Terminals : class, type, inherits, id, let, in, isvoid, not, new, case, of, esac, if, then, else, fi, while, loop, pool

$$
\begin{array}{rcl}
program & \rightarrow & class\_list \\
class\_list & \rightarrow & class\_def \\
 & \rightarrow & class\_def\ class\_list \\
class\_def & \rightarrow & class\ type\ \{\ feature\_list\ \}\ ; \\
 & \rightarrow & class\ type\ inherits\ type\ \{\ feature\_list\ \}\ ; \\
feature\_list & \rightarrow & feature\ feature\_list \\
 & \rightarrow & \epsilon \\
feature & \rightarrow & param\ ; \\
 & \rightarrow & value\_param\ ; \\
 & \rightarrow & id\ (\ )\ :\ type\ \{\ expression\ \}\ ; \\
 & \rightarrow & id\ (\ param\_list\ )\ :\ type\ \{\ expression\ \}\ ; \\
param\_list & \rightarrow & param \\
 & \rightarrow & param\ ,\ param\_list \\
param & \rightarrow & id\ :\ type \\
value\_param & \rightarrow & param\ \leftarrow\ expression \\
block & \rightarrow & expression\ ; \\
 & \rightarrow & expression\ ;\ block \\
let\_list & \rightarrow & param \\
 & \rightarrow & param\ ,\ let\_list \\
 & \rightarrow & value\_param \\
 & \rightarrow & value\_param\ ,\ let\_list \\
case\_list & \rightarrow & param\ \Rightarrow\ expression\ ; \\
 & \rightarrow & param\ \Rightarrow\ expression\ ;\ case\_list \\
func\_call & \rightarrow & .\ id\ (\ ) \\
 & \rightarrow & @\ type\ .\ id\ (\ ) \\
 & \rightarrow & .\ id\ (\ arg\_list\ ) \\
 & \rightarrow & @\ type\ .\ id\ (\ arg\_list\ ) \\
arg\_list & \rightarrow & expression \\
 & \rightarrow & expression\ ,\ arg\_list \\
member\_call & \rightarrow & id\ (\ arg\_list\ ) \\
 & \rightarrow & id\ (\ ) \\
expression & \rightarrow & special \\
 & \rightarrow & comparison\_expr
\end{array}
$$

$$
\begin{array}{rcl}
special & \rightarrow & arith \ \leq \ special\_arith \\
 & \rightarrow & arith \ < \ special\_arith \\
 & \rightarrow & arith \ = \ special\_arith \\
 & \rightarrow & special\_arith \\
special\_arith & \rightarrow & arith \ + \ special\_term \\
 & \rightarrow & arith \ - \ special\_term \\
 & \rightarrow & special\_term \\
special\_term & \rightarrow & term \ * \ special\_unary \\
 & \rightarrow & term \ / \ special\_unary \\
 & \rightarrow & special\_unary \\
special\_unary & \rightarrow & isvoid \ special\_unary \\
 & \rightarrow & \sim \ special\_unary \\
 & \rightarrow & final\_expr \\
final\_expr & \rightarrow & let \ let\_list \ in \ expression \\
 & \rightarrow & id \ \leftarrow \ expression \\
 & \rightarrow & not \ expression \\
comparison\_expr & \rightarrow & arith \ \leq \ arith \\
 & \rightarrow & arith \ < \ arith \\
 & \rightarrow & arith \ = \ arith \\
 & \rightarrow & arith \\
arith & \rightarrow & arith \ + \ term \\
 & \rightarrow & arith \ - \ term \\
 & \rightarrow & term \\
term & \rightarrow & term \ * \ unary \\
 & \rightarrow & term \ / \ unary \\
 & \rightarrow & unary \\
unary & \rightarrow & isvoid \ unary \\
 & \rightarrow & \sim \ unary \\
 & \rightarrow & func\_expr \\
func\_expr & \rightarrow & func\_expr \ func\_call \\
 & \rightarrow & atom
\end{array}
$$

$$
\begin{aligned}
atom &\rightarrow id \\
&\rightarrow bool \\
&\rightarrow string \\
&\rightarrow interger \\
&\rightarrow new\ type \\
&\rightarrow member\_call \\
&\rightarrow (\ expression\ ) \\
&\rightarrow \{\ block\ \} \\
&\rightarrow if\ expression\ then\ expression\ else\ expression\ fi \\
&\rightarrow while\ expression\ loop\ expression\ pool \\
&\rightarrow case\ expression\ of\ case\_list\ esac
\end{aligned}
$$