

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

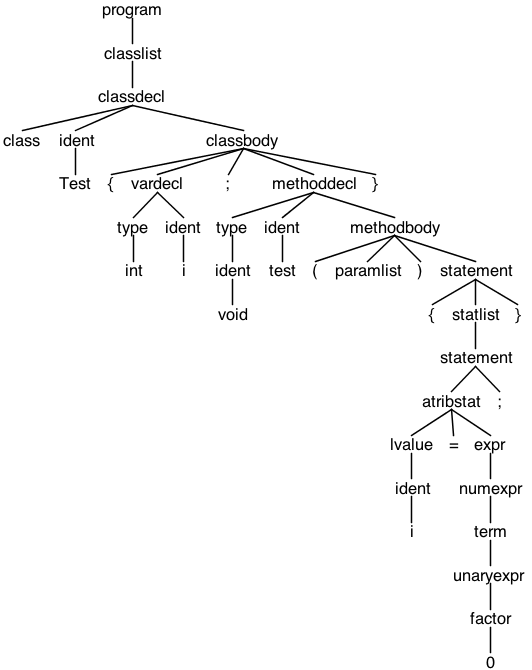
**Gramática utilizada para gerar o lexer**

1. grammar AnaliseLexica;
3. program : classlist?;
5. classlist : classdecl (classlist)?;
7. classdecl : 'class' ident ('extends' ident)? classbody;
9. classbody : '{' (classlist)? (vardecl ';')\* (constructdecl)\* (methoddecl)\* '}';
11. vardecl : type ident ('[' ']')\* (',' ident ('['']')\*)\*;
13. type : 'string' | 'int' | 'char' | ident;
15. constructdecl : 'constructor' methodbody;
17. methoddecl : type ('['']')\* ident methodbody;
19. methodbody : '(' paramlist ')' statement;
21. paramlist : (type ident ('['']')\* (',' type ident ('['']')\*)\*)?;
23. statement : vardecl ';'
24. | atribstat ';'
25. | printstat ';'
26. | readstat ';'
27. | returnstat ';'
28. | superstat ';'
29. | ifstat
30. | whilestat
31. | forstat
32. | '{' statlist '}'
33. | dowhilestat ';'
34. | switchcasestat
35. | 'break' ';'
36. | ';';
38. expr : numexpr (( '<' | '>' | '<=' | '>=' | '==' | '!=') numexpr)?;
40. atribstat : lvalue '=' (expr | alocexpr);
42. printstat : 'print' expr;
44. readstat : 'read' lvalue;
46. returnstat : 'return' (expr)?;
48. superstat : 'super' '(' arglist ')';
50. ifstat : 'if' '(' expr ')' statement ('else' statement)?;
52. forstat : 'for' '(' ((atribstat? ';' expr? ';' atribstat?) | ( type 'ident' ( '[' ']' )\* ':' lvalue )) ')' statement;
54. whilestat : 'while' '(' expr ')' statement;
56. dowhilestat : 'do' statement 'while' '(' expr ')';
58. switchcasestat : 'switch' '(' ident ')' '{'
59. ('case' expr ':' statement
60. 'break' ';')\*
61. 'default' ':' statement
62. '}';
64. lvalue : ident ('[' expr ']' | '.' ident ('(' arglist ')')?)\*;
66. alocexpr : 'new' ( ident '(' arglist ')')
67. | type ('[' expr ']')+;
69. arglist : (expr (',' expr)\*)?;
71. numexpr : term (('+' | '-') term)\*;
73. term : unaryexpr (('\*' | '/' | '%' | '\*\*') unaryexpr)\*;
75. unaryexpr : ('+' | '-')? factor;
77. factor : (INT | '\"' STRING '\"' | 'null' | lvalue | '(' expr ')');
79. statlist : statement (statlist)\*;
81. ident : (CHAR | STRING) '\_'\*;
83. CHAR : [a-zA-Z];
84. STRING : CHAR + INT\*;
85. INT : [0-9]+;
87. TI : [' ', '\t','\n', '\r', '\f'] -> skip; // ignora um espaço em branco, uma tabulação, final de linha...

