Padrões de projeto

● Definição

Um padrão oferece um esquema genérico de uma solução que pode ser aplicada em muitas situações diferentes. Ele descreve os elementos principais da solução de uma forma abstrata que pode ser especializada para o problema específico que se apresenta. Consiste em um nome que identifica o padrão, um contexto que descreve os cenários para os quais se aplica, um esquema que descreve como é aplicado e um resultado que descreve e analisa o que o padrão produz. (GOODRICH; TAMASSIA, 2013, p. 64).

Um padrão de projeto nomeia, abstrai e identifica os aspectos-chave de uma estrutura de projeto comum para torná-la útil para a criação de um projeto orientado a objetos reutilizável. O padrão de projeto identifica as classes e instâncias participantes, seus papéis, colaborações e distribuição de responsabilidades. Cada padrão de projeto focaliza um problema ou tópico particular de projeto orientado a objetos. Ele descreve em que situação pode ser aplicado, se ele pode ser aplicado em função de outras restrições de projeto e as consequências, custos e benefícios de sua utilização. (GAMMA et al*.*, 2000, p. 20).

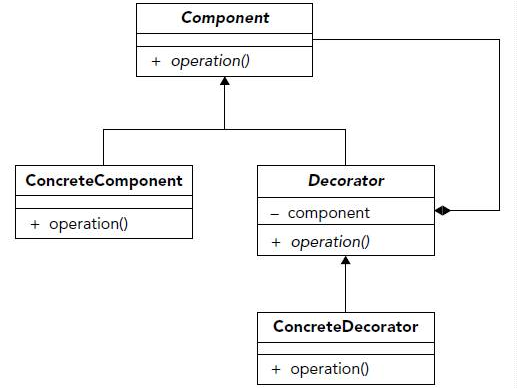
● Vantagens

“Uma das maiores vantagens do paradigma de projeto orientado a objetos é que ele facilita o desenvolvimento de software reutilizável, robusto e adaptável.” (GOODRICH; TAMASSIA, 2013).

[...] a principal vantagem do uso de padrões de projeto está no reuso das soluções propostas para determinado problema, o que permite que até mesmo profissionais menos experientes possam atuar como especialistas. Pois os padrões, geralmente, são frutos da experiência de profissionais experientes que tiveram a oportunidade de aplicar e validar tais soluções em projetos reais. Além disso, podemos destacar a facilitação da manutenção, já que um padrão representa uma unidade de conhecimento comum entre os envolvidos. (DEVMEDIA, 2022).

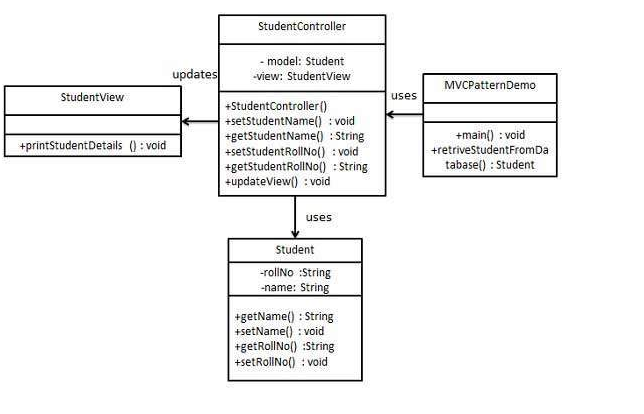
● Exemplos de padrões de projeto

Figura 1 - Diagrama de classe do Padrão de projeto Decorator



Fonte: DEVMEDIA (2022)

Figura 2 - Exemplo de Padrão de Projeto MVC



Fonte: TUTORIALSPOINT (2022)

Padrão MVC

● Definição

Apesar de muitas pessoas considerarem essa sigla como um padrão de design de interface, na verdade ele é um padrão de arquitetura de software responsável por contribuir na otimização da velocidade entre as requisições feitas pelo comando dos usuários. (BAPTISTELLA, 2022).

O MVC é um padrão de arquitetura de software que separa a sua aplicação em 3 camadas. A camada de interação do usuário View, a camada de manipulação dos dados Model e a camada de controle Controller. (ILUSTRADEV, 2022).

