Os 23 Padrões de Projeto do Gang of Four e seus melhores Use Cases

Os 23 padrões de projeto do Gang of Four (GoF) são soluções reutilizáveis para problemas comuns de design de software. Eles são divididos em três categorias: criação, estruturais e comportamentais. Cada padrão tem um propósito específico e um conjunto de contextos onde ele é mais apropriado.

Padrões de Criação

Factory Method

Define uma interface para criar um objeto, mas deixa as subclasses decidirem qual classe instanciar.
Use case: Quando você precisa de uma família de objetos relacionados e não sabe de antemão qual será necessário.

Abstract Factory

Fornece uma interface para criar famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.
Use case: Quando você precisa criar famílias de objetos relacionados, como interfaces de usuário para diferentes plataformas.

Builder

Separa a construção de um objeto complexo de sua representação, permitindo que o mesmo processo de construção crie diferentes representações.
Use case: Quando a construção de um objeto é complexa e envolve muitos passos.

Prototype

Especifica os tipos de objetos a serem criados usando uma instância protótipo e cria novos objetos copiando este protótipo.
Use case: Quando a criação de um novo objeto é cara e você quer reduzir o tempo de criação.

Singleton

Garante que uma classe tenha apenas uma instância e fornece um ponto de acesso global a ela.
Use case: Quando você precisa de exatamente um objeto de uma determinada classe, como um gerenciador de logs ou um objeto de configuração.

Padrões Estruturais

Adapter

Converte a interface de uma classe para outra interface que os clientes esperam.
Use case: Quando você precisa usar uma classe existente que não possui a interface correta.

Bridge

Separa uma abstração de sua implementação, de modo que as duas possam variar independentemente.
Use case: Quando você precisa desacoplar uma abstração de sua implementação.

Composite

Compõe objetos em estruturas de árvore para representar hierarquias parte-todo.
Use case: Quando você precisa representar hierarquias de objetos, como um sistema de arquivos ou uma estrutura de menus.

Decorator

Adiciona responsabilidades a um objeto dinamicamente.
Use case: Quando você precisa adicionar funcionalidades a um objeto sem alterar sua classe.>

Facade

Fornece uma interface unificada para um conjunto de interfaces em um subsistema.
Use case: Quando você precisa simplificar a interface de um subsistema.

Flyweight

Usa compartilhamento para suportar eficientemente um grande número de objetos de granularidade fina.
Use case: Quando você precisa representar um grande número de objetos similares.

Proxy

Fornece um substituto para outro objeto para controlar o acesso a ele.
Use case: Quando você precisa controlar o acesso a um objeto, como um proxy de rede ou um proxy virtual.

Padrões Comportamentais

Chain of Responsibility

Permite que várias classes manipulem uma solicitação, encadeando os objetos e passando a solicitação ao longo da cadeia até que um objeto a trate.
Use case: Quando você precisa implementar diferentes comportamentos para uma mesma solicitação.

Command

Encapsula uma solicitação como um objeto, permitindo que você parametrize clientes com diferentes solicitações, coloque solicitações em fila ou registre o histórico de solicitações.
Use case: Quando você precisa implementar operações undo/redo ou quando você precisa parametrizar um objeto com uma ação a ser realizada.

Interpreter

Dada uma gramática, constrói uma árvore de representação para uma sentença nessa gramática e define uma interpretação para essa árvore.
Use case: Quando você precisa implementar uma linguagem simples.

Iterator

Fornece uma maneira de acessar os elementos de um objeto agregado sequencialmente sem expor sua representação interna.
Use case: Quando você precisa percorrer os elementos de uma coleção de objetos.

Mediator

Define um objeto que encapsula como um conjunto de objetos interage. O mediador promove o acoplamento fraco, centralizando toda a comunicação.
Use case: Quando você precisa reduzir o acoplamento entre muitos objetos.

Memento

Captura e externaliza o estado interno de um objeto, permitindo que o objeto seja restaurado a esse estado posteriormente.
Use case: Quando você precisa implementar a funcionalidade undo/redo.

Observer

Define uma dependência um-para-muitos entre objetos, de modo que quando um objeto muda de estado, todos os seus dependentes são notificados e atualizados automaticamente.
Use case: Quando você precisa implementar um mecanismo de notificação.

State

Permite que um objeto altere seu comportamento quando seu estado interno muda.
Use case: Quando o comportamento de um objeto depende de seu estado interno.

Strategy

Define uma família de algoritmos, encapsula cada um deles e os torna intercambiáveis.
Use case: Quando você precisa escolher entre diferentes algoritmos em tempo de execução.

Template method

Define o esqueleto de um algoritmo em um método, diferindo alguns passos para as subclasses.
Use case: Quando você precisa implementar a estrutura de um algoritmo, mas permitir que as subclasses forneçam a implementação de alguns passos.

Visitor

Representa uma operação a ser executada sobre os elementos de uma estrutura de objeto. Permite que você defina novas operações sem mudar as classes dos elementos sobre os quais operam.
Use case: Quando você precisa adicionar novas operações a uma estrutura de objetos existente.

Lembre-se que os padrões de projeto são ferramentas poderosas, mas não são uma solução mágica para todos os problemas. Utilize-os com sabedoria e adapte-os às suas necessidades específicas.

José Cruz - 2024