UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO CENTRO MULTIDISCIPLINAR DE PAU DOS FERROS

Avaliação Unidade II

Hildi Dante Oliveira de Queiroz
Roberto Fernandes Rocha
Renato Alves Pessoa de Medeiros
Ruan Pabllo Barbosa Claudino
Gustavo Linhares Batista

Pau dos Ferros – RN

2024

01. GitHub

02. GitHub

03.

O código não implementa o Factory Method porque o método createXMLReader() é estático e já decide por conta própria qual objeto concreto criar com base numa propriedade do sistema.

No Factory Method, quem deve decidir a criação do objeto são as subclasses da fábrica, e não a própria fábrica base de forma fixa. Ou seja, o código atual está centralizando a lógica de criação dentro da própria fábrica e não deixa isso flexível para outras classes decidirem.

No Factory Method de verdade, você teria uma classe abstrata ou interface para a fábrica, e subclasses dessa fábrica é que definiriam qual implementação específica do XMLReader será criada.

obs:código no GitHub

04. GitHub

05.

a) O padrão Facade é como uma "fachada" que esconde a complexidade de um sistema. Ele cria uma interface simples para você usar, sem que você precise se preocupar com todos os detalhes internos e complicados.

Um exemplo de padrão Facade em uma situação real, seria a de uma recepção em um hotel. Quando você chega, pode ir direto à recepção, onde há uma pessoa que pode ajudá-lo com tudo que você precisa: fazer check-in, dar informações sobre a cidade, pedir um táxi, etc. Essa recepção simplifica sua experiência e esconde a complexidade que existe atrás dela, como os diferentes departamentos do hotel (limpeza, manutenção, reservas, etc.).

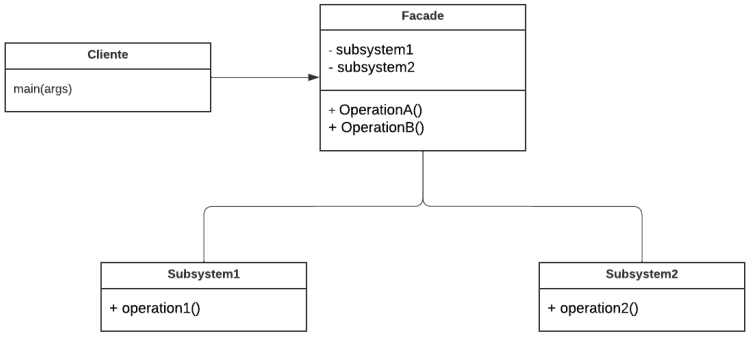
b) Quando o sistema é complicado: Se você está lidando com um sistema com muitas partes diferentes e interações complexas, o Facade pode ajudar a torná-lo mais fácil de usar.

Quando tem várias partes: Se você tem um sistema dividido em várias partes ou serviços, o Facade pode unificá-los em uma única interface.

Quando se tem sistemas antigos: Se você está trabalhando com um software mais antigo que é complicado e difícil de entender, um Facade pode criar uma nova interface mais amigável.

Quando estiver juntando serviços: Se você está tentando juntar várias APIs ou serviços diferentes, um Facade pode ajudar a simplificar como você interage com eles.

c)



06. GitHub

07.

A solução estrutural Adapter, permite que duas interfaces incompatíveis funcionem juntas. Ele serve como um intermediário, traduzindo a interface de um cliente para a que está esperando, Para situações em que a interoperabilidade entre objetos é necessária, mas as interfaces não são

compatíveis e não podem ser alteradas, o padrão Adapter é o melhor. Ele facilita a colaboração entre diferentes componentes do software, aumentando a flexibilidade e a integração em sistemas complexos.

situações importantes em que o uso do padrão Adapter é mais benéfico:

- Quando precisa fazer dois objetos incompatíveis funcionarem juntos e suas interfaces não podem ser alteradas, o adaptador estabelece uma ponte alterando uma interface para que ela se ajuste à outra.
- O adaptador modifica a interface do código antigo para ser compatível com as novas implementações, facilitando a transição e reutilização em projetos em que o código legado deve ser integrado a um novo sistema.
- O Adapter resolve essa incompatibilidade ao usar bibliotecas externas cujo design de interface não se encaixa com a arquitetura atual do seu sistema. Isso permite que a biblioteca funcione sem a necessidade de fazer modificações diretas.
- A interface do objeto pode ser adaptada para funcionar em um novo contexto com a ajuda do padrão Adapter

08.

O padrão Composite simplifica o código ao permitir que objetos simples e objetos compostos sejam tratados da mesma forma, sem precisar ficar diferenciando entre eles. Isso é possível porque todos os tipos de objetos implementam a mesma interface, então você pode chamar os mesmos métodos em folhas e compostos, sem se preocupar com o que exatamente eles são. Com isso, o código não precisa de checagens como "isso é uma folha ou um composto?", já que os próprios objetos sabem como agir. Isso deixa o sistema mais enxuto, eliminando condicionais desnecessárias e tornando o código mais fácil de entender e manter.

