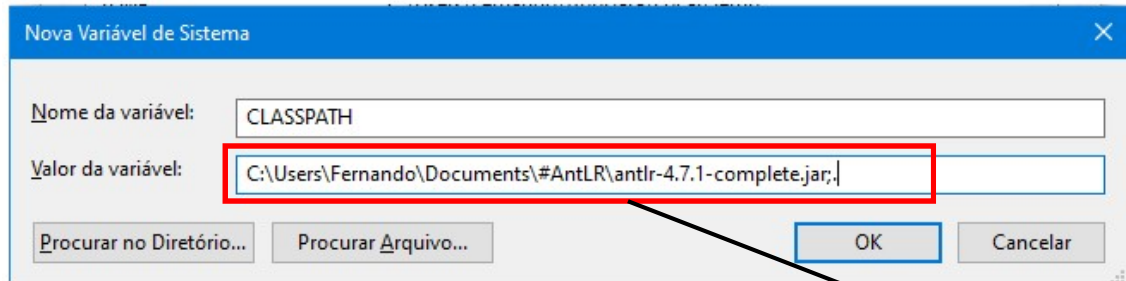


Configuração do AntLR

Caso **NÃO** tenha criado a variável de ambiente **CLASSPATH**

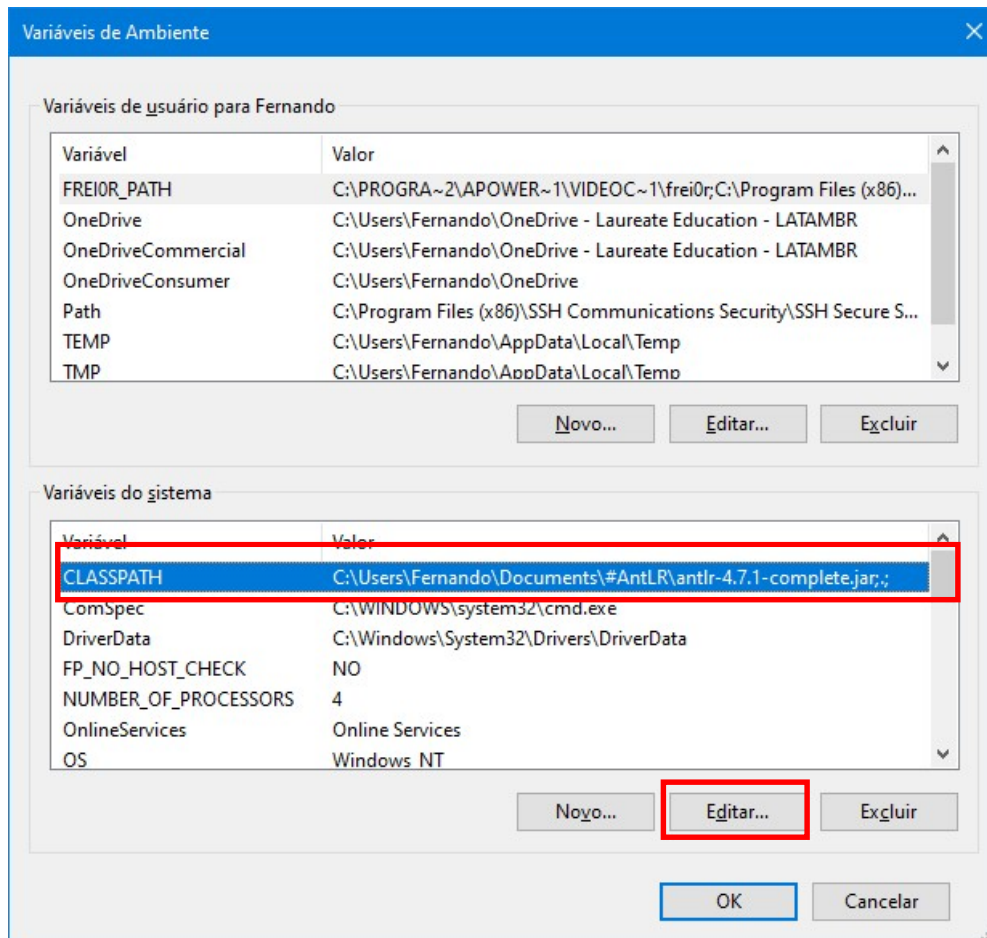
Em *valor de variável* insira o caminho, com o nome do arquivo inclusive, seguido de um ponto e vírgula e um ponto.