● Características de cada uma das camadas

Quadro 1 - Camadas e definições do padrão MVC

| Camada | Definição |
| --- | --- |
| Model ou modelo | O model é a camada que possui a lógica da aplicação. Ele é o responsável pelas regras de negócios, persistência com o banco de dados e as classes de entidades, O model recebe as requisições vindas do controller e gera respostas a partir destas requisições. |
| View ou visão | A view é a camada de visualização e representa a parte do sistema que interage com o usuário. É pela interface que haverá a entrada dos dados inseridos pelo usuário e também a saída de informações que serão exibidas para ele. Esses dados serão inseridos ou exibidos geralmente por formulários de entrada ou de saída, tabelas, grids, entre outras formas. A view não contém lógica de negócios, portanto todo o processamento é feito pela camada model e então a resposta é repassada para a view pelo controlador. |
| Controller ou controlador | Já sabemos que as requisições são enviadas pela view e a lógica de negócios é representada pelo model. Para que haja a comunicação entre essas duas camadas de maneira organizada, é necessário construir a camada controller. Sua função é ser uma camada intermediária entre a camada de apresentação (View) e a camada de negócios (Model).  Deste modo, toda requisição criada pelo usuário deve passar pelo controller, e este então se comunica com o model. Se o model gerar uma resposta para essas requisições, ele enviará as respostas ao controller que por sua vez repassa à camada view.  O controlador serve como um intermediário que organiza os eventos da interface com usuário e os direciona para a camada de modelo, assim, torna-se possível um reaproveitamento da camada de modelo em outros sistemas já que não existe dependência entre a visualização e o modelo. |

Fonte: DEVMEDIA (2022).

Quadro 2 - Camadas e definições do padrão MVC

| **Camadas** | **Definição** |
| --- | --- |
| Model ou modelo | Model ou modelo é a camada que contém a lógica da aplicação, é responsável pelas regras de negócio, para sistemas persistentes, o modelo representa a informação (dados) dos formulários e as regras SQL para manipular dados do banco, o modelo mantém o estado persistente do negócio e fornece ao controlador a capacidade de acessar as funcionalidades da aplicação, o modelo é o principal responsável toda aplicação deve representar o modelo atua isoladamente não tem conhecimento de quais serão a ou as interfaces que terá de atualizar, o modelo somente acessa á base de dados e deixa os dados prontos para o controlador este por sua vez encaminha para a visão correta. |
| View ou visão | View ou visão é a camada de apresentação com usuário, é a interface que proporcionará á entrada de dados e a visualização de respostas geradas, nas aplicações web é representado pelo HTML que é mostrado pelo browser, geralmente a visão contém formulários, tabelas, menus e botões para entrada e saída de dados. A visão deve garantir que sua apresentação reflita o estado do modelo, quando os dados do modelo mudam, o modelo notifica as vistas que dependem dele, cada vista tem a chance de atualizar-se. Desta maneira permite ligar muitas vistas a um modelo podendo fornecer diferentes apresentações, essa camada não contém códigos relacionados á lógica de negócios, ou seja, todo o processamento é feito pelo Modelo e este repassa para a visão, evidenciaremos abaixo um exemplo de duas vistas atuando sobre o mesmo modelo. |
| Controller ou controlador | Controller já descrevemos da view e do modelo, mas, temos de concordar que tudo isso se tornaria uma bagunça se tivesse alguém para organizar esta arquitetura, um controlador funciona de intermediário entre a camada de apresentação e a camada de negócios, sua função como já diz é controlar coordenar o envio de requisições feitas entre a visão e o modelo. O controller define o comportamento da aplicação, o controle é quem interpreta as solicitações (cliques, seleções de menus) feitas por usuários com bases nestes requerimentos o controlador comunica-se com o modelo que seleciona a view e atualiza-a para o usuário, ou seja, o controlador controla e mapeia as ações. |

Fonte: BAPTISTELLA (2022)

Pacotes em Java

● Definição

“Um pacote Java é um conjunto de classes relacionadas.” (HORSTMANN, 2008).

“Um conjunto de classes relacionadas, todas pertencentes ao mesmo subdiretório, pode ser um package (pacote) Java.” (GOODRICH; TAMASSIA, 2013).

