´e çã

ó é ç ã à ò à à é í ç ã ò à à ê ç ã é ç ã ç õ à à á

É é ẹ ẹ ẹ à à í ç ã É ç ã ẹ ẹ ç ã ẹ ú ẹ Í ç ã à Ẹ ç ã ẹ ò ò ç ã ò ẹ ẹ ó ç ã é à é ç õ à ç ã à Ẹ ç ã é ç ã à à ẹ á ó

á ò ẹ à

Análise de Código Java: Gerenciador de Atividades Escolares

Análise gerada por IA (Gemini)

12 de junho de 2025

1 Visão Geral do Projeto

O projeto é uma aplicação de desktop para o gerenciamento de atividades escolares, desenvolvida em Java com a biblioteca Swing. A arquitetura do código é bem estruturada e segue o padrão de projeto **Model-View-Controller (MVC)**, o que separa claramente as responsabilidades de cada parte do sistema.

- **Model**: A classe `Atividade.java`, que representa a estrutura de dados.
- **View**: O pacote `view`, contendo as classes responsáveis pela interface gráfica (`Interface`, `PainelInicio`, `PainelCRUD`).
- **Controller**: A classe `AtividadeController.java`, que contém a lógica de negócio e responde às interações do usuário.

2 Análise da Arquitetura e Arquivos

2.1 Model: Atividade.java

Esta classe define a estrutura de uma atividade. É um POJO (Plain Old Java Object) simples que armazena os dados e define como eles devem ser representados em formato de texto através do método `toString()`, que é utilizado pela `JList` na interface.

```
1  package model;
2
3  public class Atividade {
4      private String titulo;
5      private String descricao;
6      private String dataEntrega;
7
8      public Atividade(String titulo, String descricao, String dataEntrega
       ) {
9          this.titulo = titulo;
10         this.descricao = descricao;
11         this.dataEntrega = dataEntrega;
12     }
13
14     // Getters e setters...
15
```

```
16      public String toString() {
17          return "[" + dataEntrega + "]" + "[" + titulo + "]" + " - " +
        descricao;
18      }
19  }
```

Listing 1: Atividade.java

2.2 View: O Pacote `view`

As classes neste pacote são responsáveis apenas pela parte visual da aplicação.

2.2.1 Interface.java

É a janela principal (`JFrame`) que serve como contêiner para os outros painéis. Ela gerencia qual painel está visível para o usuário.

```
1  package view;
2
3  import javax.swing.*;
4
5  public class Interface extends JFrame {
6      public PainelInicio painelInicio = new PainelInicio();
7      public PainelCRUD painelCRUD = new PainelCRUD();
8
9      public Interface() {
10         setTitle("Atividades Escolares");
11         setDefaultCloseOperation(EXIT_ON_CLOSE);
12         setSize(600, 400);
13         setLocationRelativeTo(null);
14         setContentPane(painelInicio);
15         setVisible(true);
16     }
17
18     public void mostrarPainelCRUD() {
19         setContentPane(painelCRUD);
20         revalidate();
21         repaint();
22     }
23 }
```

Listing 2: Interface.java

2.2.2 PainelInicio.java e PainelCRUD.java

São os painéis (`JPanel`) que compõem as telas. `PainelInicio` é a tela de boas-vindas, e `PainelCRUD` é a tela principal onde as operações de Adicionar, Editar e Remover (CRUD) são realizadas.

```
1  package view;
2
3  import javax.swing.*;
4  import model.Atividade;
5  import java.awt.*;
6
7  public class PainelCRUD extends JPanel {
8      public JTextField campoNome = new JTextField(20);
```

```
    // ... outros componentes
    public JButton botaoAdicionar = new JButton("Adicionar");
    public JButton botaoRemover = new JButton("Remover");
    public JButton botaoEditar = new JButton("Editar");
    public DefaultListModel<Atividade> modeloLista = new
   DefaultListModel<>();
    public JList<Atividade> lista = new JList<>(modeloLista);
    private int indiceEdicao = -1; // Vari vel para controlar o modo de
     edi   o
    // ... resto da classe
}
```

Listing 3: Trecho de PainelCRUD.java

2.3 Controller: AtividadeController.java

O Controller é o "cérebro"da aplicação. Ele conecta os botões da View com as funções que manipulam os dados. Uma característica notável é a reutilização do botão "Adicionar"para também "Salvar"uma edição.

```
package controller;

import view.Interface;
import model.Atividade;
import javax.swing.JOptionPane;

public class AtividadeController {
    private final Interface tela;

    public AtividadeController(Interface tela) {
        this.tela = tela;
        // Configura todos os ActionListeners para os bot es
        tela.painelInicio.botaoIniciar.addActionListener(e -> tela.
   mostrarPainelCRUD());
        tela.painelCRUD.botaoAdicionar.addActionListener(e ->
   adicionarAtividade());
        tela.painelCRUD.botaoRemover.addActionListener(e ->
   removerSelecionada());
        tela.painelCRUD.botaoEditar.addActionListener(e ->
   editarSelecionada());
    }

    private void adicionarAtividade() {
        // ... L gica para adicionar ou salvar uma edi   o
    }

    private void removerSelecionada() {
        // ... L gica para remover item selecionado
    }

    private void editarSelecionada(){
        // ... L gica para entrar no modo de edi   o
    }
}
```

Listing 4: AtividadeController.java

2.4 Ponto de Entrada: Main.java

A classe `Main` inicia a aplicação, garantindo que a interface gráfica seja executada na thread correta (EDT - Event Dispatch Thread) para evitar problemas de concorrência, o que é uma prática recomendada para aplicações Swing.

```
1  package view;
2
3  import controller.AtividadeController;
4  import javax.swing.*;
5
6  public class Main {
7      public static void main(String[] args) {
8          try {
9              UIManager.setLookAndFeel(UIManager.
       getSystemLookAndFeelClassName());
10         } catch (Exception e) {
11             e.printStackTrace();
12         }
13         SwingUtilities.invokeLater(() -> {
14             Interface tela = new Interface();
15             new AtividadeController(tela);
16         });
17     }
18 }
```

Listing 5: Main.java