
**Centro Federal de Educação Tecnológica
CEFET-MG**

**Documentação do Compilador
1ª Entrega - Analisador Léxico**

Lucas Silvestre Viana
Victor Balbo de Oliveira

Sumário

1	Implementação	3
1.1	TokenType	3
1.2	Token	3
1.2.1	Num	3
1.3	LexicalAnalyzer	4
2	Execução	4
3	Testes	4
3.1	1.test	4
3.1.1	Incorreto	4
3.1.2	Corrigido	5
3.2	2.test	6
3.2.1	Incorreto	6
3.2.2	Corrigido	6
3.3	3.test	8
3.3.1	Incorreto	8
3.3.2	Corrigido	8
3.4	4.test	10
3.4.1	Incorreto	10
3.4.2	Corrigido	10
3.5	5.test	12
3.5.1	Incorreto	13
3.5.2	Corrigido	13
3.6	6.test	14
3.6.1	Incorreto	15
3.6.2	Corrigido	15

1 Implementação

1.1 TokenType

Token	Padrão	Descrição
PROGRAM	program	Indicador de início do programa.
END	end	Encerrar um bloco.
SCAN	scan	Recebe o valor de uma variável.
PRINT	print	Imprime o valor de uma variável.
INT	int	Tipo numérico de variável.
STRING	string	Tipo textual de variável.
IF	if	Se.
THEN	then	Então.
ELSE	else	Senão.
DO	do	Faça.
WHILE	while	Enquanto.
INT_CONSTANT	digit{digit}	Constante numérica.
LITERAL	"{ASCII char}"	Constante textual.
IDENTIFIER	letter{letter digit}	Identificador de variável.
EQUALS	==	Operador lógico de igualdade.
DIFFERENT	!=	Operador lógico de diferença.
GREATER	>	Operador lógico de maior.
GREATERTHAN	>=	Operador lógico de maior ou igual.
LESS	<	Operador lógico de menor.
LESSTHAN	<=	Operador lógico de menor ou igual.
ASSIGN	=	Operador de atribuição de valor à variável.
AND	&&	Operador lógico e.
OR		Operador lógico ou.
ADDITION	+	Operador matemático de soma.
SUBTRACTION	-	Operador matemático de subtração.
MULTIPLICATION	*	Operador matemático de multiplicação.
DIVISION	/	Operador matemático de divisão.
PARENTHESES_OPEN	(Parênteses abrindo.
PARENTHESES_CLOSE)	Parênteses fechando.
SEMICOLON	;	Ponto e vírgula.
COMMA	,	Vírgula.

1.2 Token

O token é composto de um tipo de token, TokenType, e de um valor textual. Caso o token seja uma constante numérica, o seu valor é sobrescrito para o tipo inteiro.

1.2.1 Num

Token especial do tipo numérico. Herda de Token e sobrescreve o tipo do valor.

1.3 LexicalAnalyzer

Recebe o fluxo de caracteres vindos do arquivo de entrada e transforma em tokens. Caso algum caractere não seja reconhecido pela linguagem, é lançado um erro fatal, apresentando a linha com erro.

2 Execução

Primeiramente, é necessário que o Java esteja instalado. Caso não tenha o Java instalado na máquina, baixe o Java JDK e faça sua instalação. A compilação do código é feita pelo comando no terminal:

```
$ javac *.java
```

E a execução é realizada via Terminal, inserindo o comando a seguir:

```
$ java Main <arquivo_de_entrada>
```

3 Testes

3.1 1.test

```
program
    int a, b;
    int result;
    float a,x,total;
    a = 2;
    x = .1;
    scan (b);
    scan (y)
    result = (a*b ++ 1) / 2;
    print "Resultado: ";
    print (result);
    print ("Total: ");
    total = y / x;
    print ("Total: ");
    print (total);
end
```

3.1.1 Incorreto

```
(PROGRAM)
(INT) (a, IDENTIFIER) (COMMA) (b, IDENTIFIER) (SEMICOLON)
(INT) (result, IDENTIFIER) (SEMICOLON)
(float, IDENTIFIER) (a, IDENTIFIER) (COMMA) (x, IDENTIFIER) (COMMA)
    (total, IDENTIFIER) (SEMICOLON)
(a, IDENTIFIER) (ASSIGN) (2, INT_CONSTANT) (SEMICOLON)
```

```
(x, IDENTIFIER) (ASSIGN) java.lang.Exception: Invalid token on line 6.
    Undefined token started with '.'
    at LexicalAnalyzer.GetToken(LexicalAnalyzer.java:184)
    at LexicalAnalyzer.Analyze(LexicalAnalyzer.java:31)
    at Main.main(Main.java:10)
```