● Utilidade

Um pacote Java representa um grupo de elementos Java, como classes, interfaces, enumerations e annotations. Esses elementos são agrupados para permitir controle de acesso, gerenciamento de nomes e uma melhor organização dos inúmeros programas que podemos manipular. (MENDES, 2009, p. 79).

Pacotes java são utilizados para organizar as classes da sua aplicação. Um programa pode, facilmente, ter mais de centenas de classes. Então é muito importante que todos os seus componentes fiquem organizados. Podemos pensar nos pacotes como uma pasta do seu sistema de arquivos. (HIGH5DEVS, 2022).

● Nomenclatura (convenção)

Os pacotes Java têm uma convenção de nomenclatura definida como o padrão de nomenclatura definida como o padrão de nomenclatura hierárquico. Cada nível da hierarquia é separado por pontos. Um nome de pacote inicia com o nome de domínio com nível mais alto da organização; então, os subdomínios são listados na ordem inversa. No final do nome do pacote, a empresa pode escolher como ela deseja chamá-lo. (BURTON; FELKER, 2014, p. 57).

Um pacote pode conter outros pacotes, assim como uma pasta contendo outras pastas. Por exemplo, java.util.Scanner representa que Scanner é uma classe no pacote lang e que lang é um pacote no pacote java. Por convenção, os nomes dos pacotes são todos em letras minúsculas. (BUYYA; SELVI; CHU, 2009, p. 251, tradução nossa).

“Os nomes de pacotes devem usar letras minúsculas e ser separados com sublinhados entre as palavras.” (FINEGAN; LIGUORI, 2018).

Referências

BAPTISTELLA, Adriano José. **Abordando a arquitetura MVC, e Design Patterns**: observer, composite, strategy. Observer, Composite, Strategy. Disponível em: http://www.linhadecodigo.com.br/artigo/2367/abordando-a-arquitetura-mvc-e-design-patterns-observer-composite-strategy.aspx. Acesso em: 02 jun. 2022.

BURTON, Michael; FELKER, Donn. **Desenvolvimento de aplicativos Android para leigos**. Rio de Janeiro: Alta Books, 2014.

BUYYA, Rajkumar; SELVI, S Thamarai; CHU, Xingchen. **Object-Oriented Programming with Java**: essentials and applications. New Delhi: Tata McGraw Hill, 2009.

DEVMEDIA. **A importância dos Padrões de Projeto**. Disponível em: https://www.devmedia.com.br/a-importancia-dos-padroes-de-projeto/22549. Acesso em: 07 jun. 2022.

DEVMEDIA. **Como implementar o padrão Decorator no Java EE**. Disponível em: https://www.devmedia.com.br/como-implementar-o-padrao-decorator-no-java-ee/32834. Acesso em: 07 jun. 2022.

DEVMEDIA. **Padrão MVC - Java Magazine**. Disponível em: https://www.devmedia.com.br/padrao-mvc-java-magazine/21995#2. Acesso em: 02 jun. 2022.

FINEGAN, Edward; LIGUORI, Robert. **OCA Java SE 8**: guia de estudos para o exame 1z0-808. Porto Alegre: Bookman, 2018.

GAMMA, Erich *et al*. **Padrões de projeto**: soluções reutilizáveis de software orientado a objetos. Porto Alegre: Bookman, 2000.

GOODRICH, Michael T.; TAMASSIA, Roberto. **Estruturas de Dados & Algoritmos**. 5. ed. Porto Alegre: Bookman, 2013.

HIGH5DEVS. **Pacotes Java**. Disponível em: http://high5devs.com/2015/01/pacotes-java/. Acesso em: 07 jun. 2022.

HORSTMANN, Cay. **Conceitos de computação com Java**. 5. ed. Porto Alegre: Bookman, 2008.

ILUSTRADEV. **O que é MVC? Explicação Simples**. Disponível em: https://ilustradev.com.br/o-que-e-mvc-explicacao-simples/. Acesso em: 07 jun. 2022.

MENDES, Douglas Rocha. **Programação Java com ênfase em orientação a objetos**. São Paulo: Novatec, 2009.

TUTORIALSPOINT. **Design Patterns - MVC Pattern**. Disponível em: https://www.tutorialspoint.com/design\_pattern/mvc\_pattern.htm. Acesso em: 07 jun. 2022.