```java
package ufc.frontEnd.lexing;

import java.util.ArrayList;
import java.util.List;

import ufc.common.data.Consts;
import ufc.common.data.Location;
import ufc.common.errors.TCharError;
import ufc.common.errors.TError;
import ufc.common.helpers.Util;

public class Lexer {
    private char[] Txt; // Conjunto de simbolos que formam o programa fonte (texto).
    private Location Loc; // Cuida da localizacao de erros durante a execucao de um programa
    private Character CurrentChar; // o simbolo atual na leitura do texto

    public Lexer(String text) { // Passa-se o codigo fonte inicial. Aqui chamamos de "text".
        this.Txt = text.toCharArray(); // Converte "text" para um vetor de caracteres
        this.Loc = new Location(-1, 0, -1, text); // parametros: indice -1, linha 0, coluna -1, texto
        this.CurrentChar = null; // Simbolo atual eh nulo
        this.Advance(); // chamamos o Advance() para avancar e dar o primeiro passo na analise lexica.
    }
    private void Advance() {
        // Avancamos pelo Location, passando o simbolo atual:
        this.Loc.Advance(this.CurrentChar);
        //Se ja tiver lido o ultimo simbolo de "Txt", "CurrentChar" = null. Caso contrario, CurrentChar=Txt[Idx]:
        this.CurrentChar = Loc.Idx < Txt.length ? Txt[Loc.Idx] : null;
    }
    public LexerOut MakeTokens() { // Produzir Tokens
        List<Token> tokens = new ArrayList<Token>();

        while (this.CurrentChar != null) {
            if (Util.em(this.CurrentChar, " \t")) {
                this.Advance();
            } else if (this.CurrentChar == '#') {
                this.SkipComment();
            } else if (Util.em(this.CurrentChar, ";\n\r\n")) {
                tokens.add(new Token(Consts.NEWLINE, null, this.Loc));
                this.Advance();
            } else if (Util.em(this.CurrentChar, Consts.DIGITS)) {
                tokens.add(this.MakeNumber());
            } else if (Util.em(this.CurrentChar, Consts.LETTERS)) {
                tokens.add(this.MakeIdentifier());
            } else if (this.CurrentChar == '"') {
                tokens.add(this.MakeString());
            } else if (this.CurrentChar == '+') {
                tokens.add(new Token(Consts.PLUS, null, this.Loc));
                this.Advance();
            } else if (this.CurrentChar == '-') {
                tokens.add(this.MakeMinusOrArrow());
            } else if (this.CurrentChar == '*') {
                tokens.add(new Token(Consts.MUL, null, this.Loc));
                this.Advance();
            } else if (this.CurrentChar == '/') {
                tokens.add(new Token(Consts.DIV, null, this.Loc));
                this.Advance();
            } else if (this.CurrentChar == '^') {
                tokens.add(new Token(Consts.POW, null, this.Loc));
                this.Advance();
            } else if (this.CurrentChar == '(') {
                tokens.add(new Token(Consts.LPAR, null, this.Loc));
                this.Advance();
            } else if (this.CurrentChar == ')') {
                tokens.add(new Token(Consts.RPAR, null, this.Loc));
                this.Advance();
            } else if (this.CurrentChar == '[') {
                tokens.add(new Token(Consts.LSQUARE, null, this.Loc));
                this.Advance();
            } else if (this.CurrentChar == ']') {
                tokens.add(new Token(Consts.RSQUARE, null, this.Loc));
                this.Advance();
            } else if (this.CurrentChar == '!') {
                Token token = this.MakeNotEquals();
                if (token.Error != null) return new LexerOut(new ArrayList<Token>(), token.Error);
                tokens.add(token);
            } else if (this.CurrentChar == '=') {
                tokens.add(this.MakeEquals());
            } else if (this.CurrentChar == '<') {
                tokens.add(this.MakeLessThan());
            } else if (this.CurrentChar == '>') {
                tokens.add(this.MakeGreaterThan());
            } else if (this.CurrentChar == ',') {
                tokens.add(new Token(Consts.COMMA, null, this.Loc));
                this.Advance();
            } //Inseri o 65279 seguinte para evitar erros legados adversos: Unicode Character 'ZERO WIDTH NO-BREAK SPACE'(U + FEFF)
            else if (this.CurrentChar == 65279) { this.Advance(); }
            else {
                Location locIni = this.Loc.Copy();
```

```
                Character _char = this.CurrentChar;
                this.Advance();
                TError erro = new TCharError(locIni, this.Loc, "Erro lexico: Illegal Character", "'" + _char + "'");
                return new LexerOut(new ArrayList<Token>(), erro);
            }
        }
        tokens.add(new Token(Consts.EOF, null, this.Loc, null));
        return new LexerOut(tokens, null);
    }
    private Token MakeMinusOrArrow() { // Funcao completa de Exemplo:
            String tokType = Consts.MINUS;
            Location locIni = this.Loc.Copy();
            this.Advance();

            if (this.CurrentChar == '>'){
                    this.Advance();
                    tokType = Consts.ARROW;
        }
        return new Token(tokType, null, locIni, this.Loc);
    }
    private Token MakeNotEquals() { // Funcao semi-completa de Exemplo:
            Location locIni = this.Loc.Copy();
            this.Advance();

            // Implementar sua estrategia de codigo ...