3.1.2 Corrigido

Removendo o . da linha 6 o teste fica correto lexicamente.

```
program
    int a, b;
    int result;
    float a,x,total;
    a = 2;
    x = 1;
    scan (b);
    scan (y)
    result = (a*b ++ 1) / 2;
    print "Resultado: ";
    print (result);
    print ("Total: ");
    total = y / x;
    print ("Total: ");
    print (total);
end
```

```
(PROGRAM)
(INT) (a, IDENTIFIER) (COMMA) (b, IDENTIFIER) (SEMICOLON)
(INT) (result, IDENTIFIER) (SEMICOLON)
(float, IDENTIFIER) (a, IDENTIFIER) (COMMA) (x, IDENTIFIER) (COMMA)
    (total, IDENTIFIER) (SEMICOLON)
(a, IDENTIFIER) (ASSIGN) (2, INT_CONSTANT) (SEMICOLON)
(x, IDENTIFIER) (ASSIGN) (1, INT_CONSTANT) (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (b, IDENTIFIER) (PARENTHESES_CLOSE)
    (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (y, IDENTIFIER) (PARENTHESES_CLOSE)
(result, IDENTIFIER) (ASSIGN) (PARENTHESES_OPEN) (a, IDENTIFIER)
    (MULTIPLICATION) (b, IDENTIFIER) (ADDITION) (ADDITION)
    (1, INT_CONSTANT) (PARENTHESES_CLOSE) (DIVISION)
    (2, INT_CONSTANT) (SEMICOLON)
(PRINT) (Resultado: , LITERAL) (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (result, IDENTIFIER) (PARENTHESES_CLOSE)
    (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (Total: , LITERAL) (PARENTHESES_CLOSE)
```

```

        (SEMICOLON)
(total, IDENTIFIER) (ASSIGN) (y, IDENTIFIER) (DIVISION)
        (x, IDENTIFIER) (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (Total: , LITERAL) (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (total, IDENTIFIER) (PARENTHESES_CLOSE)
        (SEMICOLON)
(END)

```

3.2 2.test

program

```

    int: a, c;
    float d, _e;
    a = 0; d = 3.5
    c = d / 1.2;

    Scan (a);
    Scan (c);
    b = a * a;
    c = b + a * (1 + a*c);
    print ("Resultado: ");
    print c;
    a = b + c + d)/2;
    e = val + c + a;
    print ("E: ");
    print (e);

```

3.2.1 Incorreto

```

(PROGRAM)
(INT) java.lang.Exception: Invalid token on line 2. Undefined token
      started with ':'
      at LexicalAnalyzer.GetToken(LexicalAnalyzer.java:183)
      at LexicalAnalyzer.Analyze(LexicalAnalyzer.java:31)
      at Main.main(Main.java:10)

```

3.2.2 Corrigido

Para corrigir lexicamente o teste é necessário remover o ':' presente na linha 2, o '_' do identificador e na linha 3 e os '.' das constantes numéricas das linhas 4 e 5.

program

```

    int a, c;
    float d, e;
    a = 0; d = 35
    c = d / 12;

```

```

Scan (a);
Scan (c);
b = a * a;
c = b + a * (1 + a*c);
print ("Resultado: ");
print c;
a = b + c + d)/2;
e = val + c + a;
print ("E: ");
print (e);

