UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CIÊNCIAS DA COMPUTAÇÃO

INE5426 – Construção de Compiladores

Trabalho 2

Analisador Léxico

Alunos: Mario Baldini
Caio Cordeiro da Silva
Vicente Silveira Inácio

Objetivo

- Apresentação da gramática utilizada para gerar o lexer;
- Código gerado pela ferramenta ANTLR4;
- Exemplos de programa com e sem erros, abrangendo os principais problemas da linguagem

Gramática utilizada para gerar o lexer

```
1.  grammar AnaliseLexica;
2.
3.  program : classlist?;
4.
5.  classlist : classdecl (classlist)?;
6.
7.  classdecl : 'class' ident ('extends' ident)? classbody;
8.
9.  classbody : '{' (classlist)? (vardecl ';')* (constructdecl)* (methoddecl)* '}';
10.
11. vardecl : type ident ('[' ']')* (',' ident ('['']')*)*;
12.
13. type : 'string' | 'int' | 'char' | ident;
14.
15. constructdecl : 'constructor' methodbody;
16.
17. methoddecl : type ('['']')* ident methodbody;
18.
19. methodbody : '(' paramlist ')' statement;
20.
21. paramlist : (type ident ('['']')* (',' type ident ('['']')*)*)?;
22.
23. statement : vardecl ';'
24.             | atribstat ';'
25.             | printstat ';'
```

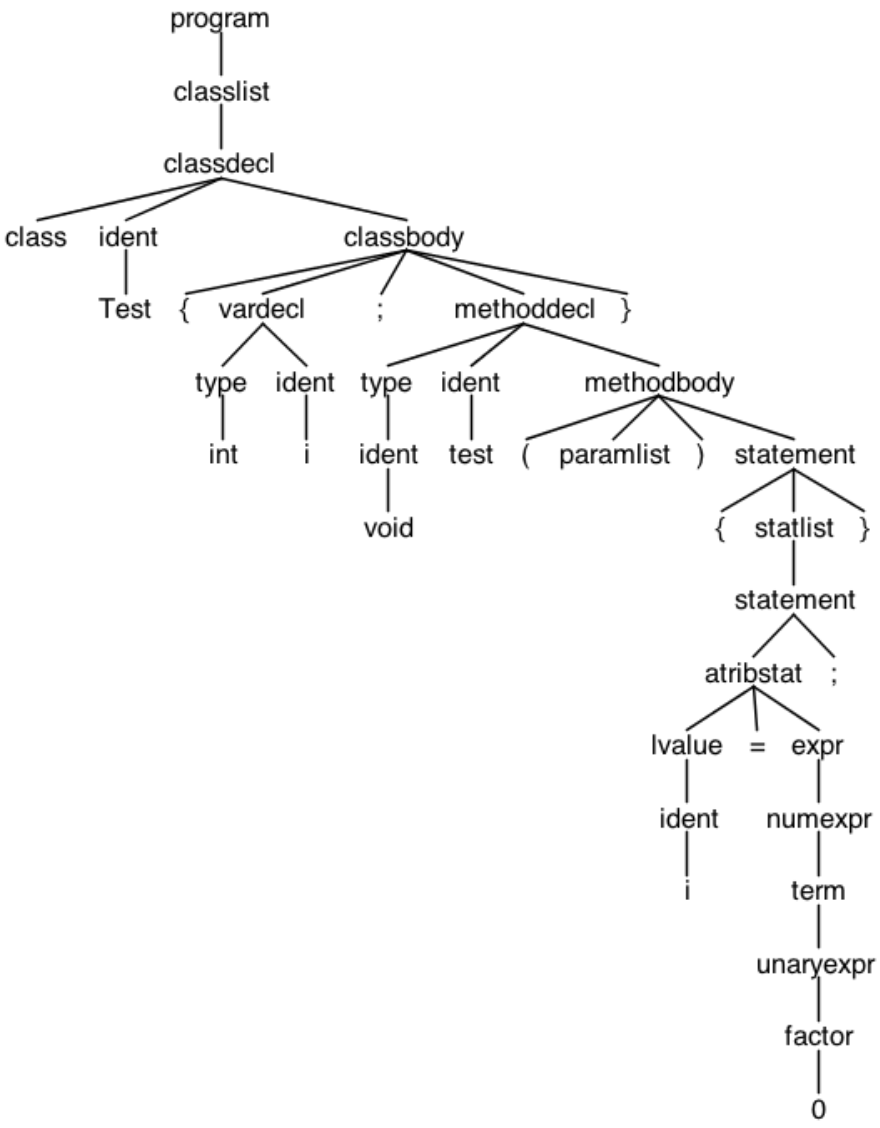
```
            | readstat ';'
            | returnstat ';'
            | superstat ';'
            | ifstat
            | whilestat
            | forstat
            | '{' statlist '}'
            | dowhilestat ';'
            | switchcasestat
            | 'break' ';'
            | ';';

expr : numexpr (( '<' | '>' | '<=' | '>=' | '==' | '!=') numexpr)?;

atribstat : lvalue '=' (expr | alocexpr);

printstat : 'print' expr;

readstat : 'read' lvalue;

returnstat : 'return' (expr)?;

superstat : 'super' '(' arglist ')';

ifstat : 'if' '(' expr ')' statement ('else' statement)?;

forstat : 'for' '(' ((atribstat? ';' expr? ';' atribstat?) | ( type 'ident' ( '[' ']' )* ':' lvalue
)) ')' statement;

whilestat : 'while' '(' expr ')' statement;

dowhilestat : 'do' statement 'while' '(' expr ')';

switchcasestat : 'switch' '(' ident ')' '{'
                ('case' expr ':' statement
                'break' ';')*
                'default' ':' statement
                '}';

lvalue : ident ('[' expr ']' | '.' ident ('(' arglist ')')?)*;

alocexpr : 'new' ( ident '(' arglist ')')
            | type ('[' expr ']')+;
```

```
68.
69. arglist : (expr (',' expr)*)?;
70.
71. numexpr : term (('+' | '-') term)*;
72.
73. term : unaryexpr (('*' | '/' | '%' | '**') unaryexpr)*;
74.
75. unaryexpr : ('+' | '-')? factor;
76.
77. factor : (INT | '\"' STRING '\"' | 'null' | lvalue | '(' expr ')');
78.
79. statlist : statement (statlist)*;
80.
81. ident : (CHAR | STRING) '_'*;
82.
83. CHAR : [a-zA-Z];
84. STRING : CHAR + INT*;
85. INT : [0-9]+;
86.
87. TI : [' ', '\t','\n', '\r', '\f'] -> skip; // ignora um espaço em branco, uma tabulação, final de
    linha...
```