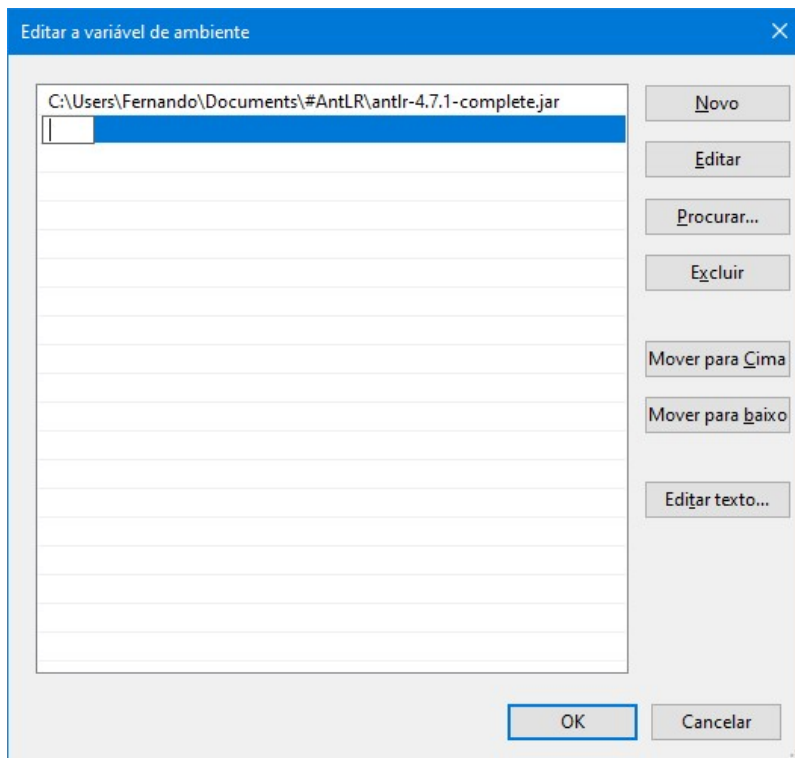
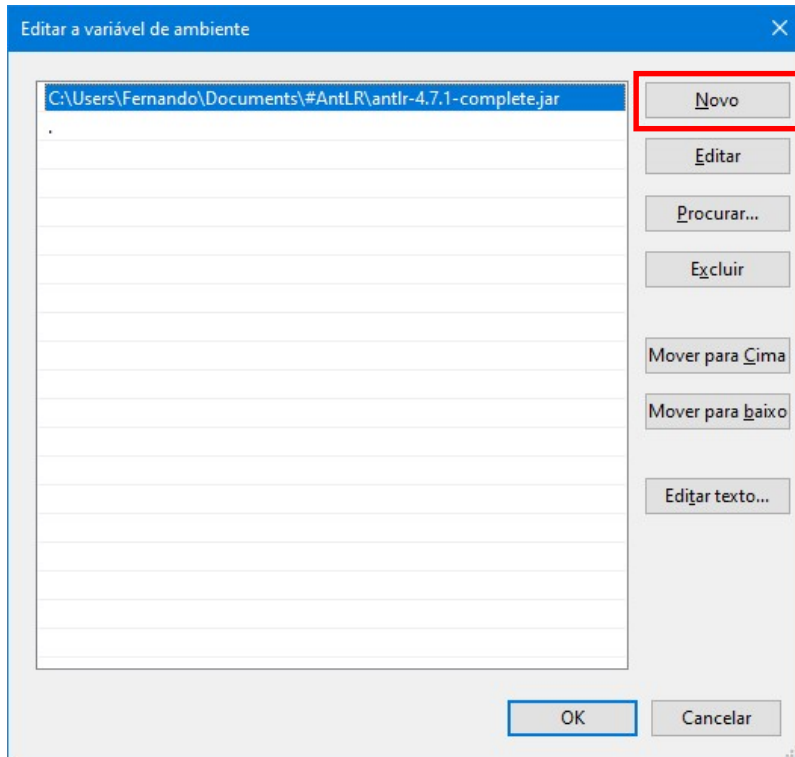
C:\Users\Fernando\Documents\#AntLR\antlr-4.7.1-complete.jar;.