```

```

(PROGRAM)
(INT) (a, IDENTIFIER) (COMMA) (c, IDENTIFIER) (SEMICOLON)
(float, IDENTIFIER) (d, IDENTIFIER) (COMMA) (e, IDENTIFIER)
(SEMICOLON)
(a, IDENTIFIER) (ASSIGN) (0, INT_CONSTANT) (SEMICOLON) (d, IDENTIFIER)
(ASSIGN) (35, INT_CONSTANT)
(c, IDENTIFIER) (ASSIGN) (d, IDENTIFIER) (DIVISION) (12, INT_CONSTANT)
(SEMICOLON)
(Scan, IDENTIFIER) (PARENTHESES_OPEN) (a, IDENTIFIER)
(PARENTHESES_CLOSE) (SEMICOLON)
(Scan, IDENTIFIER) (PARENTHESES_OPEN) (c, IDENTIFIER)
(PARENTHESES_CLOSE) (SEMICOLON)
(b, IDENTIFIER) (ASSIGN) (a, IDENTIFIER) (MULTIPLICATION)
(a, IDENTIFIER) (SEMICOLON)
(c, IDENTIFIER) (ASSIGN) (b, IDENTIFIER) (ADDITION) (a, IDENTIFIER)
(MULTIPLICATION) (PARENTHESES_OPEN) (1, INT_CONSTANT)
(ADDITION) (a, IDENTIFIER) (MULTIPLICATION) (c, IDENTIFIER)
(PARENTHESES_CLOSE) (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (Resultado: , LITERAL) (PARENTHESES_CLOSE)
(SEMICOLON)
(PRINT) (c, IDENTIFIER) (SEMICOLON)
(a, IDENTIFIER) (ASSIGN) (b, IDENTIFIER) (ADDITION) (c, IDENTIFIER)
(ADDITION) (d, IDENTIFIER) (PARENTHESES_CLOSE) (DIVISION)
(2, INT_CONSTANT) (SEMICOLON)
(e, IDENTIFIER) (ASSIGN) (val, IDENTIFIER) (ADDITION) (c, IDENTIFIER)
(ADDITION) (a, IDENTIFIER) (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (E: , LITERAL) (PARENTHESES_CLOSE)
(SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (e, IDENTIFIER) (PARENTHESES_CLOSE)
(SEMICOLON)

```

3.3 3.test

```
program
  int pontuacao, pontuacaoMaxima, disponibilidade;
  string pontuacaoMinima;
  disponibilidade = "Sim";
  pontuacaoMinima = 50;
  pontuacaoMaxima = 100;

  /* Entrada de dados
  Verifica aprovação de candidatos
  do
    print("Pontuacao Candidato: ");
    scan(pontuacao);
    print("Disponibilidade Candidato: ");
    scan(disponibilidade);
    if ((pontuação > pontuacaoMinima) and (disponibilidade=="Sim") then
      print("Candidato aprovado");
    else
      print("Candidato reprovado")
    end
  while (pontuação >= 0)end
end
```

3.3.1 Incorreto

```
(PROGRAM)
(INT) (pontuacao, IDENTIFIER) (COMMA) (pontuacaoMaxima, IDENTIFIER) (COMMA) (disponibili
(STRING) (pontuacaoMinima, IDENTIFIER) (SEMICOLON)
(disponibilidade, IDENTIFIER) (ASSIGN) (Sim, LITERAL) (SEMICOLON)
(pontuacaoMinima, IDENTIFIER) (ASSIGN) (50, INT_CONSTANT) (SEMICOLON)
(pontuacaoMaxima, IDENTIFIER) (ASSIGN) (100, INT_CONSTANT) (SEMICOLON) java.lang.Excepti
    at LexicalAnalyzer.GetToken(LexicalAnalyzer.java:74)
    at LexicalAnalyzer.Analyze(LexicalAnalyzer.java:31)
    at Main.main(Main.java:10)
```

3.3.2 Corrigido

Para corrigir lexicamente o teste é necessário fechar o bloco de comentário iniciado na linha 8 e remover os caracteres especiais nos identificadores das linhas 15 e 20.

```
program
  int pontuacao, pontuacaoMaxima, disponibilidade;
  string pontuacaoMinima;
  disponibilidade = "Sim";
  pontuacaoMinima = 50;
```



```

pontuacaoMaxima = 100;

/* Entrada de dados
Verifica aprovação de candidatos */
do
    print("Pontuacao Candidato: ");
    scan(pontuacao);
    print("Disponibilidade Candidato: ");
    scan(disponibilidade);
    if ((pontuacao > pontuacaoMinima) and (disponibilidade=="Sim") then
        print("Candidato aprovado");
    else
        print("Candidato reprovado")
    end
while (pontuacao >= 0)end
end

