASSPATH%: In a Windows environment, the `%CLASSPATH%` variable is an environment variable that defines the search path for Java class files and libraries. It is used by Java applications and tools to locate and load classes and resources.

When a Java program is executed, it needs to find the necessary class files and libraries required for its execution. The `%CLASSPATH%` variable specifies the directories or JAR files where Java should look for these files.

Here's a breakdown of what `%CLASSPATH%` stands for:

- `%`: The percent sign is used as a prefix and suffix to indicate an environment variable in Windows.
- `CLASSPATH`: The name of the environment variable. It specifically refers to the Java classpath.

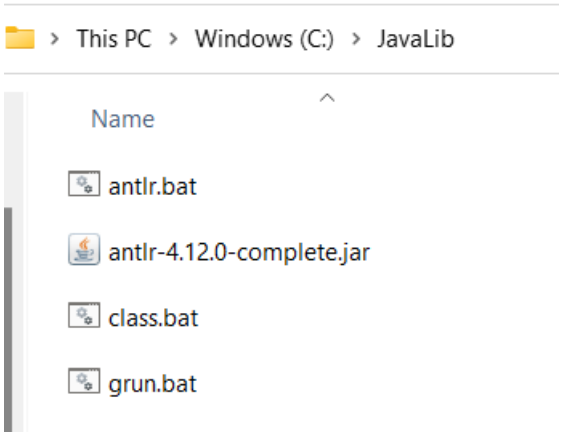
By setting the `%CLASSPATH%` variable, you can specify one or more directories or JAR files separated by semicolons (;). Java will search for class files and resources in these locations when executing a Java program.

It's worth noting that modern Java development practices typically rely more on build tools like Apache Maven or Gradle, which manage dependencies and classpath configurations automatically. However, the `%CLASSPATH%` variable can still be used in certain cases where manual classpath configuration is necessary.