Exemplos de Programas

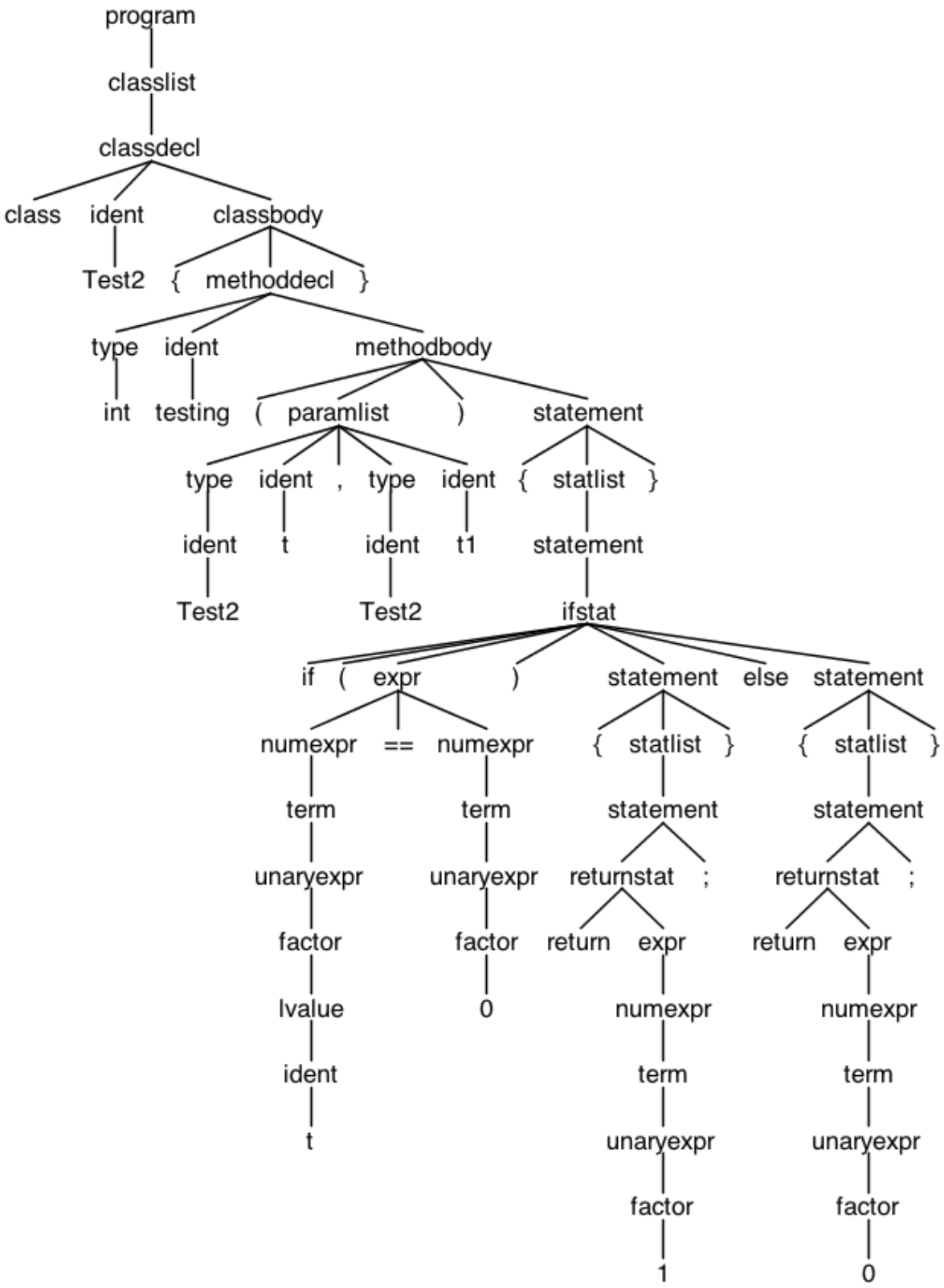
Test 1

```
1. class Test {
2.         int i;
3.
4.         void test(){
5.                 i = 0;
6.         }
7. }
```