Caso **JÁ** tenha criado a variável de ambiente **CLASSPATH** e queira realizar o “reparo”

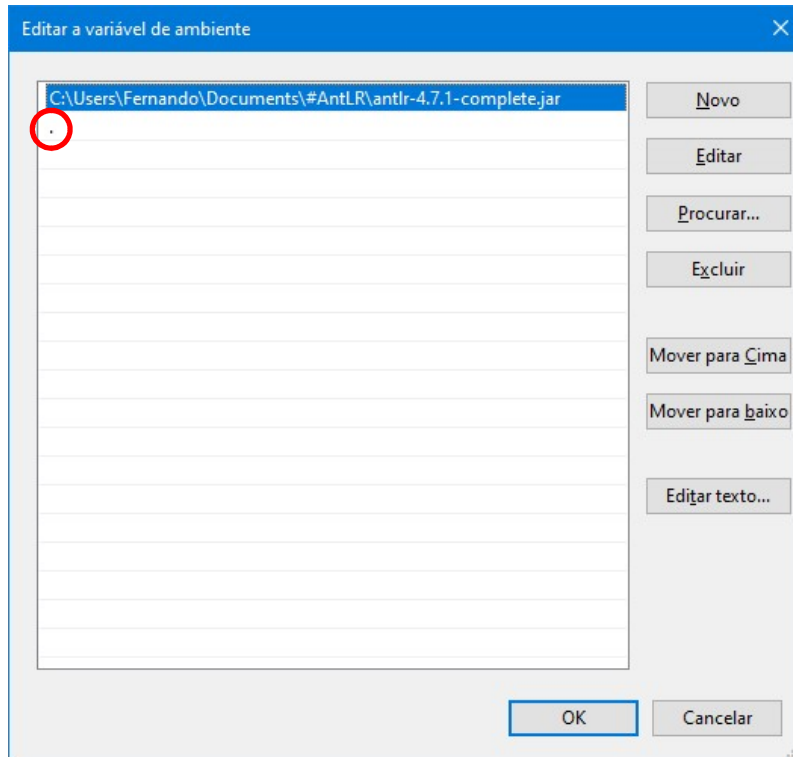
Se já tenha configurado a variável de ambiente, clique na variável **CLASSPATH** e clique em editar



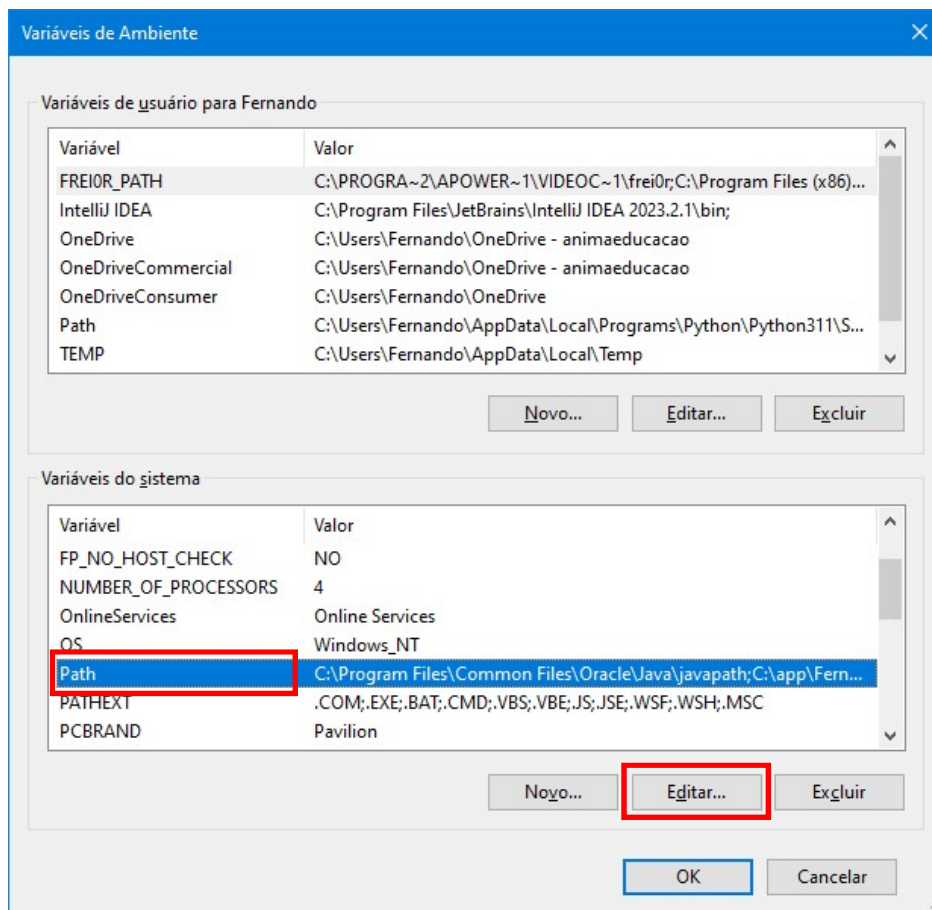
Em seguida clique em novo para adicionar mais um valor



