Relatório do Padrão MVC em Java

Introdução

O padrão MVC (Model-View-Controller) é amplamente utilizado para separar as responsabilidades em uma aplicação, o que facilita o desenvolvimento, a manutenção e a escalabilidade do sistema. Ele divide a aplicação em três componentes principais:

Model: Responsável pelos dados e a lógica de negócios.
View: Responsável pela apresentação e exibição dos dados para o usuário.
Controller: Intermediário que conecta o Model à View, gerenciando a lógica de entrada do usuário e as atualizações dos dados.

Neste relatório, apresentaremos um exemplo prático de implementação do padrão MVC em Java, utilizando uma aplicação de cadastro de clientes.

Descrição:

Atributos: `nome`, `endereco`, `telefone` representam os dados de um cliente.
Construtor: Inicializa os atributos.
Métodos: `getters` e `setters` permitem acessar e modificar os atributos do cliente.

View - Classe ClienteView

- A classe `ClienteView` é responsável por apresentar as informações do cliente ao usuário. Ela contém um método que exibe os detalhes do cliente, fornecidos pelo Controller.

Descrição:

- **Método**: `exibirDetalhesCliente` recebe os dados do cliente e os exibe no console.
- A View não realiza nenhuma lógica de negócios. Ela apenas mostra os dados que lhe são fornecidos.

Controller - Classe ClienteController

- A classe `ClienteController` atua como intermediária entre o Model e a View. Ela recebe as entradas do usuário, atualiza o Model e informa à View para exibir as atualizações.

Descrição:

Atributos: Contém instâncias de `Cliente` (Model) e `ClienteView` (View).

Métodos:

`setNomeCliente`, `setEnderecoCliente`, `setTelefoneCliente`: Atualizam os dados no Model.
`getNomeCliente`, `getEnderecoCliente`, `getTelefoneCliente`: Retornam os dados do Model.
`atualizarView`: Chama a View para exibir os dados atualizados do Model.

Main - Demonstração de Uso

A classe `MVCDemo` demonstra o uso do padrão MVC. Nela, um cliente é criado e atualizado utilizando o Controller, e as informações são exibidas pela View.

Descrição:

1. O cliente é inicialmente criado com o nome "Kauã".
2. A View exibe os dados do cliente.
3. O nome do cliente é alterado para "Nathany" pelo Controller, e a View é atualizada para refletir essa mudança.

Vantagens do Padrão MVC

Separação de responsabilidades: Cada camada tem uma função clara, o que facilita a manutenção e a evolução do código.
Reutilização de código: As Views podem ser alteradas sem modificar a lógica de negócios, permitindo o uso de diferentes interfaces para o mesmo sistema.
Facilidade de teste: Como o Model e o Controller são independentes da interface de usuário, eles podem ser testados isoladamente.

Conclusão

Este exemplo prático de uma aplicação simples de cadastro de clientes ilustra o uso do padrão MVC em Java. O padrão promove a organização e a separação clara entre a lógica de negócios (Model), a interface do usuário (View) e o controle das interações (Controller), resultando em um código mais modular, fácil de manter e escalável.

DIAGRAMA