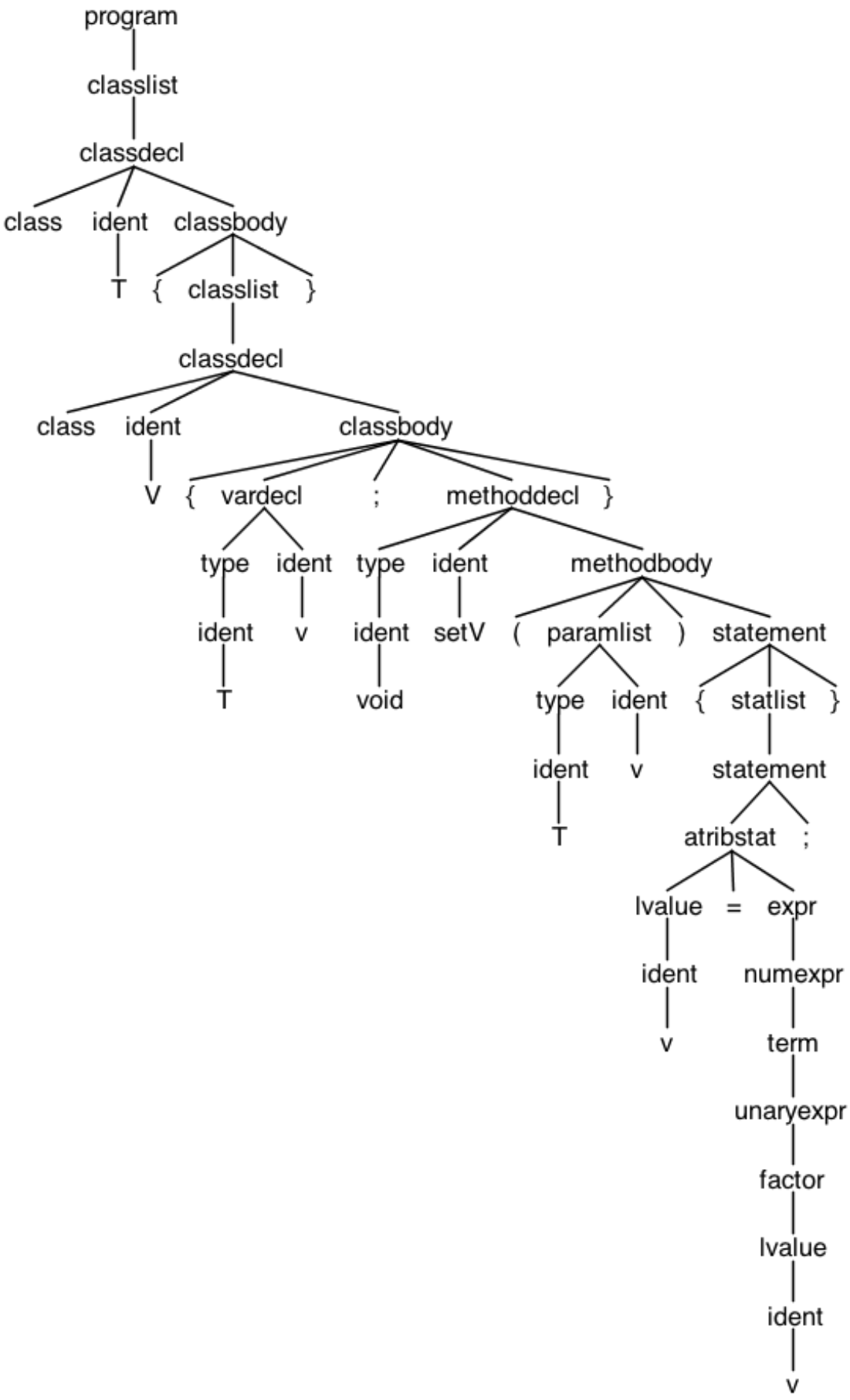
Test 2

```
1. class Test2{
2.
3.         int testing(Test2 t, Test2 t1)
4.         {
5.                 if(t == 0)
6.                 {
7.                         return 1;
8.                 }
9.                 else{
10.                                 return 0;
11.                         }
12.             }
13.     }
```


