Padrões de projeto (design patterns) são soluções típicas para problemas comuns em projeto de software. Cada padrão é como uma planta de construção que você

pode customizar para resolver um problema de projeto particular em seu código. São um conjunto de ferramentas para soluções de problemas comuns em design

de software.

Escolha: FACADE PATTERN

O Facade é um padrão de projeto estrutural que fornece uma interface simplificada para uma biblioteca, um framework, ou qualquer conjunto complexo de classes.

Imagine que você precisa fazer seu código funcionar com um amplo conjunto de objetos que pertencem a uma sofisticada biblioteca ou framework. Normalmente,

você precisaria inicializar todos aqueles objetos, rastrear as dependências, executar métodos na ordem correta, e assim por diante. Como resultado, a lógica

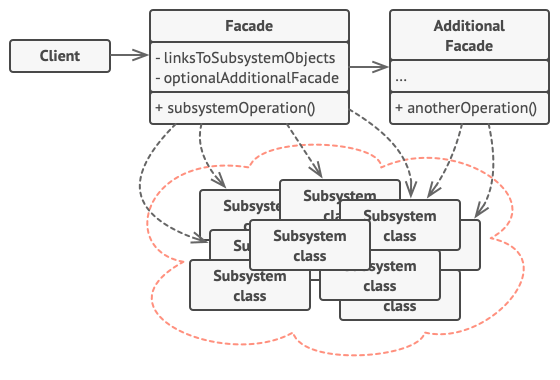
de negócio de suas classes vai ficar firmemente acoplada aos detalhes de implementação das classes de terceiros, tornando difícil compreendê-lo e mantê-lo.

Uma fachada é uma classe que fornece uma interface simples para um subsistema complexo que contém muitas partes que se movem. Uma fachada pode fornecer funcionalidades

limitadas em comparação com trabalhar com os subsistemas diretamente. Contudo, ela inclui apenas aquelas funcionalidades que o cliente se importa.

Exemplo: Quando você liga para uma loja para fazer um pedido, um operador é sua fachada para todos os serviços e departamentos da loja. O operador fornece a você uma

simples interface de voz para o sistema de pedido, pagamentos, e vários sistemas de entrega.



Ao invés de fazer seu código funcionar com dúzias de classes framework diretamente, você cria a classe fachada que encapsula aquela funcionalidade e a esconde do resto do código. Essa estrutura também ajuda você a minimizar o esforço usando para atualizar para futuras versões do framework ou substituí-lo por outro. A única coisa que você precisaria mudar em sua aplicação seria a implementação dos métodos da fachada.

Vantagens:

* Você pode isolar seu código da complexidade de um subsistema.

Desvantagens:

* Uma fachada pode se tornar [**um objeto deus**](https://refactoring.guru/pt-br/antipatterns/god-object) acoplado a todas as classes de uma aplicação.

Referências:

**Mergulho nos PADRÕES de PROJETO**. Alexander Shvets, 2021, disponível em: https://refactoring.guru/design-patterns/book