Generar automáticamente un parser de Clafer a través de su gramática

# Requisitos

* El fichero de la gramática de Clafer (clafer.cf).  
  clafer.cf: <https://www.clafer.org/p/documentation.html>
* La utilidad “bnfc” para generar el código Java a partir de la gramática. No hace falta añadirla al path.  
  BNFC: <https://bnfc.digitalgrammars.com/>
* La utilidad “antlr” para generar el código Java a partir de la gramática. Hace falta añadirla al path y al classpath (instrucciones: https://stackoverflow.com/questions/41021963/how-to-install-antlr4)  
  ANTLR: <https://www.antlr.org/>
* La utilidad “make” para compilar el código y enlazar “bnfc” y “antlr”. Make ya está disponible en Linux, en Windows descargar “MinGW”  
  MinGW: <http://www.mingw.org/>

# Procedimiento

1. En una carpeta poner la gramática (clafer.cf) y el ejecutable “bnfc”. En una línea de comandos ejecutar:

**bnfc --java-antlr -m clafer.cf**

Esto genera el código Java para la utilidad “antlr”.

Problemas con la gramática de Clafer:  
Saltarán varios errores debido a nomenclatura de los elementos en la gramática. En este punto se puede forzar la herramienta para continuar con los errores pero yo he preferido resolverlos (desconozco si aparecen más errores en los procesos posteriores si no se resuelven ahora).  
Para resolverlos abrir la gramática y cambiar el nombre de los elementos indicados en la salida de la consola (cambiar solo los elementos del principio de cada línea, es decir, aquellos que terminan con un punto al principio de la línea y no las reglas de la gramática). Por ejemplo, cambiar “Module.” Por “MModule” o por “Module2”.

1. Compilar con “make”. En una terminal ejecutar:

**make**

Esto compilará el código usando la librería “antlr” y generará las clases finales que necesitamos con el Lexer y el Parser de Clafer.

##### Problemas con el código generado:

Saltarán varios errores al compilar debidos a errores generados por la utilidad “bnfc” al tratar con la gramática de Clafer. Entre ellos 3 warnings, de los cuáles el primero hay que resolverlo entrando en el fichero que indica y modificando la secuencia de escape “\”. Al volver a compilar saltará otro error más en un fichero Java que tendremos que modificar añadiendo adecuadamente la secuencia de escape (“\\”) en el lugar que nos indique. Si sale algún otro error corregir igualmente.

Ya tendríamos el código listo en una carpeta denominada igual que la gramática (clafer) la cual podemos copiar en nuestro proyecto Java como un paquete nuevo “clafer” y empezar a usarla.

# Uso

CharStream codePointCharStream = CharStreams.*fromFileName*(filename);

claferLexer lexer = **new** claferLexer(codePointCharStream);

claferParser parser = **new** claferParser(**new** CommonTokenStream(lexer));

parser.addParseListener(**new** claferParserBaseListener());

##### Abstract syntax tree:

claferParser.Start\_ModuleContext pc = parser.start\_Module();

clafer.Absyn.Module ast = pc.result;

System.out.println();

System.out.println("Parse Succesful!");

System.out.println();

System.out.println("[Abstract Syntax]");

System.out.println();

System.out.println(PrettyPrinter.show(ast));

System.out.println();

System.out.println("[Linearized Tree]");

System.out.println();

System.out.println(PrettyPrinter.print(ast));

##### Recorrer el Parse tree:

ParseTree tree = parser.module();

ParseTreeWalker walker = **new** ParseTreeWalker();

claferParserListener listener = **new** PrintEverything();

walker.walk(listener, tree);