program
classlist
classdecl
class
ident
classbody
Test2
{
methoddecl
}
type
ident
methodbody
int
testing
(
paramlist
)
statement
type
ident
,
type
ident
{
statlist
}
ident
t
ident
t1
statement
Test2
Test2
ifstat
if
(
expr
)
statement
else
statement
numexpr
==
numexpr
{
statlist
}
{
statlist
}
term
term
statement
statement
unaryexpr
unaryexpr
returnstat
;
returnstat
;
factor
factor
return
expr
return
expr
lvalue
0
numexpr
numexpr
ident
term
term
t
unaryexpr
unaryexpr
factor
factor
1
0

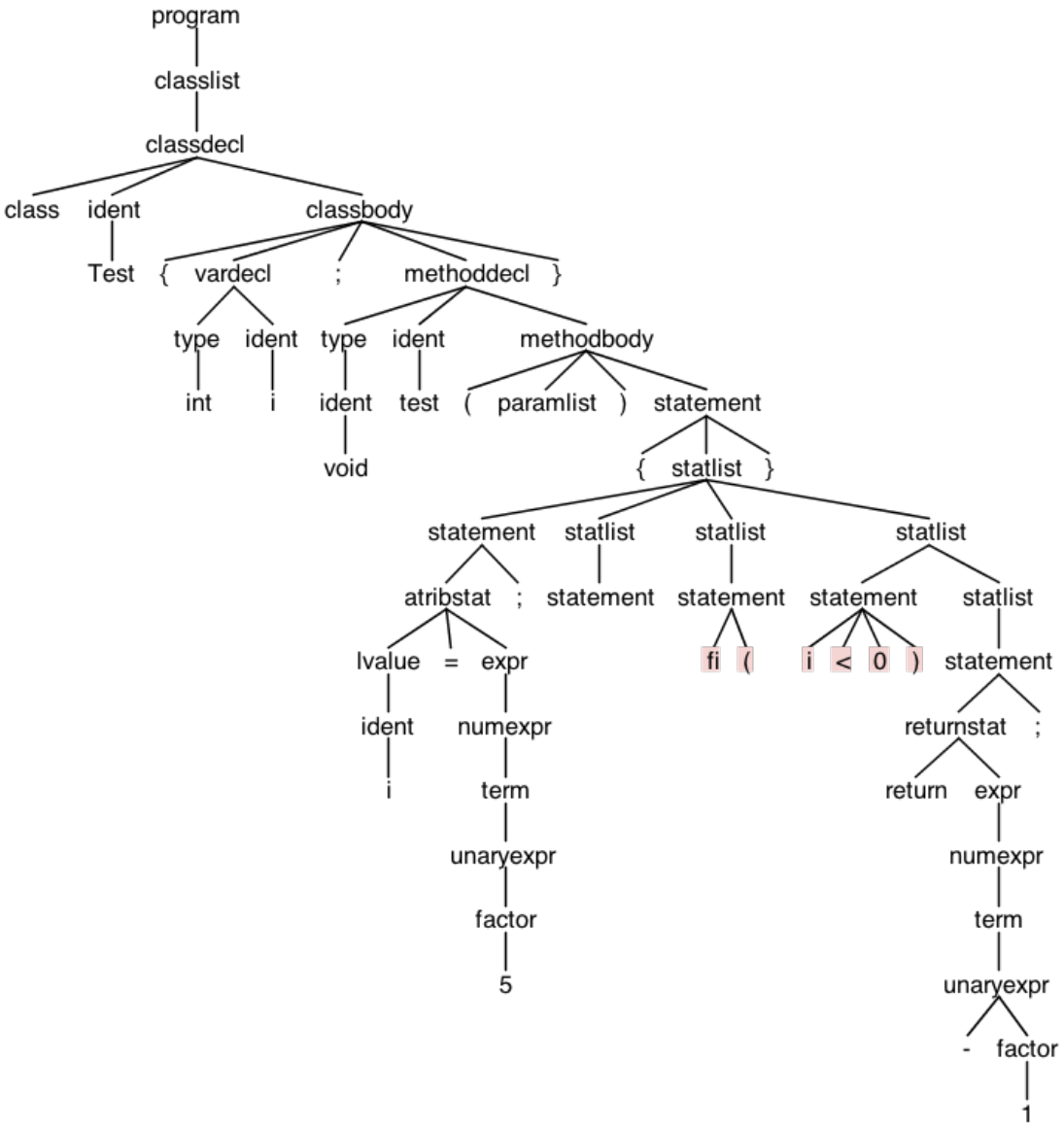
Test 3

```
1. class T {
2.
3.         class V {
4.
5.                 T v;
6.
7.                 void setV(T v)
8.                 {
9.                         v = v;