```

```

(PROGRAM)
(INT) (pontuacao, IDENTIFIER) (COMMA) (pontuacaoMaxima, IDENTIFIER)
      (COMMA) (disponibilidade, IDENTIFIER) (SEMICOLON)
(STRING) (pontuacaoMinima, IDENTIFIER) (SEMICOLON)
(disponibilidade, IDENTIFIER) (ASSIGN) (Sim, LITERAL) (SEMICOLON)
(pontuacaoMinima, IDENTIFIER) (ASSIGN) (50, INT_CONSTANT) (SEMICOLON)
(pontuacaoMaxima, IDENTIFIER) (ASSIGN) (100, INT_CONSTANT) (SEMICOLON)
(DIVISION)
(Verifica, IDENTIFIER) (aprovação, IDENTIFIER) (de, IDENTIFIER)
      (candidatos, IDENTIFIER)
(DO)
(PRINT) (PARENTHESES_OPEN) (Pontuacao Candidato: , LITERAL)
        (PARENTHESES_CLOSE) (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (pontuacao, IDENTIFIER) (PARENTHESES_CLOSE)
        (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (Disponibilidade Candidato: , LITERAL)
        (PARENTHESES_CLOSE) (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (disponibilidade, IDENTIFIER)
        (PARENTHESES_CLOSE) (SEMICOLON)
(IF) (PARENTHESES_OPEN) (PARENTHESES_OPEN) (pontuação, IDENTIFIER)
      (GREATER) (pontuacaoMinima, IDENTIFIER) (PARENTHESES_CLOSE)
      (and, IDENTIFIER) (PARENTHESES_OPEN)
      (disponibilidade, IDENTIFIER) (EQUALS)
      (Sim, LITERAL) (PARENTHESES_CLOSE) (THEN)
(PRINT) (PARENTHESES_OPEN) (Candidato aprovado, LITERAL)
        (PARENTHESES_CLOSE) (SEMICOLON)
(ELSE)

```

```

(PRINT) (PARENTHESES_OPEN) (Candidato reprovado, LITERAL)
      (PARENTHESES_CLOSE)
(END)
(WHILE) (PARENTHESES_OPEN) (pontuação, IDENTIFIER) (GREATERTHAN)
      (0, INT_CONSTANT) (PARENTHESES_CLOSE) (END)
(END)

```

3.4 4.test

```

int: a, aux$, b;
    string nome, sobrenome, msg;
    print(Nome: );
    scan (nome);
    print("Sobrenome: ");
    scan (sobrenome);
    msg = "Ola, " + nome + " " +
    sobrenome + "!";
    msg = msg + 1;
    print (msg);
    scan (a);
    scan(b);
    if (a>b) then
        aux = b;
        b = a;
        a = aux;
    end;
    print ("Apos a troca: ");
    out(a);
    out(b)
end

```

3.4.1 Incorreto

```

(INT) java.lang.Exception: Invalid token on line 1. Undefined token
      started with ':'
      at LexicalAnalyzer.GetToken(LexicalAnalyzer.java:183)
      at LexicalAnalyzer.Analyze(LexicalAnalyzer.java:31)
      at Main.main(Main.java:10)

```

3.4.2 Corrigido

Para corrigir os erros lexicos deste teste é necessário remover o ':' e o '\$' da linha 1, e adicionar aspas ao conteúdo dentro do print da linha 3.

```

int a, aux, b;
    string nome, sobrenome, msg;

```

```

print("Nome: ");
scan (nome);
print("Sobrenome: ");
scan (sobrenome);
msg = "Ola, " + nome + " " +
sobrenome + "!";
msg = msg + 1;
print (msg);
scan (a);
scan(b);
if (a>b) then
    aux = b;
    b = a;
    a = aux;
end;
print ("Apos a troca: ");
out(a);
out(b)
end

```

```

(INT) (a, IDENTIFIER) (COMMA) (aux, IDENTIFIER) (COMMA)
      (b, IDENTIFIER) (SEMICOLON)
(STRING) (nome, IDENTIFIER) (COMMA) (sobrenome, IDENTIFIER) (COMMA)
         (msg, IDENTIFIER) (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (Nome: , LITERAL) (PARENTHESES_CLOSE)
        (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (nome, IDENTIFIER) (PARENTHESES_CLOSE)
       (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (Sobrenome: , LITERAL) (PARENTHESES_CLOSE)
        (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (sobrenome, IDENTIFIER) (PARENTHESES_CLOSE)
       (SEMICOLON)
(msg, IDENTIFIER) (ASSIGN) (Ola, , LITERAL) (ADDITION)
                  (nome, IDENTIFIER) (ADDITION) ( , LITERAL) (ADDITION)
(sobrenome, IDENTIFIER) (ADDITION) (!, LITERAL) (SEMICOLON)
(msg, IDENTIFIER) (ASSIGN) (msg, IDENTIFIER) (ADDITION)
                  (1, INT_CONSTANT) (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (msg, IDENTIFIER) (PARENTHESES_CLOSE)
        (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (a, IDENTIFIER) (PARENTHESES_CLOSE)
       (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (b, IDENTIFIER) (PARENTHESES_CLOSE)
       (SEMICOLON)
(IF) (PARENTHESES_OPEN) (a, IDENTIFIER) (GREATER) (b, IDENTIFIER)

