UNIVERSIDADE ESTÁCIO DE SÁ
ANÁLISE E DESENVOLVIMENTO DE SISTEMA
UNIDADE CAMPO GRANDE

PADRÕES DE PROJETOS DE SOFTWARE COM JAVA

Caroline da Silva Longo - 202208616712
Maria Bruna Oliveira da Silva Gonçalves - 202202883409

Rio de Janeiro - RJ
Novembro / 2023

Padrões Gof Comportamentais

Chain of Responsibility: Permite que diferentes objetos lidem com pedidos em uma ordem específica, evitando que um objeto precise saber sobre todos os outros.

Command: Encapsula pedidos como objetos, permitindo que sejam executados e desfeitos de forma flexível. É útil para criar sistemas que podem "desfazer" ações.

Iterator: Facilita percorrer elementos em uma coleção sem saber como ela é organizada internamente. Torna a iteração mais independente da estrutura da coleção.

Mediator: Ajuda a reduzir o acoplamento entre objetos, agindo como um intermediário para suas interações, especialmente útil em sistemas complexos.

Memento: Permite salvar e restaurar o estado de um objeto, útil para criar funções de "desfazer" em aplicativos.

Strategy: Permite escolher entre diferentes formas de fazer algo, encapsulando algoritmos em objetos intercambiáveis.

Observer: Mantém objetos atualizados quando algo muda em outro objeto. É útil para criar sistemas que reagem a eventos.

Visitor: Permite adicionar novas operações a objetos sem alterá-los diretamente, especialmente útil para interpretar linguagens.

State: Permite que um objeto mude seu comportamento quando seu estado interno muda, ajudando a evitar condicionais complicados.

Interpreter: É usado para interpretar e processar linguagens específicas, tornando-as compreensíveis e executáveis, especialmente útil para sistemas que lidam com linguagens personalizadas.

Template Method: Define um esqueleto de algoritmo em uma classe base, onde partes específicas são implementadas nas subclasses, promovendo a reutilização de código.

Cada padrão resolve problemas específicos de design de software e torna o código mais flexível e fácil de manter.

Padrões GRASP

Os padrões GRASP, que significam "General Responsibility Assignment Software Patterns" (Padrões Gerais de Atribuição de Responsabilidade de Software), são um conjunto de diretrizes e princípios de design de software que ajudam os desenvolvedores a criar sistemas orientados a objetos mais flexíveis, coesos e de fácil manutenção. Esses padrões têm como objetivo principal a distribuição eficiente de responsabilidades entre classes e objetos em um sistema de software. Alguns dos padrões GRASP mais conhecidos incluem:

Coesão alta: Significa que um componente de software deve ter uma única responsabilidade bem definida, o que torna o código mais organizado e fácil de manter.

Controlador: Este padrão define um controlador que atua como um intermediário entre a interface do usuário e o sistema, gerenciando as interações e a lógica de negócios.

Especialista na Informação: Esse padrão sugere que a responsabilidade deve ser atribuída à classe que possui as informações necessárias para cumprir essa responsabilidade.

Criador: O padrão Creator orienta a responsabilidade de criar instâncias de uma classe para uma classe relacionada, evitando acoplamento excessivo.

Acoplamento Baixo: Significa reduzir a dependência entre componentes de software, permitindo que eles interajam de forma independente e flexível, facilitando a manutenção e a reutilização do código.

Invenção Pura: Quando nenhuma classe natural se encaixa para assumir uma responsabilidade, você pode criar uma classe puramente para cumprir essa responsabilidade.

Indireção: Esse padrão introduz uma camada intermediária para desacoplar classes que possam estar diretamente acopladas de outra forma.

Polimorfismo: Promove a utilização de polimorfismo para realizar ações com base no tipo do objeto, em vez de condicionais que verificam tipos.

Variações Protegidas: Não é um padrão de design de software amplamente reconhecido. Pode ser um termo específico usado em um contexto particular. Mais informações são necessárias para uma descrição adequada.

Esses padrões GRASP fornecem orientações valiosas para o design de software orientado a objetos, promovendo práticas que levam a sistemas mais flexíveis, extensíveis e de fácil manutenção. Eles são uma parte importante da engenharia de software e ajudam a melhorar a qualidade e a estrutura dos sistemas de software.

Padrões de Delegação de Responsabilidade: Distribuem tarefas entre objetos para melhor organização e eficiência.

Padrões de Herança: Estabelecem relações de herança entre classes para compartilhar funcionalidades comuns.

Padrões de Injeção de Dependência: Fornecem dependências externamente às classes, promovendo flexibilidade e reutilização.

Padrões de Fábrica: Permitem a criação de objetos de forma flexível e desacoplada, com base em critérios específicos.