Introduction to Language Theory and Compilation
Solutions
Session 11: lex/flex scanner generator

The files for each exercises are available on the *Université Virtuelle*. You can use this generic JFlex file definition:

```
//import java_cup.runtime.*; uncommet if you use CUP

%%// Options of the scanner

%class Lexer5 //Name
%unicode      //Use unicode
%line       //Use line counter (yyline variable)
%column     //Use character counter by line (yycolumn variable)

//you can use either %cup or %standalone
//   %standalone is for a Scanner which works alone and scan a file
//   %cup is to interact with a CUP parser. In this case, you have to return
//        a Symbol object (defined in the CUP library) for each action.
//        Two constructors:
//                          1. Symbol(int id,int line, int column)
//                          2. Symbol(int id,int line, int column,Object value)
%standalone

////////
//CODE//
////////
%init{//code to execute before scanning
  System.out.println("Initialization");
%init}

%{//adding Java code (methods, inner classes, ...)
%}

%eof{//code to execute after scanning
   System.out.println("Done");
%eof}

////////////////////////////////
//Extended Regular Expressions//
////////////////////////////////

EndOfLine = "\r"?"\n"

//////////
//States//
//////////

xstate YYINITIAL,PRINT;

%%//Identification of tokens and actions

<YYINITIAL>{
   {EndOfLine} {yybegin(PRINT);}
   .           {} //by default, all non matched char are printed on output
                  //we force to not print them
}

<PRINT>{
  {EndOfLine} {yybegin(YYINITIAL);}
  .           {System.out.println(yytext());} //we print them explicitly
}
```