10.                    }
11.             }
12.     }
```


program
classlist
classdecl
class
ident
classbody
T
{
classlist
}
classdecl
class
ident
classbody
V
{
vardecl
;
methoddecl
}
type
ident
type
ident
methodbody
ident
v
ident
setV
(
paramlist
)
statement
T
void
type
ident
{
statlist
}
ident
v
statement
T
atribstat
;
lvalue
=
expr
ident
numexpr
v
term
unaryexpr
factor
lvalue
ident
v

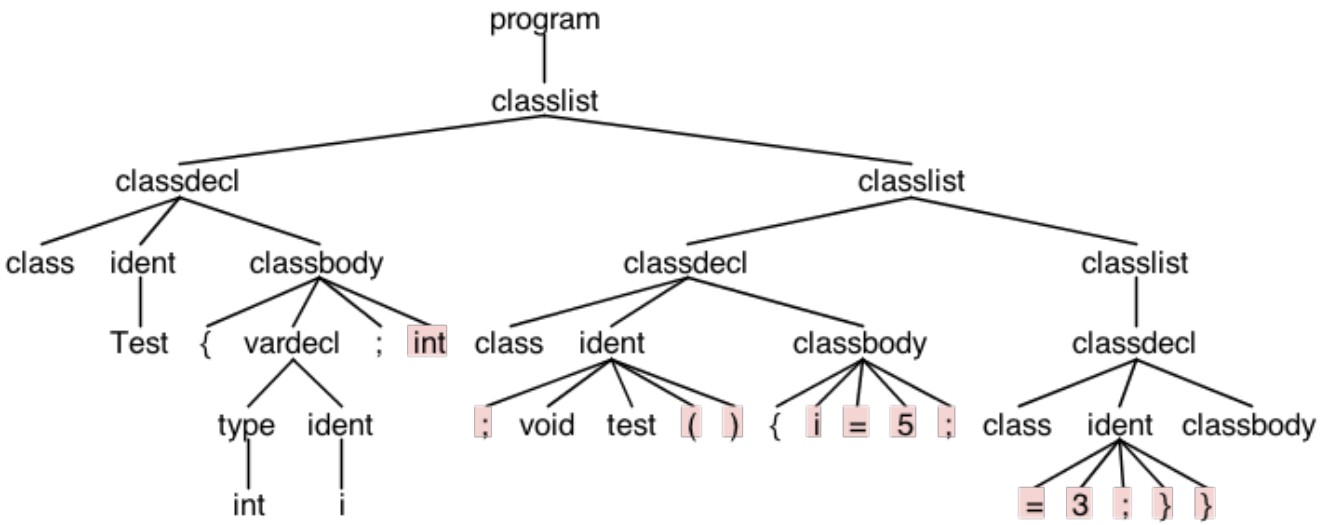
Test5_Parse_Erro-lexico-Simbolo-Invalido

```
1. class Test {
2.         int i;
3.
4.         void test(){
5.                 i = 5;
6.
7.                 fi(i < 0)
8.                         return -1;
9.
10.             }
11.
12.     }
```