```

```

(PARENTHESES_CLOSE) (THEN)
(aux, IDENTIFIER) (ASSIGN) (b, IDENTIFIER) (SEMICOLON)
(b, IDENTIFIER) (ASSIGN) (a, IDENTIFIER) (SEMICOLON)
(a, IDENTIFIER) (ASSIGN) (aux, IDENTIFIER) (SEMICOLON)
(END) (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (Apos a troca: , LITERAL)
(PARENTHESES_CLOSE) (SEMICOLON)
(out, IDENTIFIER) (PARENTHESES_OPEN) (a, IDENTIFIER)
(PARENTHESES_CLOSE) (SEMICOLON)
(out, IDENTIFIER) (PARENTHESES_OPEN) (b, IDENTIFIER)
(PARENTHESES_CLOSE)
(END)

```

3.5 5.test

```

program
  int a, b, c, maior, outro;

  do
    print("A");
    scan(a);
    print("B");
    scan(b);
    print("C");
    scan(c);
    //Realizacao do teste
    if ( (a>b) && (a>c) )
      maior = a

    else
      if (b>c) then
        maior = b;

        else
          maior = c;
        end
      end
    end
    print("Maior valor:");
    print (maior);
    print ("Outro? ");
    scan(outro);
    while (outro >= 0)
  end

```

3.5.1 Incorreto

```
(PROGRAM)
(INT) (a, IDENTIFIER) (COMMA) (b, IDENTIFIER) (COMMA) (c, IDENTIFIER) (COMMA) (maior, IDENTIFIER)
(DO)
(PRINT) (PARENTHESES_OPEN) (A, LITERAL) (PARENTHESES_CLOSE) (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (a, IDENTIFIER) (PARENTHESES_CLOSE) (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (B, LITERAL) (PARENTHESES_CLOSE) (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (b, IDENTIFIER) (PARENTHESES_CLOSE) (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (C, LITERAL) (PARENTHESES_CLOSE) (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (c, IDENTIFIER) (PARENTHESES_CLOSE) (SEMICOLON)
(IF) (PARENTHESES_OPEN) (PARENTHESES_OPEN) (a, IDENTIFIER) (GREATER) (b, IDENTIFIER) (PARENTHESES_CLOSE) (PARENTHESES_CLOSE)
(GREATER) (c, IDENTIFIER) (PARENTHESES_CLOSE) (PARENTHESES_CLOSE)
(maior, IDENTIFIER) (ASSIGN) (a, IDENTIFIER)
(ELSE)
(IF) (PARENTHESES_OPEN) (b, IDENTIFIER) (GREATER) (c, IDENTIFIER) (PARENTHESES_CLOSE) (PARENTHESES_CLOSE)
(maior, IDENTIFIER) (ASSIGN) (b, IDENTIFIER) (SEMICOLON)
(ELSE)
(maior, IDENTIFIER) (ASSIGN) (c, IDENTIFIER) (SEMICOLON)
(END)
(END)
(PRINT) (PARENTHESES_OPEN) (Maior valor:, LITERAL) java.lang.Exception: Invalid token on
    at LexicalAnalyzer.GetToken(LexicalAnalyzer.java:160)
    at LexicalAnalyzer.Analyze(LexicalAnalyzer.java:31)
    at Main.main(Main.java:10)
```

3.5.2 Corrigido

O único erro lexico deste teste é a aspa "aberta na linha 23 que não é fechada, por isso basta remove-la.

```
program
    int a, b, c, maior, outro;

    do
        print("A");
        scan(a);
        print("B");
        scan(b);
        print("C");
        scan(c);
        //Realizacao do teste
        if ( (a>b) && (a>c) )
            maior = a
```

```

        else
            if (b>c) then
                maior = b;

            else
                maior = c;
            end
        end
        print("Maior valor:");
        print (maior);
        print ("Outro? ");
        scan(outro);
        while (outro >= 0)
end

