**1 PADRÕES DE PROJETO**

1.1 Definição

“Um padrão descreve um conjunto composto por um contexto, um problema e uma solução. Em outras palavras, pode-se descrever um padrão como uma solução para um determinado problema em um contexto. Porém um padrão não descreve qualquer solução, mas uma solução que já tenha sido utilizada com sucesso em mais de um contexto. Exatamente por esse motivo que a descrição dos padrões normalmente sempre indica alguns de seus usos conhecidos. Um padrão não descreve soluções novas, mas soluções consolidadas.” (GUERRA, 2014).

**“Do inglês Design Patterns, podemos definir Padrões de Projeto como modelos de soluções para algum problema específico encontrado frequentemente dentro de um projeto de software. Eles servem como templates a serem aplicados para desenvolver uma solução para os problemas. Não são códigos implementados que podem ser copiados para outros softwares (na maioria das vezes), mas apenas a definição de sua aplicação. Por isso, eles funcionam nos mais variados tipos de escopos e para diversos fins.” (ANHAIA, 2018).**

“Christopher Alexander afirma: "cada padrão descreve um problema no nosso ambiente e o cerne de sua solução, de tal forma que você possa usar essa solução mais de um milhão de vezes, sem nunca fazê-lo da mesma maneira.” Muito embora Alexander, estivesse falando acerca de padrões em construções e cidades, o que ele diz é verdadeiro em relação aos padrões de projeto orientado a objeto. Nossas soluções são expressas em termos de objetos e interface em vez de um problema num determinado contexto. Padrões de projeto não são projetos, como listas encadeadas e tabelas de acesso aleatório, que podem ser codificadas em classes e ser reutilizadas tais como estão. Tampouco são projetos complexos, de domínio específico, para uma aplicação inteira ou subsistema. Padrões de projeto, neste livro, são descrições de objetos e classes comunicantes que precisam ser personalizadas para resolver um problema geral de um contexto particular. 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 reutilizáveis. O padrão de projeto identifica as classes e instâncias participantes, seus papéis, colaborações e a 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 aplicada, se ele pode ser aplicado em função e outras restrições de projeto e as consequências, custos e benefícios de sua utilização.” (GAMMA, HELM, JOHNSON, VLISSIDES, 2000).

1.2 Vantagens

“Um padrão bem escrito discute em detalhes o problema e as questões que precisam ser resolvidas e mostra como fazê-lo. Lendo um padrão, muitas questões desafiadoras e possivelmente escondidas podem ser discutidas e consideradas logo de início.” (Marinescu, 2008).

“Os design patterns não apenas ajudam a empregar os princípios básicos de projeto OOP como também são soluções já testadas por outros, ajudando a criar projetos mais simples e eficientes.” (SAMPAIO, 2007).

1.3 Exemplos De Padrões De Projeto

“Existem muitos Padrões de Projeto documentados na web, mas, sem dúvida, os desenvolvidos pela GOF são os mais conhecidos. Ao total, são 23 padrões divididos em três categorias:

1. Padrões de criação;

Responsáveis por abstrair a criação de objetos, eles transferem e centralizam responsabilidades de como e onde instanciá-los, tornando o sistema mais flexível.

2. Padrões estruturais;

Os padrões estruturais definem como classes e objetos são compostos entre si, permitindo alterações nessa composição em tempo de execução. Estão mais relacionados à estrutura como as classes e objetos estão ligados.

3. Padrões comportamentais.

Eles definem a comunicação entre classes e objetos, como também de seus comportamentos. Estão ligados diretamente à atribuição de responsabilidades desses objetos e como os algoritmos se comportam." (ANHAIA, 2018).

**2 PADRÃO MVC**

2.1 Definição

“A tríade de classes Modelo/Visão/Controlador (Model-View-Controller) (MVC) é usada para construir interfaces. Observar os padrões de projeto dentro do MVC ajudará a perceber o que queremos dizer com o termo "padrão". A abordagem MVC é composta por três tipos de objetos. O modelo é o projeto de aplicação, a visão é a apresentação na tela e o controlador é o que define a maneira como a interface do usuário tende a agrupar esses objetos. A MVC separa esses objetos para aumentar a flexibilidade e a reutilização.” (GAMMA, HELM, JOHNSON, VLISSIDES, 2000).