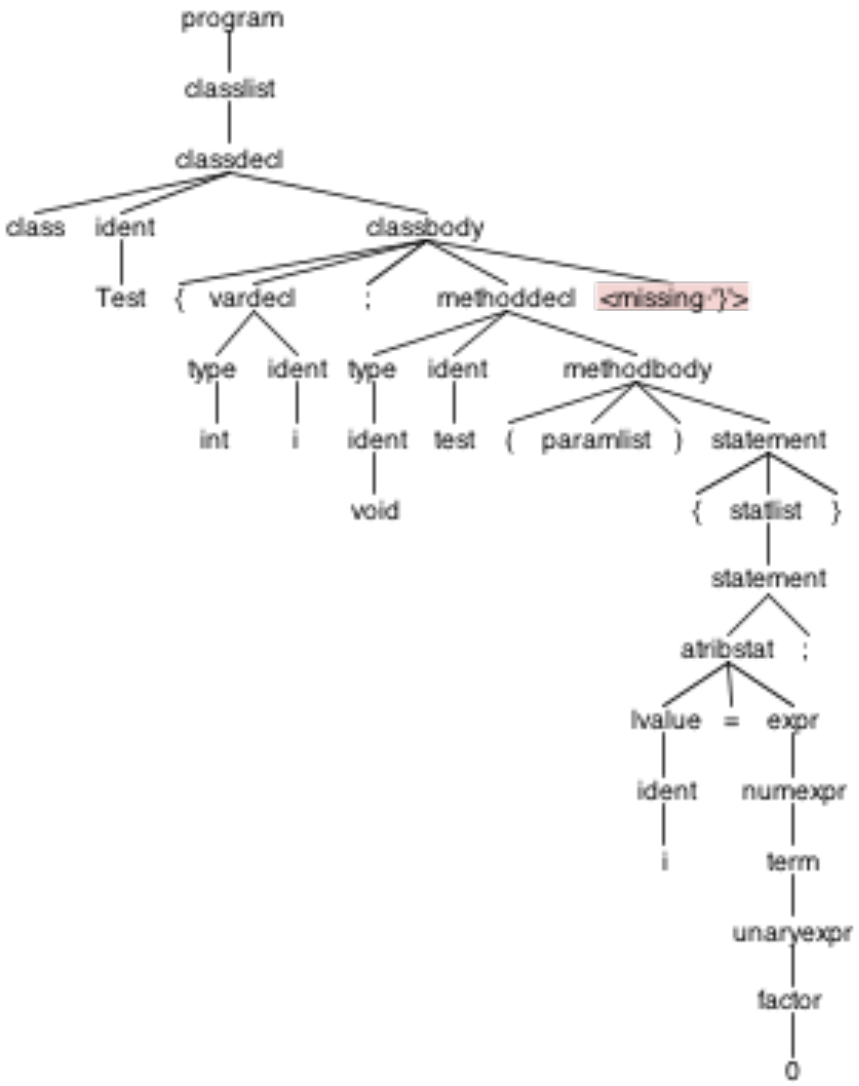
Test6_Parse_Erro-Lexico_Palavra-Reservada

```
1. class Test {
2.         int i;
3.         int class;
4.
5.         void test(){
6.
7.                 i = 5;
8.                 class = 3;
9.
10.             }
11.     }
```



Test7_Parse_Erro-Lexico_Elemento-Mal-Formado_Faltando-Chave

```
1. class Test {
2.         int i;
3.
4.         void test(){
5.                 i = 0;
6.
7. }
```



Test8_Erro-Lexico_Caracter-Invalido

```
1. class Test {
2.         int ^;
3.
4.         void test(){
5.                 i = 0;
6.         }
7. }
8.
9.
```