            this.Advance();
            return new Token(null, null, null, null, new TCharError(locIni, this.Loc, "Expected Character", "'=' (after '!')"));
    }
    private Token MakeEquals() { // Funcao semi-completa de Exemplo:
            String tokType = Consts.EQ;
            Location locIni = this.Loc.Copy();
            this.Advance();

            // Implementar sua estrategia de codigo

        return new Token(tokType, null, locIni, this.Loc);
    }
    private Token MakeLessThan() {// Funcao semi-completa de Exemplo:
        String tokType = Consts.LT;
        Location locIni = this.Loc.Copy();

        // Implementar sua estrategia de codigo
        return new Token(tokType, null, locIni, this.Loc);
    }
    private Token MakeGreaterThan() { // Funcao semi-completa de Exemplo:
        String tokType = Consts.GT;
        Location locIni = this.Loc.Copy();
        this.Advance();

        // Implementar sua estrategia de codigo
        return new Token(tokType, null, locIni, this.Loc);
    }
    private Token MakeIdentifier() { // Funcao semi-completa de Exemplo:
        String idStr = ""; /// pode-se usar como concatenador de simbolos, tendo como final o lexema (ou a representacao string) do
Token. Ex: "while", "numero", etc
        Location locIni = this.Loc.Copy(); // nao precisa alterar isso

        /// Implemente sua logica para classificar se eh ou nao um identificador

        String tokType = "";
        // Implementar

        return new Token(tokType, idStr, locIni, this.Loc); // nao precisa alterar isso
    }
    private Token MakeNumber() { // Funcao completa de Exemplo: para ler um numero inteiro ou real
        String numStr = "";
        int dotCount = 0;
        Location locIni = this.Loc.Copy();

        while (this.CurrentChar != null && Util.em(this.CurrentChar, Consts.DIGITS + ".")) {
            if (Util.em(this.CurrentChar, ".")) {
                if (dotCount == 1) break;
                dotCount += 1;
                numStr += ".";
            } else {
                numStr += this.CurrentChar;
            }
            this.Advance();
        }
        if (dotCount == 0) {
            return new Token(Consts.INT, numStr, locIni, this.Loc);
        } else
            return new Token(Consts.FLOAT, numStr, locIni, this.Loc);
    }
    private Token MakeString() { // Funcao semi-completa de Exemplo:
            String str = ""; // nao precisa alterar, pode-se usar para coletar a palavra ou lexema do Token
            Location locIni = this.Loc.Copy(); // nao precisa alterar
            // ?? mais variaveis para sua estrategis???
```

```
            this.Advance(); // avancar, pois estamos no simbolo ", que eh o comeco da string. Ex: "joao", comecamos por ".

            // Implementar a identificacao da string, cologanco na variavel "str"

        this.Advance(); // saimos do ultimo " da string.
            return new Token(Consts.STRING, str, locIni, this.Loc); // Para identificar erros, locIni eh o marcado inicial e Loc o
final.
    }
    private void SkipComment() { // Funcao completa de Exemplo:
        this.Advance();
        while (this.CurrentChar != '\n') // Enquanto nao pular linha
          this.Advance(); // descartar
        this.Advance(); // descartar o \n para continuar lendo novos Tokens
    }
}
```