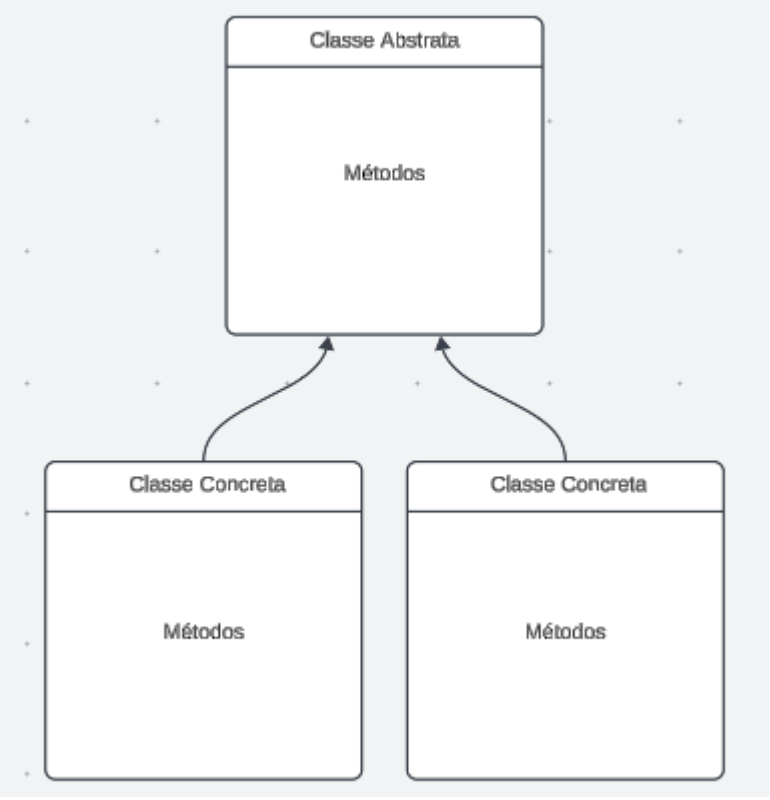
Um exemplo prático é com arquivos e pastas. Arquivos são objetos simples (folhas), enquanto pastas podem conter outros arquivos ou até subpastas (objetos compostos). você pode tratar tanto arquivos quanto pastas

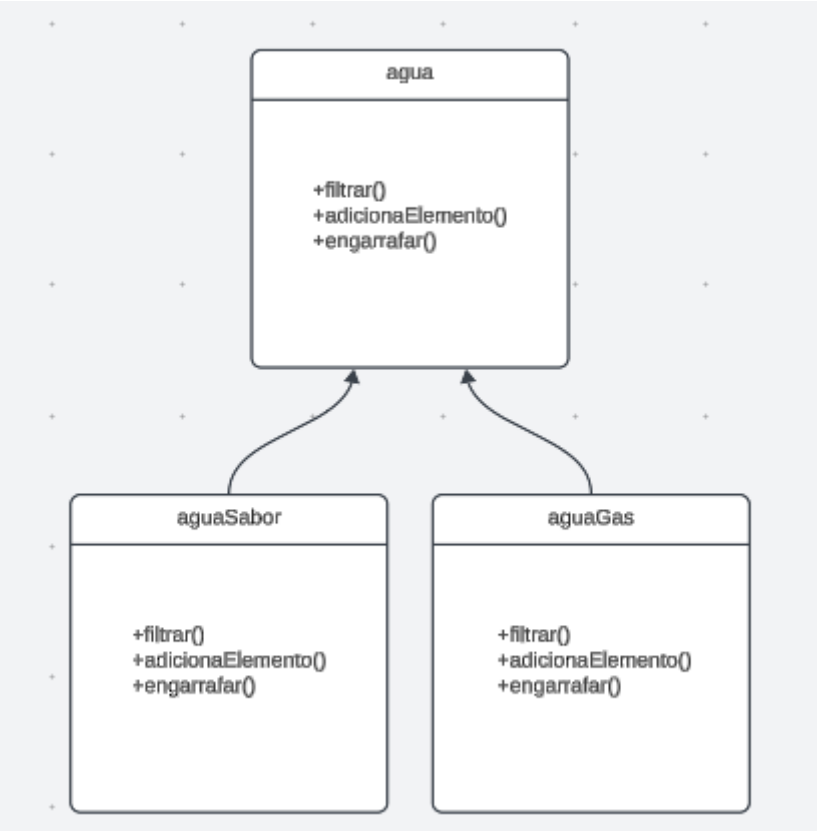
da mesma forma, sem precisar se preocupar com o que é o quê.

09.

Problema: Existe uma empresa que tem como seu principal produto água, porém existem dois tipos de água que a empresa trabalha individualmente. A primeira é a água saborizada, que passa pelo processo de filtragem, é adicionada à essência, e após isso, engarrafada. A segunda é a água com gás, onde o processo é parecido, mas uma adição diferente, no lugar da essência é adicionado o gás.

Solução: Os dois produtos compartilham de processos muito semelhantes, então a proposta é construir uma estrutura geral que compartilhe as características semelhantes dos dois produtos. E isso é possível utilizando o Template.




agua
+filtrar()
+adicionaElemento()
+engarrafar()
aguaSabor
+filtrar()
+adicionaElemento()
+engarrafar()
aguaGas
+filtrar()
+adicionaElemento()
+engarrafar()