UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA

Programação Orientada a Objectos II

MVC

Docente: Ruben Manhiça

Maputo, 19 de agosto de 2023

Conteúdo da Aula

1. MVC;

Introdução

Nesta aula iremos abordar um padrão de projeto muito interessante, o MVC (Model, View, Controller).

Segundo Erich Gamma, "A abordagem MVC separa Visão e Modelos pelo estabelecimento de um protocolo do tipo inserção/notificação (subscribe/notify) entre eles. Uma visão deve garantir que a sua aparência reflita o estado do modelo"

O padrão MVC coloca ordem nisso tudo, estipulando regras de separação do código de acordo com as funcionalidades, distribuindo a aplicação em camadas e fazendo com que elas sejam o mais independente possível umas das outras.

Analogia com o mundo real

Uma analogia do MVC com o mundo real poderia ser o funcionamento de um carro. No carro temos o motor que faz o processo principal, gerar força mecanica. Temos também os pedais e câmbio de marchas. Além disso, temos o painel de controle do carro que exibe informações de como está o seu funcionamento, como temperatura, pressão do óleo e medidor de rotação do motor.

- Colocando o exemplo do carro no padrão MVC temos a seguinte estrutura:

Analogia com o mundo real

- O motor do carro é certamente a camada model, pois se trata do núcleo da aplicação (carro), exercendo o maior trabalho.

Analogia com o mundo real

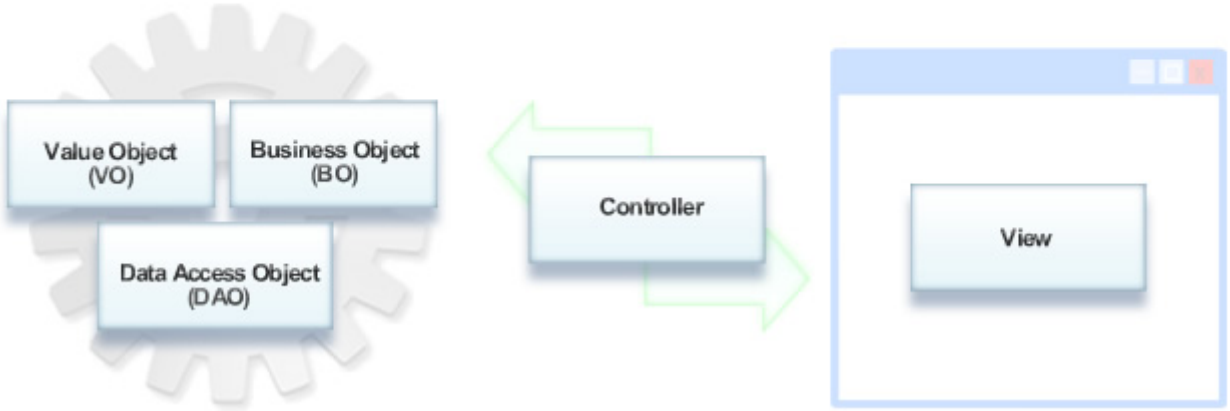
- Os pedais, as mudanças e painel fazem parte da camada view, embora o painel seja diferente dos pedais e câmbio já que é responsável em exibir dados e os outros em colher dados (interação com o usuário).

Analogia com o mundo real

- Agora imaginem, já temos o motor e os elementos de interação com o usuário mas está faltando algo. Se não houvesse nada entre esses elementos e o motor, nada aconteceria, você poderia pisar o quanto quisesse no acelerador, trocar qualquer mudança que nada aconteceria, o painel seria apenas um conjunto de ponteiros e luzes que não funcionariam para nada. É nesse ponto que entra a camada de controller. Essa camada corresponderia aos sensores de temperatura, de rotação do motor e de pressão do óleo do motor, responsaveis em fazer a interação entre a camada model e a view, fazendo com que a pressão exercida nos pedais interfira no funcionamento do motor e que o painel mostre o estado do motor.

Estrutura MVC

- A estrutura básica do MVC é a seguinte:
  - Model
    - Value Object
    - Business Object
    - Data Access Object
  - View
  - Controller



Se encontrarem outras estruturas MVC's diferentes mundo a fora, não se assustem, afinal de contas futebol, religião e MVC são difíceis de se encontrar o melhor e o pior.