**“MVC é uma estratégia de separação de camadas de software que visa desacoplar a interface de seu tratamento e de seu estado.” (SAMPAIO, 2007).**

2.2 Características De Cada Uma Das Camadas

**“A camada de modelo, onde vivem as classes e entidades responsáveis por todas as regras de negócio da aplicação (classes Java convencionais, que usam e abusam orientação a objetos). a camada de visualização, responsável pela interface com o usuário (em aplicações web são os nosso arquivos JSP, HTML, CSS etc); e a camada de controlador, que faz a ligação entre a interação do usuário na camada de visualização e as regras de negócio que vivem no modelo.” (ANCHE, 2015).**

“Model (Modelo): Mantém o estado atual da View. É responsável por alterar o estado e fornecer a View os valores atuais.

View (visualização): Captura ações do usuário e comunica ao Controller para que sejam executadas. Também atualiza as informações exibidas sempre que o controller avisa que devem ser atualizadas.

Controller(controlador): Recebe solicitações da View, invoca as regras de negócio necessárias, comanda a alteração do estado do Model e a atualização das informações exibidas na view.” (SAMPAIO, 2007).

**3 PACOTES EM JAVA**

3.1 Definição

**“Um pacote (package) em Java é um diretório (pasta) em que está armazenado um conjunto de classes. Geralmente, as classes na mesma afinidade, ou mesmo um propósito, são armazenadas em um mesmo local. Alguns exemplos de pacote próprio são: awt, beans, io, lang, math, net, rmi etc.” (FURGERI, 2018).**

“Programas em Java são normalmente organizados em conjunto de pacotes. Cada pacote tem seu próprio conjunto de nomes de tipos (ou classes), ou seja, dois pacotes diferentes podem ter classes com o mesmo nome, o que previne conflitos de nomes.” (MATTOS, 2007).

3.2 Utilidade

“O uso de pacotes, é um modo de organizar grupos de classes. Um pacote contém qualquer quantidade de classes relacionadas com, finalidade, em escopo ou por herança.

Os pacotes reduzem problemas com conflitos em relação a nomes. À medida que o número de classes Java cresce, cresce também a probabilidade de você usar o mesmo nome de classe de outro desenvolvedor, aumentando a possibilidade de conflitos de nomes e mensagens ao tentar integrar grupos de classes em um único programa.” (Cadenhead, 2005).

3.3 Nomenclatura (Convenção)

**“Por convenção, o nome de um pacote sempre deve ser em letras minúsculas, (isso permite diferenciar do nome de classes), podendo ser separado por ponto final (.). É indicado também que os pacotes tenham o nome reverso do site no qual eles se encontram armazenados.” (FURGERI, 2018).**

**Referências:**

GUERRA, Eduardo. **Design Patterns com Java: Projeto orientado a objetos guiado por padrões.** Casa do Código: São Paulo, 2014.

ANHAIA, Gabriel. **Design Patterns com PHP 7: Desenvolva com as melhores soluções.** Casa do Código: São Paulo, 2018.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. **Padrões de Projeto**. São Paulo: Bookman, 2000.

MARISNECY, Floyd. **Padrões de Projeto EJB**. São Paulo: Bookman, 2008.

SAMPAIO, Cleuton. **Guia do Java: Enterprise Edition 5: desenvolvendo aplicações corporativas**. Rio de Janeiro: Brasport, 2007.

ANCHE, Mauricio. **Orientação a Objetos e SOLID para Ninjas**. São Paulo: Casa do Código, 2015.

FURGERI, Sérgio. **Java Ensino Didático: desenvolvimento e implementação de aplicações**. São Paulo: Erica, 2018.

**MATTOS, Erico Casella Tavares de. Programação de sofwares em Java. São Paulo: Digerati Books, 2007.**

CADENHEAD, Rogers. **Aprenda em 21 dias Java 2**. Rio de Janeiro: Elsevier, 2005.