```
package Parser;

import java_cup.runtime.*;
import AST.*;

parser code {:
public void syntax_error(Symbol s) {
  report_error("Error de sintaxis en linea " + s.left, null);
}

public void unrecovered_syntax_error(Symbol s) throws
  java.lang.Exception {
  report_fatal_error("", null);
}
:};

terminal PUNTOCOMA, ASOP, IF, THEN, ENDIF, IGUALQUE, PROG, IN, OUT, LOCAL,
MAS, ABRELLAVE, CIERRALLAVE, COMA, PAREN, TESIS;

terminal Integer CENT;
terminal Boolean CLOG, TIPO;
terminal String IDENT;

non terminal Prog Prog;
non terminal LDecl In, Out, Local, LDecl;
non terminal Decl Decl;
non terminal LVar LVar;
non terminal Sentencia Sent, SentSimp, Body;
non terminal Asignacion Asign;
non terminal Condicional Cond;
non terminal Exp Exp;

precedence left IGUALQUE;
precedence left MAS;

start with Prog;

Prog ::= PROG IDENT:i1 In:i2 Out:o Local:l Body:b {:RESULT=new
         Progv1(i1, i2, o, l, b); :}
     |   PROG IDENT:i1 In:i2 Out:o Body:b       {:RESULT=new
         Progv2(i1,i2,o,b); :} ;

In   ::= IN LDecl:l                             {:RESULT=l; :} ;

Out  ::= OUT LDecl:l                            {:RESULT=l; :} ;

Local::= LOCAL LDecl:l                          {:RESULT=l; :} ;

LDecl::= Decl:d PUNTOCOMA                       {:RESULT=new
         LDecl2(d); :}
     |   Decl:d PUNTOCOMA LDecl:l2              {:RESULT=new
         LDecl1(d, l2); :} ;

Decl ::= TIPO:t LVar:l                          {:RESULT=new
         Decl(t.booleanValue(), l); :} ;

LVar ::= IDENT:i                                {:RESULT=new
         LVar2(i); :}
     |   IDENT:i COMA LVar:l                    {:RESULT=new
         LVar1(i,l); :} ;

Body ::= ABRELLAVE Sent:s CIERRALLAVE           {:RESULT=s; :} ;

Sent ::= SentSimp:s1 PUNTOCOMA Sent:s2          {:RESULT=new
```

```
         SentenciaCompuesta(s1, s2); :}
     |   SentSimp:s PUNTOCOMA                   {:RESULT=s; :} ;

SentSimp::= Asign:s                             {:RESULT=s; :}
        |   Cond:s                              {:RESULT=s; :} ;


Asign::= IDENT:id ASOP Exp:e           {:RESULT=new
         Asignacion(id, e); :} ;

Cond::= IF Exp:e THEN Sent:s1 ENDIF
                                                {:RESULT=new
        Condicional(e, s1); :} ;


Exp::= CLOG:c                                {:RESULT=new
          ConstanteBooleana(c.booleanValue()); :}
      |   IDENT:s                               {:RESULT=new
          Variable(s); :}
      |   Exp:e1 MAS Exp:e2            {:RESULT=new Suma(e1,
           e2); :}
      |   CENT:n                               {:RESULT=new
               ConstanteEntera(n.intValue()); :}
      |   PAREN Exp:e TESIS                   {:RESULT= e; :}
      |   Exp:e1 IGUALQUE Exp:e2            {:RESULT=new
               IgualQue(e1, e2); :} ;
```