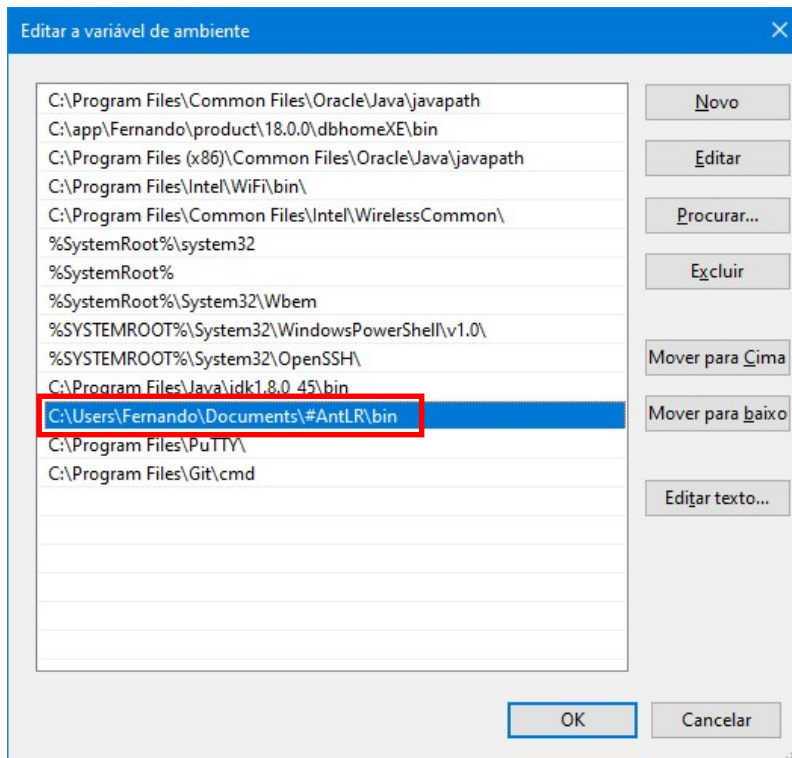
Digite o '.' (ponto) como mais um registro na variável de ambiente



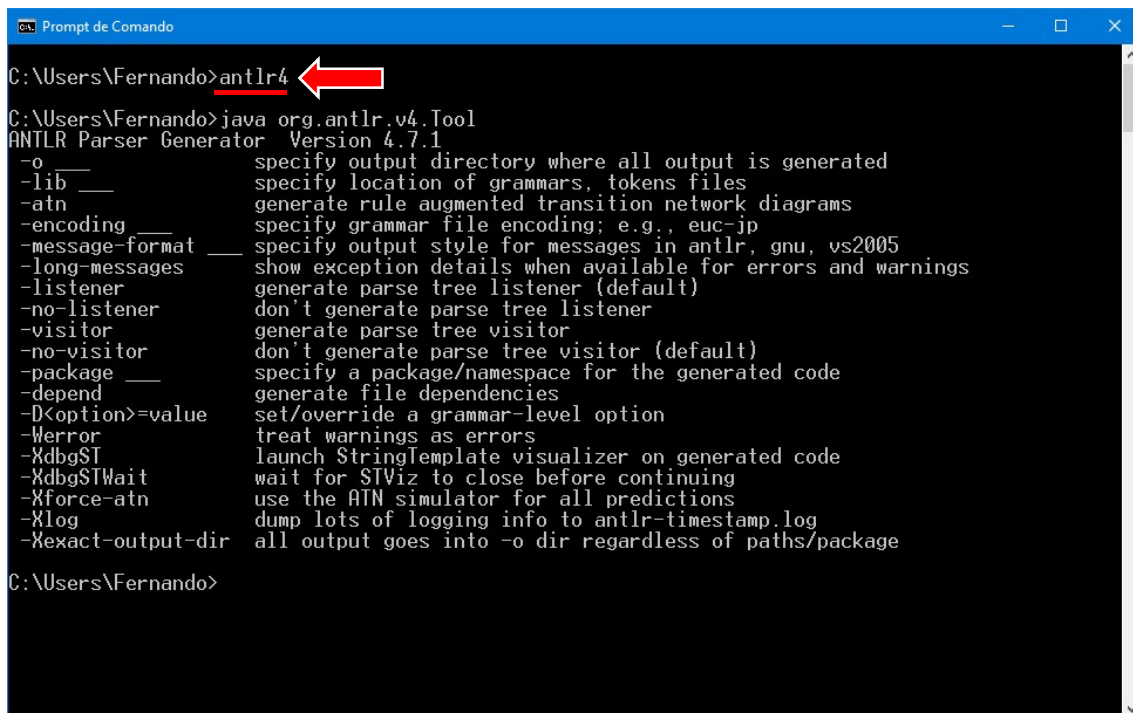
Agora procure a variável de ambiente **Path** e clique em editar



