**EXERCÍCO: MD03\_CP02\_EditorDeTexto**

aplique o padrão de projeto Observer para criar um editor de texto simples, implemente novas classes. os requisitos para avaliar o projeto são:

- Implementar uma subclasse da classe Editor chamada TextEditor;

- aplica o método insertLine, que recebe as parâmetros lineNumber e text;

- inserir o texto na ordem correspondente a lineNumber e deslocar as linhas posteriores se necessário.

- aplicar o método removeLine, que recebe lineNumber como parâmetro, deleta o texto da linha correspondente e desloca as linhas posteriores se necessário;

- instanciar um TextEditor e disparar o evento "open";

- receber as linhas de texto, que serão inseridas no objeto textEditor, do usuário até que ele envie o texto "EOF", que não deve ser inserido no objeto textEditor;

- por fim, o textEditor deve disparar o evento "save" para que o conteúdo seja salvo no arquivo configurado e imprimir todo o conteúdo do arquivo na tela.

**CLASSE MODEL:**

// editor.ts

import { Observador } from './observer';

export class Editor implements *Observador* {

  atualizar(): *void* {

      throw **new** *Error*('Method not implemented.');

  }

  notificar(): *void* {

    // Lógica para notificar observadores

  }

}

// observer.ts

export interface *Observador* {

    atualizar(): *void*;

  }

// text-editor.ts

import { VisualizadorDeTexto } from '../View/view';

import { Editor } from './editor';

export class EditorDeTexto extends Editor {

  private linhas: *string*[] = [];

  inserirLinha(*numeroLinha*: *number*, *texto*: *string*): *void* {

*this*.linhas.splice(*numeroLinha* - 1, 0, *texto*);

*this*.notificar();

  }

  removerLinha(*numeroLinha*: *number*): *void* {

*this*.linhas.splice(*numeroLinha* - 1, 1);

*this*.notificar();

  }

  obterLinhas(): *string*[] {

    return [...*this*.linhas];

  }

}

**CLASSE CONTROLLER:**

// controller.ts

import \* as readlineSync from 'readline-sync';

import { EditorDeTexto } from '../Model/text-editor';

export class ControladorDoEditorDeTexto {

  private editorDeTexto: EditorDeTexto;

  constructor(*editorDeTexto*: EditorDeTexto) {

*this*.editorDeTexto = *editorDeTexto*;

  }

  iniciar(): *void* {

    console.log('Editor de Texto - Digite suas linhas (Digite "EOF" para encerrar):');

    let numeroLinha = 1;

    while (true) {

      const entradaUsuario = readlineSync.question(`Linha ${numeroLinha}: `);

      if (entradaUsuario.toLowerCase() === 'eof') {

        break;

      }

*this*.editorDeTexto.inserirLinha(numeroLinha, entradaUsuario);

      numeroLinha++;

    }

*this*.editorDeTexto.notificar();

  }

}

**CLASSE VIEW:**

// view.ts

import { Observador } from '../Model/observer';

import { EditorDeTexto } from '../Model/text-editor';

export class VisualizadorDeTexto implements *Observador* {

  private editorDeTexto: EditorDeTexto;

  constructor(*editorDeTexto*: EditorDeTexto) {

*this*.editorDeTexto = *editorDeTexto*;

    //this.editorDeTexto.adicionarObservador(this);

  }

  atualizar(): *void* {

*this*.exibirConteudo();

  }

  private exibirConteudo(): *void* {

    console.log('Conteúdo do Arquivo:');

    const linhas = *this*.editorDeTexto.obterLinhas();

    linhas.forEach((*linha*, *indice*) => console.log(`${*indice* + 1}: ${*linha*}`));

  }

}

**APP:**

// app.ts

import { EditorDeTexto } from './Model/text-editor';

import { ControladorDoEditorDeTexto } from './Controller/controller';

import { VisualizadorDeTexto } from './View/view';

const editorDeTexto = **new** EditorDeTexto();

const controladorDoEditorDeTexto = **new** ControladorDoEditorDeTexto(editorDeTexto);

const visualizadorDeTexto = **new** VisualizadorDeTexto(editorDeTexto);

controladorDoEditorDeTexto.iniciar();