Camada model

Explicado todos esses conceitos, vamos agora falar da principal camada da aplicacação, a camada model.

Como vimos anteriormente, ela é dividida em três tipos de classes:

- As Value Objects,
- Business Object e
- Data Access Object.

Sua função é prover todas as funcionalidades do software independente de interação com o usuário ou parte gráfica.

Camada model (Value Objects (VO))

Os objetos de valores (Values Objects) são classes que contém variáveis e metodos de acessos, além de construtores. Um exemplo desse tipo de classe seria assim:

Camada model (Value Objects (VO))

```java
package mz.co.exemploPOO.model;

/**
 *
 * @author Manhica
 */
public class Pessoa {

    //Variaveis

    private String cpf;
    private String nome;

    //Construtor
    public Pessoa(String cpf, String nome) {
        this.cpf = cpf;
        this.nome = nome;
    }

    //Get and Set
    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
}
```

Camada model (Data Access Object (DAO)

As classes do tipo DAO são encarregadas de fazer o acesso à dados, seja eles em um fluxo de rede, arquivo ou banco de dados. Por exemplo, métodos responsáveis em fazer acesso ao banco de dados devem estar nesse tipo de classe, assim como métodos que manipulam arquivos ou que enviam e recebem dados pela rede. Lembre-se, entrada e saída de dados!

Camada model (Data Access Object (DAO)

```java
package mz.co.exemploPOO.model;

import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.List;

/**
 *
 * @author Manhica
 */
public class ExemploDAO {
    /**
         * Metodo que recebe todos os passos (lista) e salva todos em um arquivo
         * @param passos
         * @throws FileNotFoundException
         */
        public void salvarPassos(List<Pessoa> passos) throws FileNotFoundException{
            PrintWriter pw = new PrintWriter("passos.txt");
            for (Pessoa p : passos){
                pw.print(p);
            }
            pw.flush();
            pw.close();
        }
}
```

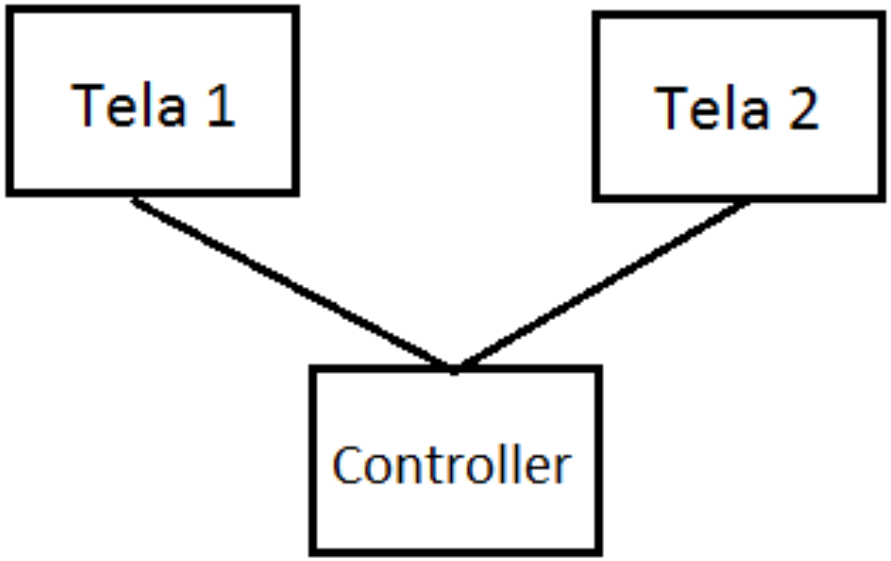
Camada model (Business Object (BO))

Esse tipo de classe também compõem a model da aplicação, assim como os dois tipos de classe ditos anteriormente. A especialidade das classes BO's é resolver operações complexas, são os processos principais da aplicação, digamos que o "miolo" do software. Nessas classes são processadas regras de negócio e tomadas de decisão.

View

- Nessa parte do projeto, iremos desenvolver a interface gráfica, mas apenas no que se refere aos componentes, nenhum tipo de comportamento ou interação com o usuário será implementado.

Controller



FIM!!!

Duvidas e Questões?