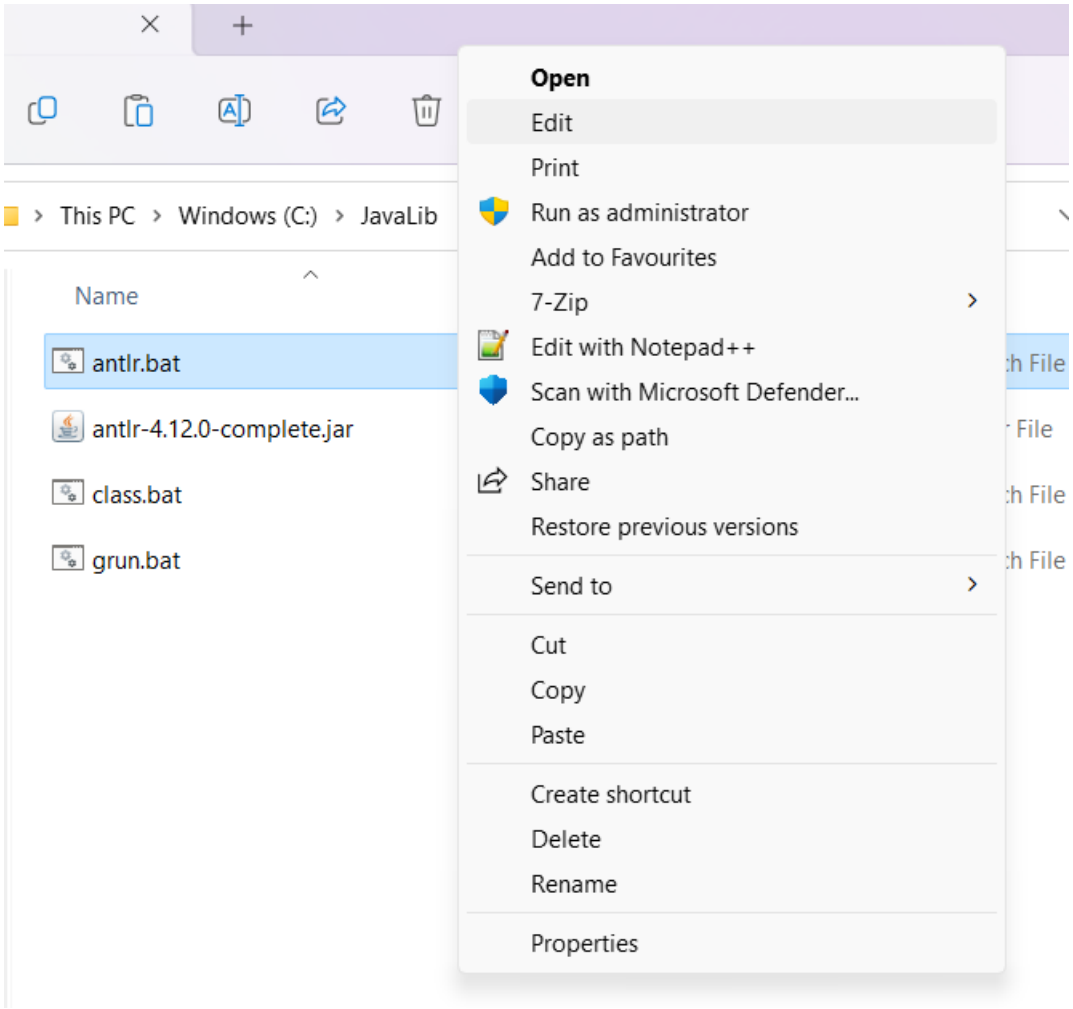
//End of side note

Step 4: Create the batch files

We will need to create three files: ***antlr.bat***, ***class.bat*** and ***grun.bat***. The previous ***JavaLib*** folder we created will look like this:



We need then to edit the three bat files. Choose any editor of your convenience. In our case, we will use notepad.



For antlr.bat copy the following code:

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

In case antlr is still not recongnized, you can write the following instead:

```
java -cp ".;C:\JavaLib\antlr-4.12.0-complete.jar" org.antlr.v4.Tool %*
```

For class.bat the following:

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

And for grun.bat the following:

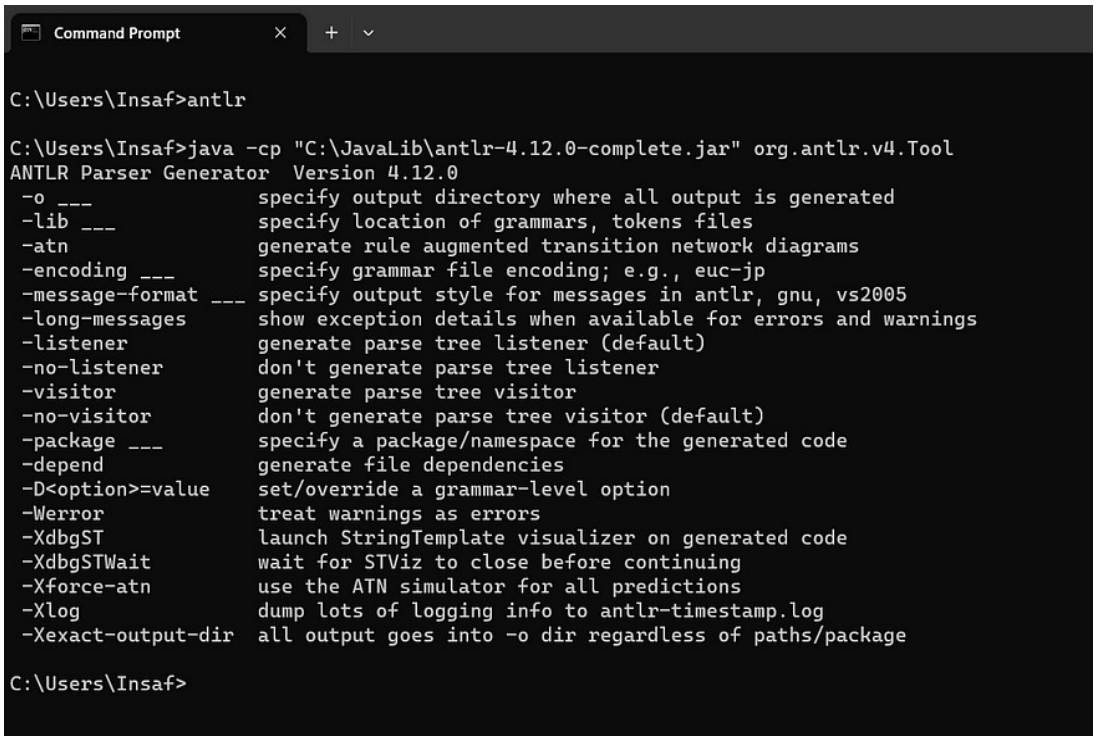
```
java org.antlr.v4.gui.TestRig %*
```