```

```

(PROGRAM)
(INT) (a, IDENTIFIER) (COMMA) (b, IDENTIFIER) (COMMA) (c, IDENTIFIER) (COMMA) (maior, IDENTIFIER) (DO)
(PRINT) (PARENTHESES_OPEN) (A, LITERAL) (PARENTHESES_CLOSE) (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (a, IDENTIFIER) (PARENTHESES_CLOSE) (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (B, LITERAL) (PARENTHESES_CLOSE) (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (b, IDENTIFIER) (PARENTHESES_CLOSE) (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (C, LITERAL) (PARENTHESES_CLOSE) (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (c, IDENTIFIER) (PARENTHESES_CLOSE) (SEMICOLON)
(IF) (PARENTHESES_OPEN) (PARENTHESES_OPEN) (a, IDENTIFIER) (GREATER) (b, IDENTIFIER) (PARENTHESES_CLOSE) (GREATER) (c, IDENTIFIER) (PARENTHESES_CLOSE) (PARENTHESES_CLOSE)
(maior, IDENTIFIER) (ASSIGN) (a, IDENTIFIER)
(ELSE)
(IF) (PARENTHESES_OPEN) (b, IDENTIFIER) (GREATER) (c, IDENTIFIER) (PARENTHESES_CLOSE) (PARENTHESES_CLOSE)
(maior, IDENTIFIER) (ASSIGN) (b, IDENTIFIER) (SEMICOLON)
(ELSE)
(maior, IDENTIFIER) (ASSIGN) (c, IDENTIFIER) (SEMICOLON)
(END)
(END)
(PRINT) (PARENTHESES_OPEN) (Maior valor:, LITERAL) (PARENTHESES_CLOSE) (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (maior, IDENTIFIER) (PARENTHESES_CLOSE) (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (Outro? , LITERAL) (PARENTHESES_CLOSE) (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (outro, IDENTIFIER) (PARENTHESES_CLOSE) (SEMICOLON)
(WHILE) (PARENTHESES_OPEN) (outro, IDENTIFIER) (GREATERTHAN) (0, INT_CONSTANT) (PARENTHESES_CLOSE)
(END)

```

3.6 6.test

program

```

    int a, b:
    scan(a);
    scan (b);
    int r = 2 * (a + b) / (b - a);
    print ("resultado: ");
    print(r);
end

```

3.6.1 Incorreto

(PROGRAM)

```

(INT) (a, IDENTIFIER) (COMMA) (b, IDENTIFIER) java.lang.Exception:
    Invalid token on line 2. Undefined token started with ':'
    at LexicalAnalyzer.GetToken(LexicalAnalyzer.java:184)
    at LexicalAnalyzer.Analyze(LexicalAnalyzer.java:31)
    at Main.main(Main.java:10)

```

3.6.2 Corrigido

Para corrigir lexicamente este teste basta remover o ':' ou substitui-lo por ';' na linha 2.

program

```

    int a, b;
    scan(a);
    scan (b);
    int r = 2 * (a + b) / (b - a);
    print ("resultado: ");
    print(r);
end

```

(PROGRAM)

```

(INT) (a, IDENTIFIER) (COMMA) (b, IDENTIFIER) (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (a, IDENTIFIER) (PARENTHESES_CLOSE)
    (SEMICOLON)
(SCAN) (PARENTHESES_OPEN) (b, IDENTIFIER) (PARENTHESES_CLOSE)
    (SEMICOLON)
(INT) (r, IDENTIFIER) (ASSIGN) (2, INT_CONSTANT) (MULTIPLICATION)
    (PARENTHESES_OPEN) (a, IDENTIFIER) (ADDITION) (b, IDENTIFIER)
    (PARENTHESES_CLOSE) (DIVISION) (PARENTHESES_OPEN)
    (b, IDENTIFIER) (SUBTRACTION) (a, IDENTIFIER)
    (PARENTHESES_CLOSE) (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (resultado: , LITERAL) (PARENTHESES_CLOSE)
    (SEMICOLON)
(PRINT) (PARENTHESES_OPEN) (r, IDENTIFIER) (PARENTHESES_CLOSE)
    (SEMICOLON)
(END)

```