Em seguida adicione o caminho até o diretório **bin** do AntLR



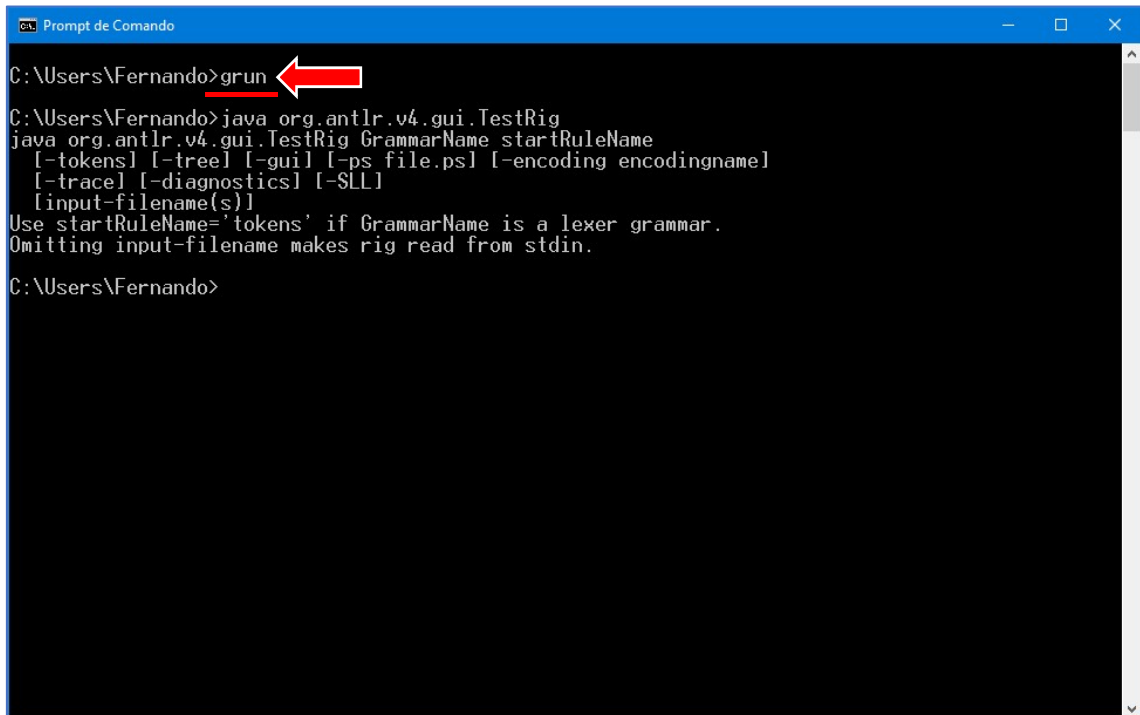
Vamos fazer um teste para verificar se o AntLR está funcionando corretamente. Abra o prompt de comando e digite **antlr4**. Se tudo estiver OK a mensagem na tela será semelhante a essa da figura abaixo.



```
C:\Users\Fernando> antlr4
C:\Users\Fernando> java org.antlr.v4.Tool
ANTLR Parser Generator Version 4.7.1
-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

C:\Users\Fernando>
```

Faça outro teste digitando **grun** e veja se aparece a mensagem como na figura a seguir.



```
C:\Users\Fernando>grun
C:\Users\Fernando>java org.antlr.v4.gui.TestRig
java org.antlr.v4.gui.TestRig GrammarName startRuleName
[-tokens] [-tree] [-gui] [-ps file.ps] [-encoding encodingname]
[-tracel [-diagnostics] [-SLL]
[input-filename(s)]
Use startRuleName='tokens' if GrammarName is a lexer grammar.
Omitting input-filename makes rig read from stdin.
C:\Users\Fernando>
```

The screenshot shows a Windows Command Prompt window titled "Prompt de Comando". The user has entered the command `grun`, which is underlined with a red line and has a red arrow pointing to it. The command has been executed, resulting in the output shown above. The output includes the Java command to run `TestRig`, its usage instructions, and the current directory path.

Agora sim o AntLR está funcionando corretamente.