**Relatorio do analizador Lexico**

**Nome:** Davi Nicollas de Paula Periceles **RA:**115113243

**Nome:** Pedro Augusto Marcelino Trindade Claudio **RA:**115111412

Este documento especifica com detalhes e o código do projeto de um analisador léxico para a disciplina de compiladores feito em JAVA utilizando à IDE NetBeans .

Estrutura dos arquivos

Hellow1.txt Programa valido.

Hellow2.txt Programa valido.

Hellow3.txt Programa nao valido.

LexerAluno.java​

Tag.java

Token.java

TS.java

**ArquivoJavazin.txt:** Local onde o codigo *javazin* de ser inserido

**Lexer.java​:** Arquivo principal, nele se encontra todo o processo de análise léxica,juntamente com o modo pânico.

*public class Lexer{...}* Descrição: ​Construtor da classe.

*public Lexer(String input\_data)*

Descrição: ​Este método tem como objetivo abrir o arquivo para ser analisado. Caso dê algum erro na abertura do arquivo, um erro é retornado.

*public void fechaArquivo()*

Descrição:​ Este método tem como objetivo fechar o arquivo que estava aberto para a análise.

*public void sinalizaErroLexico(String mensagem)*

Descrição: ​Este método tem como objetivo escrever no arquivo de texto os erros retornados durante a análise.

*public void retornaPonteiro()*

Descrição: ​Este método tem como objetivo voltar à posição anterio do buffer de leitura

*public Token proxToken()*

Descrição: ​Este método tem como objetivo obter o proximo token seguindo o caminho demonstrado pela AFD

*public static void main(String[] args)*

Descrição: ​Este método tem como objetivo de receber o programa fonte para ser lido é imprimir o token é à linha onde o mesmo se encontra além de imprimir à tabela de símbolos

**TS.java** Arquivo onde se encontra à tabela de simbolos.

**Token.java** Contém o modelo de Token padronizado.

*public Token(Tag nome, String lexema, int linha, int coluna)*

*nome* :nome do token que foi definido na classe tags.

*lexema*: string contendo o lexema encontrado ao analisar o arquivo.

*linha* : linha onde se encontra o token.

*coluna*: coluna onde se encontra o token.

**Tag.java:**Neste arquivo se encontra, as tags do analisador léxico que possuem a função de identificar os tokens recebidos

**Modo Pânico:** Ignora o erro porêm apresenta a mensgem ate determinada quatidade de erros definidas pelo usuarios. O modo panico tenta encontrar o fim do lexema que foi iniciado. Por exemplo se vier na linha 1 esses simbolos /\* o analisador tera que procurar ate o \*/ para considerar um comentario.

Exemplo de implementação de modo pânico:

Declaração;

private static int cont\_erro = 0;

if(cont\_erro>=5)// numero definido pelo professor;

{ System.out.println("Foram encontrados 5 erros, desta forma compilação não foi concluida");

exit(1);

}

case 35:

if (c == '\*') {

estado = 36;

} else if (lookahead == END\_OF\_FILE) {

sinalizaErroLexico("O comentário não foi fechado na linha: " + n\_line + " coluna: " + n\_column);

estado = 1;

cont\_erro++;

if (c == '\n') {

n\_line++;

}

}

|  |  |
| --- | --- |
| **Nome do lexema** | **Simbolo** |
| EOF | Fim de arquivo |
| RELOP\_LT | < |
| RELOP\_LE | <= |
| RELOP\_GT | > |
| RELOP\_GE | >= |
| RELOP\_EQ | == |
| RELOP\_ASSIGN | = |
| RELOP\_SUM | + |
| RELOP\_MINUS | - |
| RELOP\_MULT | \* |
| RELOP\_DIV | / |
| RELOP\_NE | != |
| SMB\_OP | ( |
| SMB\_CP | ) |
| SMB\_SEMICOLON | ; |
| SMB\_ABCH | { |
| SMB\_FCCH | } |
| SMB\_VIR | , |
| OP\_NAO | ! |
| OP\_NEGATIVO | - |
| OP\_OR | || |
| OP\_AND | && |
| ID | IF |
| ConstString | “davi nicollas” |
| ContInteira | 10 |
| Cont\_float | 9.5 |

**AFD:**

