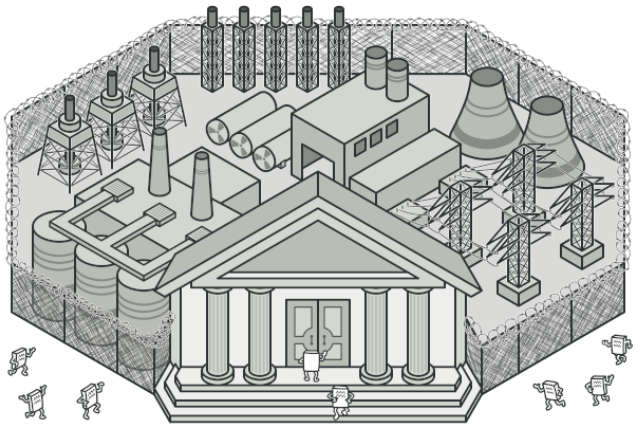
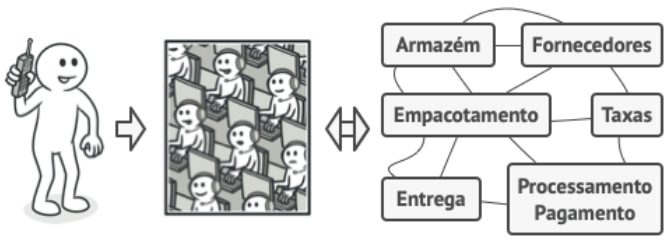
2025-2 / Lorenzon : OO - Design Patterns - Facade

Design Patterns – Facade

O **Facade** é mais um padrão de projeto estrutural do catálogo GoF. Ele fornece **uma interface unificada e simplificada** para um conjunto de interfaces em um subsistema complexo. Em outras palavras, o Facade funciona como uma **fachada de um prédio**: o usuário interage com uma porta de entrada única, sem precisar conhecer todos os detalhes internos da construção.



O padrão **Facade** serve para **deixar as coisas mais fáceis**. Imagine que dentro de um sistema existem muitas classes e funções diferentes que precisam ser usadas em uma ordem certa. Para o usuário, isso pode ser confuso e trabalhoso. O Facade resolve esse problema criando **uma única porta de entrada**, ou seja, um método simples que chama tudo o que precisa lá dentro, sem que o usuário saiba dos detalhes. É como apertar um botão de “assistir filme” no controle remoto: por trás, ele liga o som, a TV, a internet e prepara tudo, mas você só usou um comando. Assim, o código fica mais limpo, organizado e fácil de entender.



Quando usar

- Quando há **sistemas complexos** com muitas classes, bibliotecas ou APIs que o cliente não deve manipular diretamente;
- Para **isolar** partes do sistema, oferecendo um **ponto de acesso único** e reduzindo o acoplamento entre subsistemas;
- Em integrações de **sistemas legados**, quando não se quer expor a complexidade para novas camadas;
- Para **organizar dependências** e **simplificar código cliente** que teria que lidar com múltiplos objetos.

Benefícios

- **Simplicidade**: reduz a complexidade para o cliente;
- **Baixo acoplamento**: o cliente não depende diretamente de classes internas;
- **Organização**: melhora a estrutura e leitura do código;
- **Flexibilidade**: facilita evolução do sistema sem impactar o cliente.

Exemplo de implementação

A estratégia de implementação e um exemplo de código:

1. Verifique se é possível providenciar uma interface mais simples que a que o subsistema já fornece. Você está no caminho certo se essa interface faz o código cliente independente de muitas classes do subsistema;
2. Declare e implemente essa interface em uma nova classe fachada. A fachada deve redirecionar as chamadas do código cliente para os objetos apropriados do subsistema. A fachada deve ser responsável por inicializar o subsistema e gerenciar seu ciclo de vida a menos que o código cliente já faça isso;
3. Para obter o benefício pleno do padrão, faça todo o código cliente se comunicar com o subsistema apenas através da fachada. Agora o código cliente fica protegido de qualquer mudança no código do subsistema. Por exemplo, quando um subsistema recebe um upgrade para uma nova versão, você só precisa modificar o código na fachada;
4. Se a fachada ficar grande demais, considere extrair parte de seu comportamento para uma nova e refinada classe fachada.

```
/*public*/ class SistemaAudio {
    public void ligarAudio() {
        System.out.println("Áudio ligado.");
    }
}
```

```
/*public*/ class SistemaVideo {
    public void ligarVideo() {
        System.out.println("Vídeo ligado.");
    }
}
```

```
/*public*/ class SistemaRede {
    public void conectar() {
        System.out.println("Rede conectada.");
    }
}
```

```
/*public*/ class HomeTheaterFacade {
    private SistemaAudio audio;
    private SistemaVideo video;
    private SistemaRede rede;

    public HomeTheaterFacade() {
        this.audio = new SistemaAudio();
        this.video = new SistemaVideo();
        this.rede = new SistemaRede();
    }

    public void assistirFilme() {
        System.out.println("Preparando para assistir ao filme...");
        audio.ligarAudio();
        video.ligarVideo();
        rede.conectar();
        System.out.println("Filme pronto para iniciar!");
    }
}
```

```
public class FacadeApp {
    public static void main(String[] args) {
        // Cliente acessa apenas a fachada
        HomeTheaterFacade homeTheater = new HomeTheaterFacade();
        homeTheater.assistirFilme();
    }
}
```

Considerações

O **Facade não adiciona novas funcionalidades**; ele **organiza e simplifica o acesso** às já existentes.

Não elimina a possibilidade de acessar diretamente as classes internas, mas orienta o uso pelo ponto de entrada único.

É bastante usado em **APIs**, **frameworks** e **integrações** para esconder detalhes complexos do cliente.