**Exemplos de Programas**

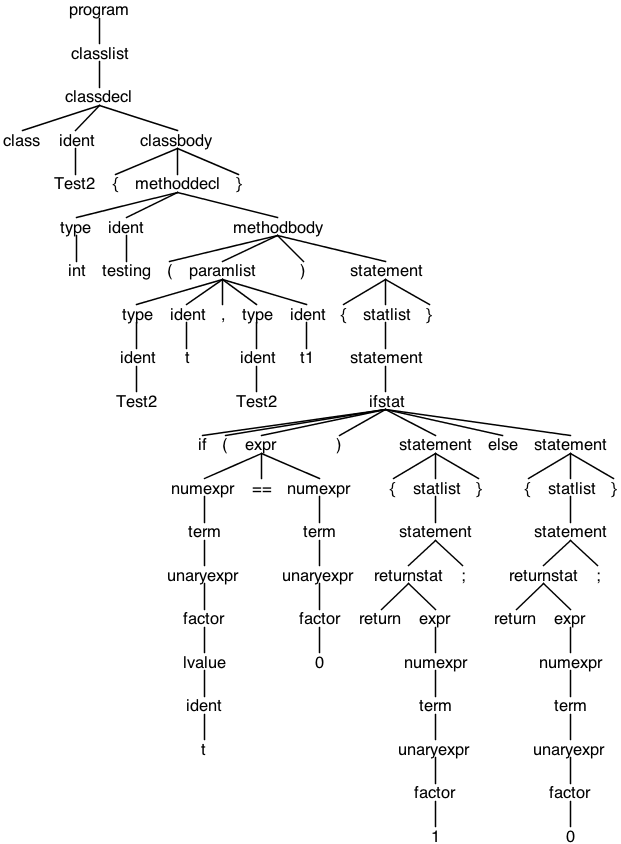
**Test 1**

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

****

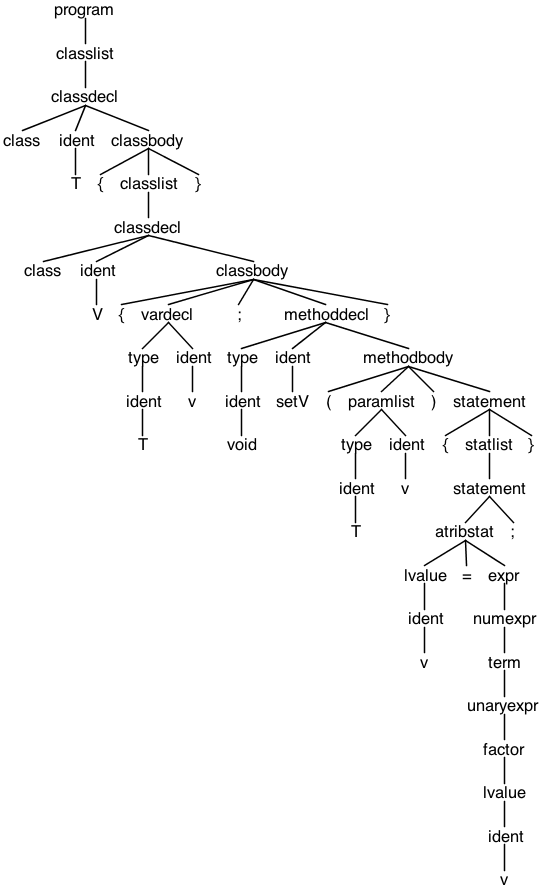
**Test 2**

1. **class** Test2{
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. }

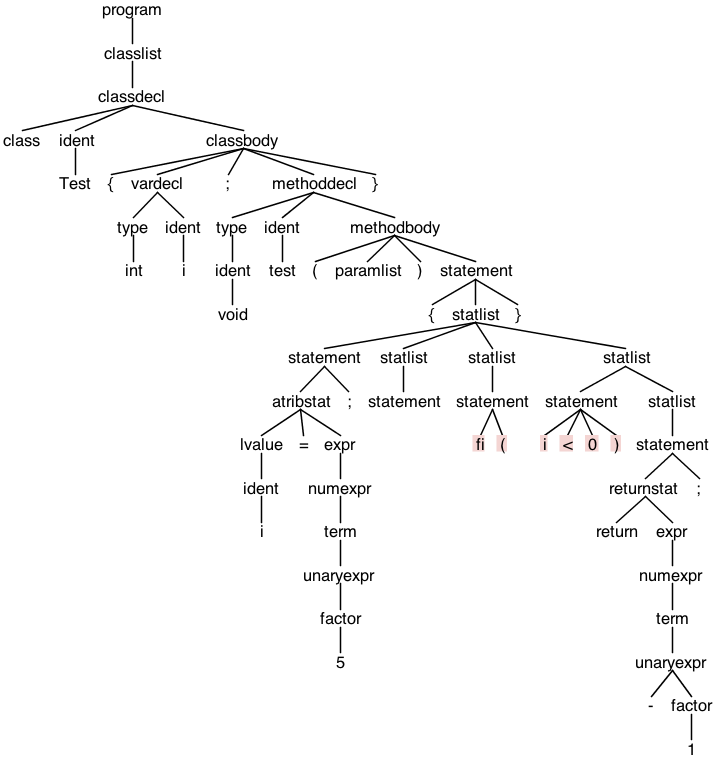


**Test 3**

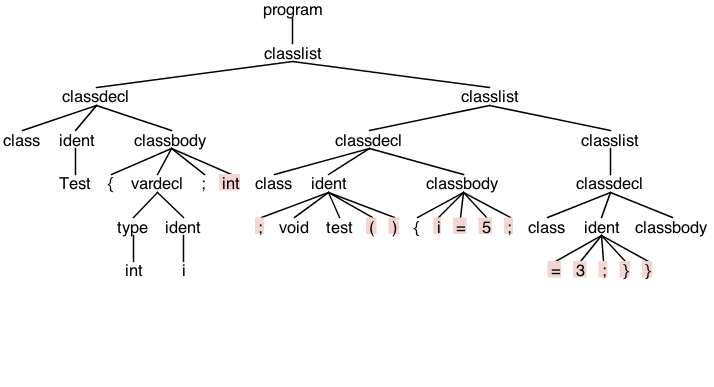
1. **class** T {
3. **class** V {
5. T v;
7. **void** setV(T v)
8. {
9. v = v;
10. }
11. }
12. }

**Test5\_Parse\_Erro-lexico-Simbolo-Invalido**

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

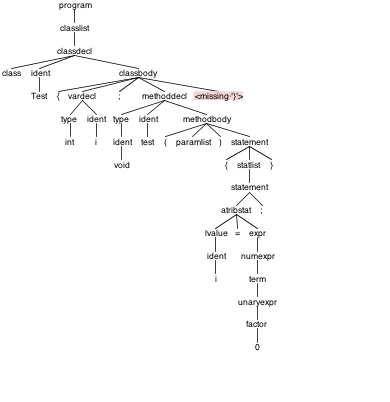
**Test6\_Parse\_Erro-Lexico\_Palavra-Reservada**

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



**Test7\_Parse\_Erro-Lexico\_Elemento-Mal-Formado\_Faltando-Chave**

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



**Test8\_Erro-Lexico\_Caracter-Invalido**

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