Similarily, if grun does not work; try the following:

```
Java -cp ".;C:\JavaLib\antlr-4.12.0-complete.jar" org.antlr.v4.gui.TestRig %*
```

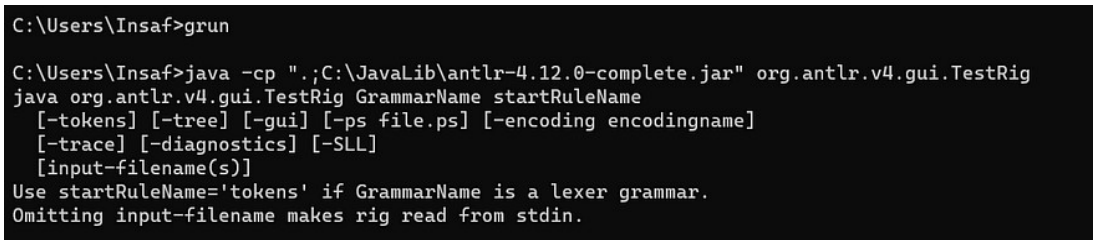
Step 5: Check the installation

The first thing we need to do is to make sure that ANTLR is properly installed. To do so, we will go to the command prompt and type antlr. To open the command prompt, we will just type *cmd* in the search bar of windos. If ANTLR is properly installed, typing antlr in the cmd will give details about the version installed:



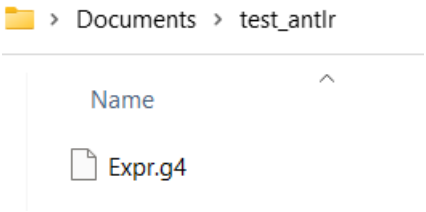
We can see that the version installed in 4.12.0

Running the grun batch file will also reflect that our system knows ANTLR. The result is as follows:



Step 6: Test ANTLR with a simple grammar

Our final step is to test the ANTLR library with a very basic grammar. Let us create a folder and a file with g4 extension for that:



Copy paste the following content into the *Expr.g4* file and save the file:

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

Now, to run ANTLR with the grammar we just created, we need to point to the corresponding file in the command prompt. Our grammar is saved in a folder named test_antlr inside Documents:

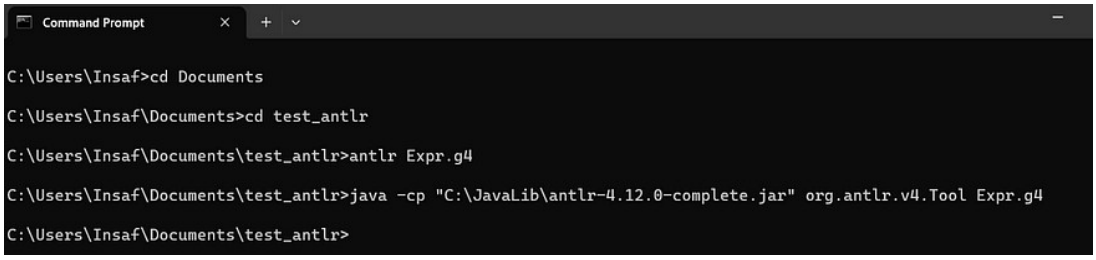
```
cd Documents


cd test_antlr
```

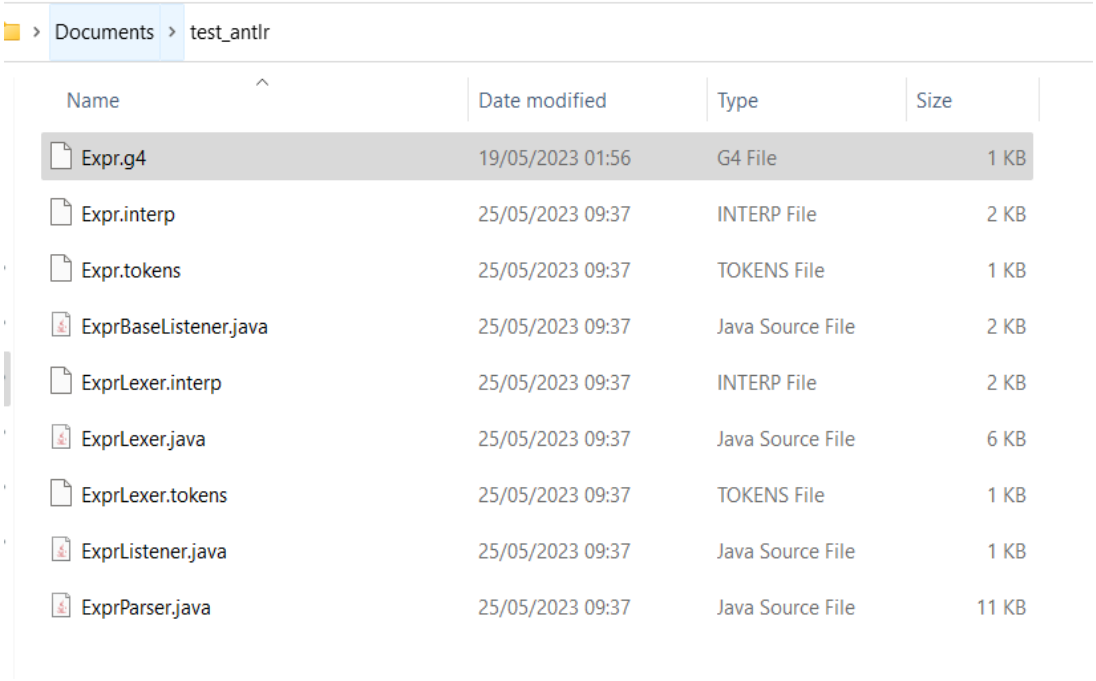
To generate the java code and classes for this grammar we need to write in the command prompt:

```
antlr Expr.g4
```

The result will be the set of java and class files generated in the folder we created our grammar:



command lines in the command prompt

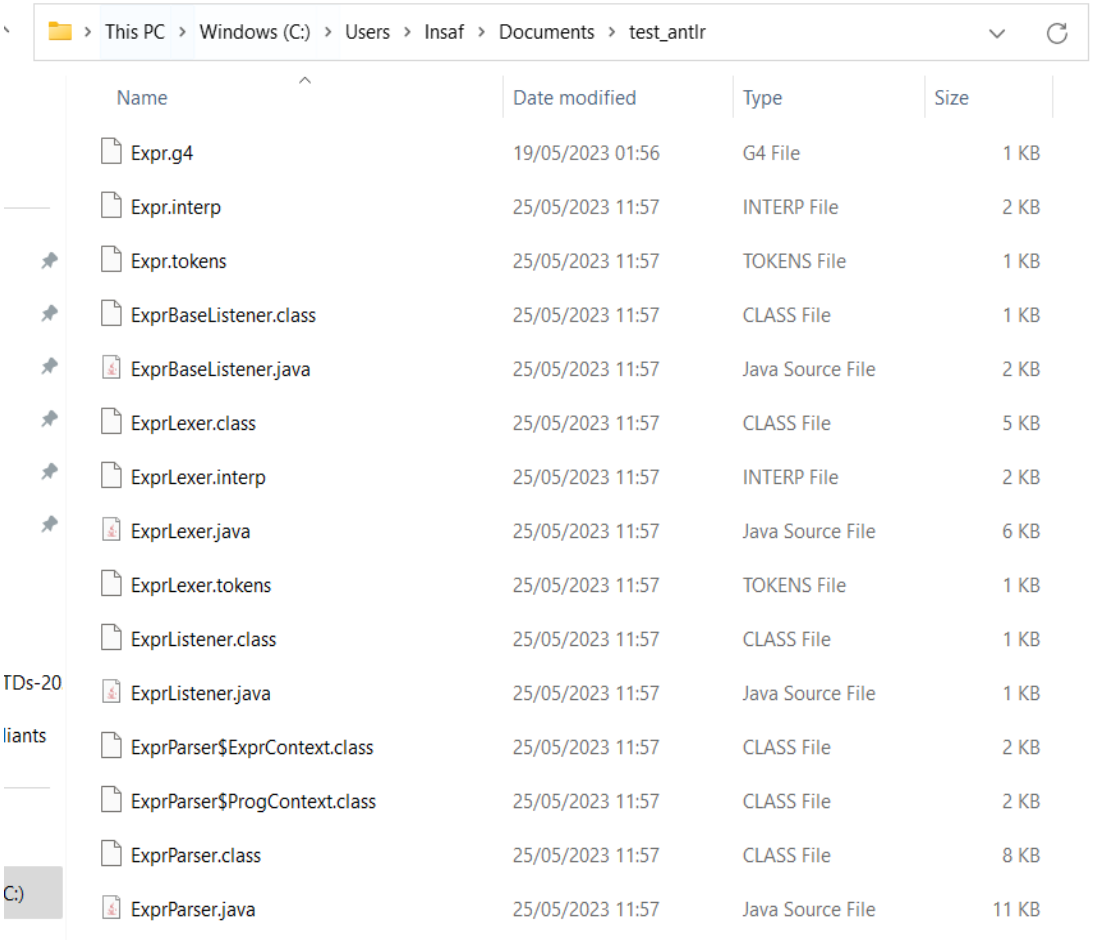


result of the previous command lines (generation of the java files)

Now that our javafiles have been generated, we can compile and run them. To compile the java file simply use the javac (java compiler):

```
javac Expr*.java
```

The generated java class files will be in the same folder as previously:



to run the java codes with the user interface we will type:

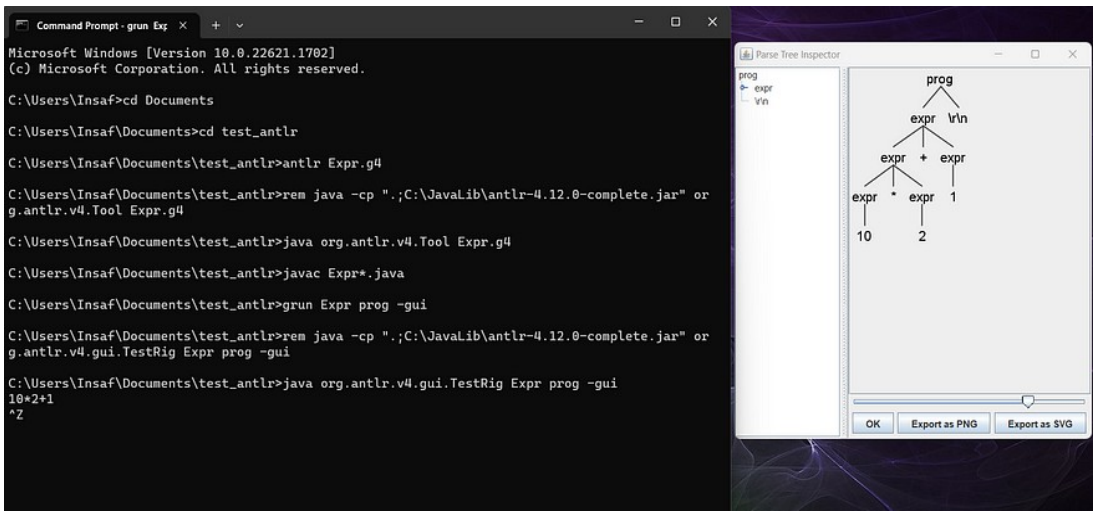
```
grun Expr prog -gui
```

Note the parameters of grun: the grammar file name (*Expr*) and the grammar name (*prog*). *-gui* is used to ask grun to launch the graphical interface to show the execution tree.

This command will expect an example of code that will follow the grammar we defined. Let us take the example:

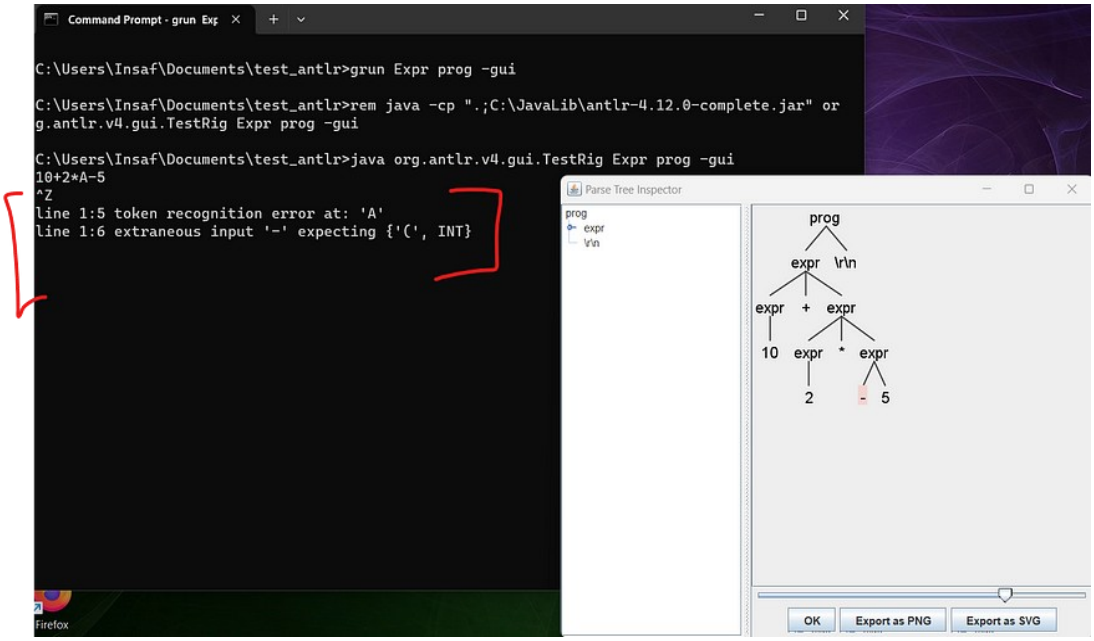
```
10*2+1
```

Since we are testing on the command prompt, we need to tell the command that after (*10+2+1)* we need to add an EndOfFile, which can be done with *Ctrl+Z.* ANTLR will now analyse the code (*10*2+1*) according to the grammar we defined. To visualize the execution tree, we will type *ctrl+d.* Note that this will take some time.



A new window is opened to show the grammar. You can close the windows by typing *Ctrl+C.*

Note that if we write a code that does not respect our defined grammar, then ANTLR will show us the erros but still the tree will be shows:



Well, all done. Enjoy using ANTLR !