Aula 12 - Design Pattern Adapter

Disciplina: Arquitetura de Software

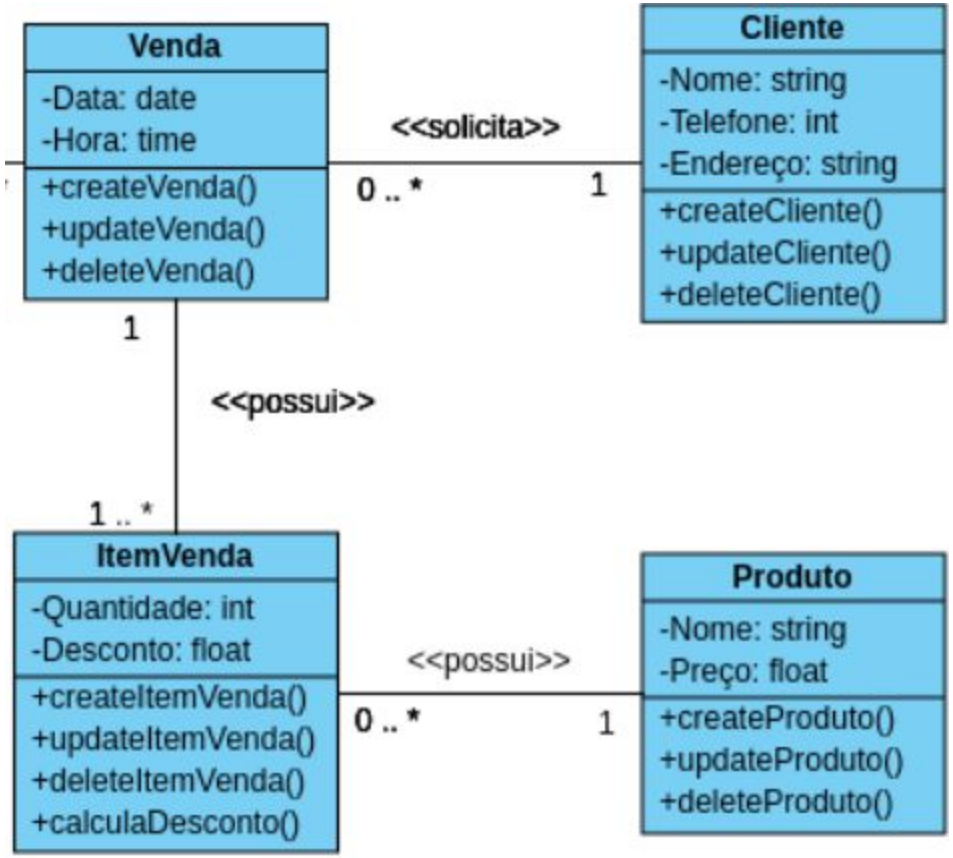
Prof. Me. João Paulo Biazotto

UniCesumar

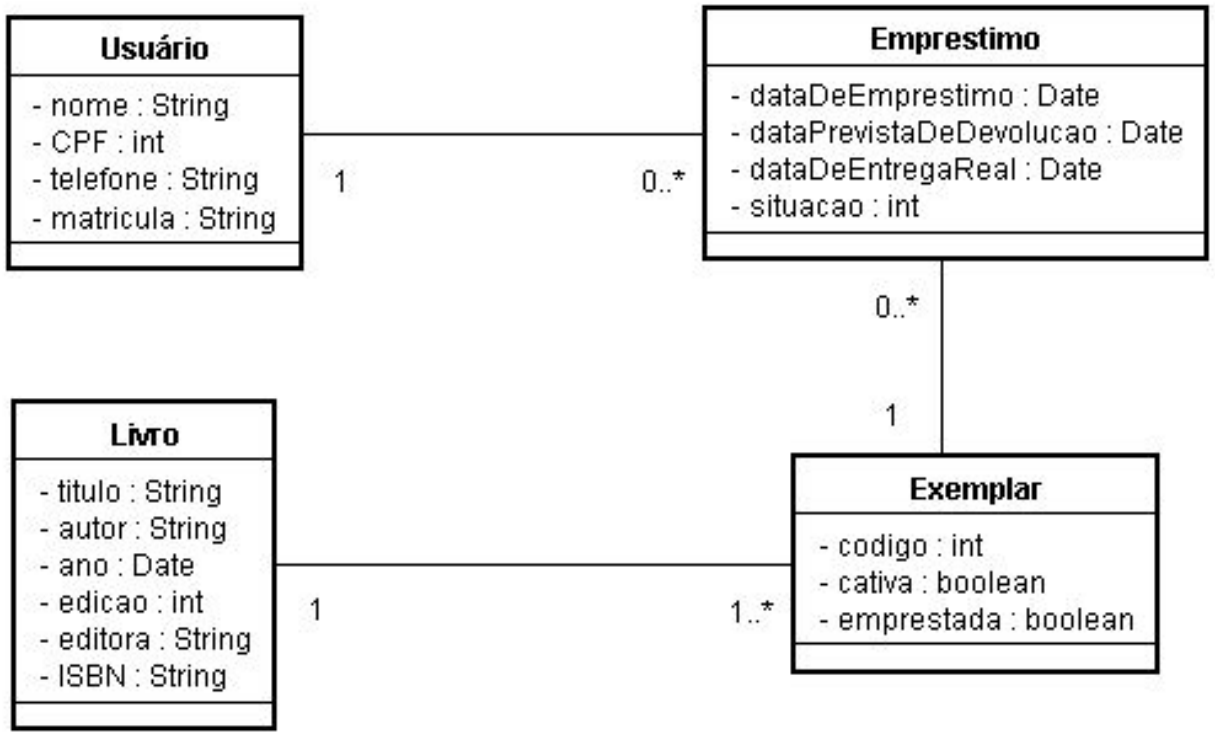
Problemática e Motivação

- Vários problemas de design de software se repetem em diferentes projetos;
- Milhares (ou até milhões) de programadores planejam e executam soluções para esses problemas;

Sistema de Vendas



Sistema de Bibliotecas



Problemática e Motivação

- Tais semelhanças também ocorrem na implementação dos sistemas;
- Uma venda pode ter muitas formas de pagamento, cada uma com uma especificidade:
  - Boleto;
  - Cartão de Crédito;
  - Cartão de Débito;

Problemática e Motivação

- Um sistema de notificação pode enviar mensagens por diferentes canais:
  - Instagram;
  - Whatsapp;
  - SMS;
- O mesmo sistema usando diferentes ***estratégias.***
- Qual seria a forma mais eficiente de implementar isso?

Problemática e Motivação

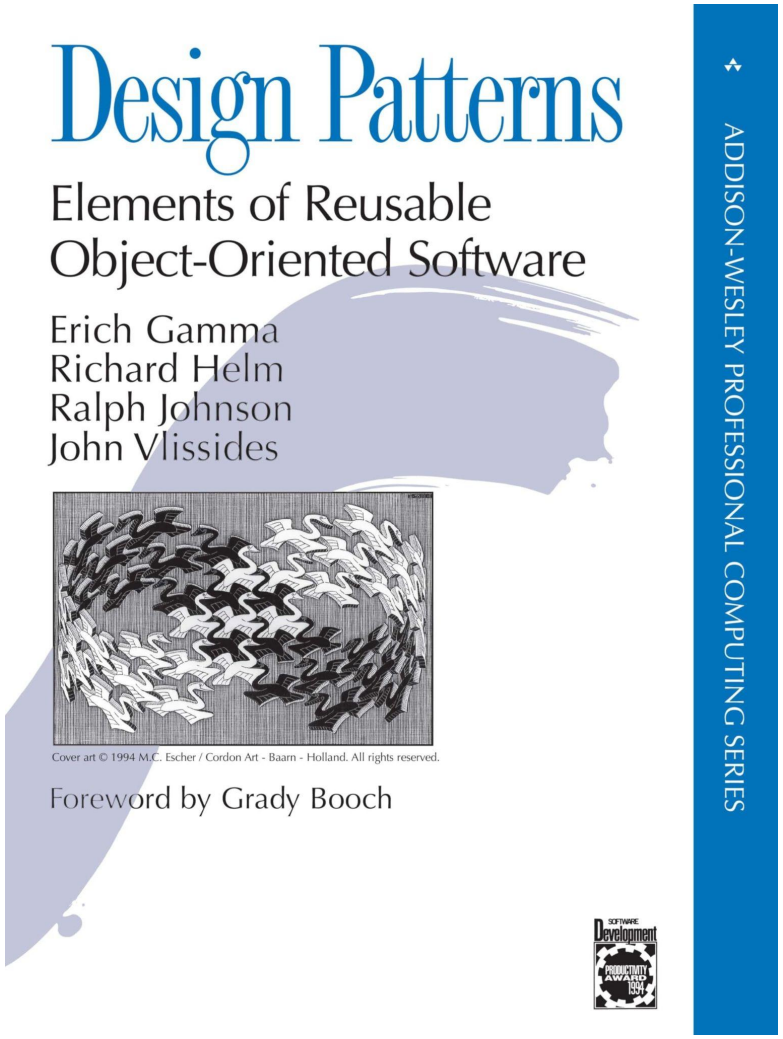
- Precisamos sempre implementar uma nova solução para os mesmos problemas?

Design Patterns

- Em 1994, quatro autores (que ficaram conhecidos como *Gang of Four - GoF)* perceberam que alguns desafios do design de software eram comuns;
- Vários desenvolvedores já haviam proposto soluções para tais problemas, e tais soluções poderiam ser re-usadas.
- Eles então catalogaram tais soluções e escreveram um livro;

Design Patterns

- O catálogo de padrões GoF é uma referência até hoje para o desenvolvimento eficiente de sistemas orientados a objetos;
- Eles descrevem 23 padrões de projetos.



O que são Design Patterns?

Design Patterns (ou padrões de projeto) são soluções reutilizáveis para problemas recorrentes no design de software. Eles fornecem um modelo comprovado para estruturar código de forma eficiente e escalável.

Por que usar Design Patterns?

- Melhor organização do código
- Maior reutilização
- Facilidade de manutenção
- Padrões amplamente conhecidos e documentados

Benefícios

- Reduzem a complexidade do código
- Facilitam a comunicação entre desenvolvedores
- Melhoram a testabilidade e extensibilidade

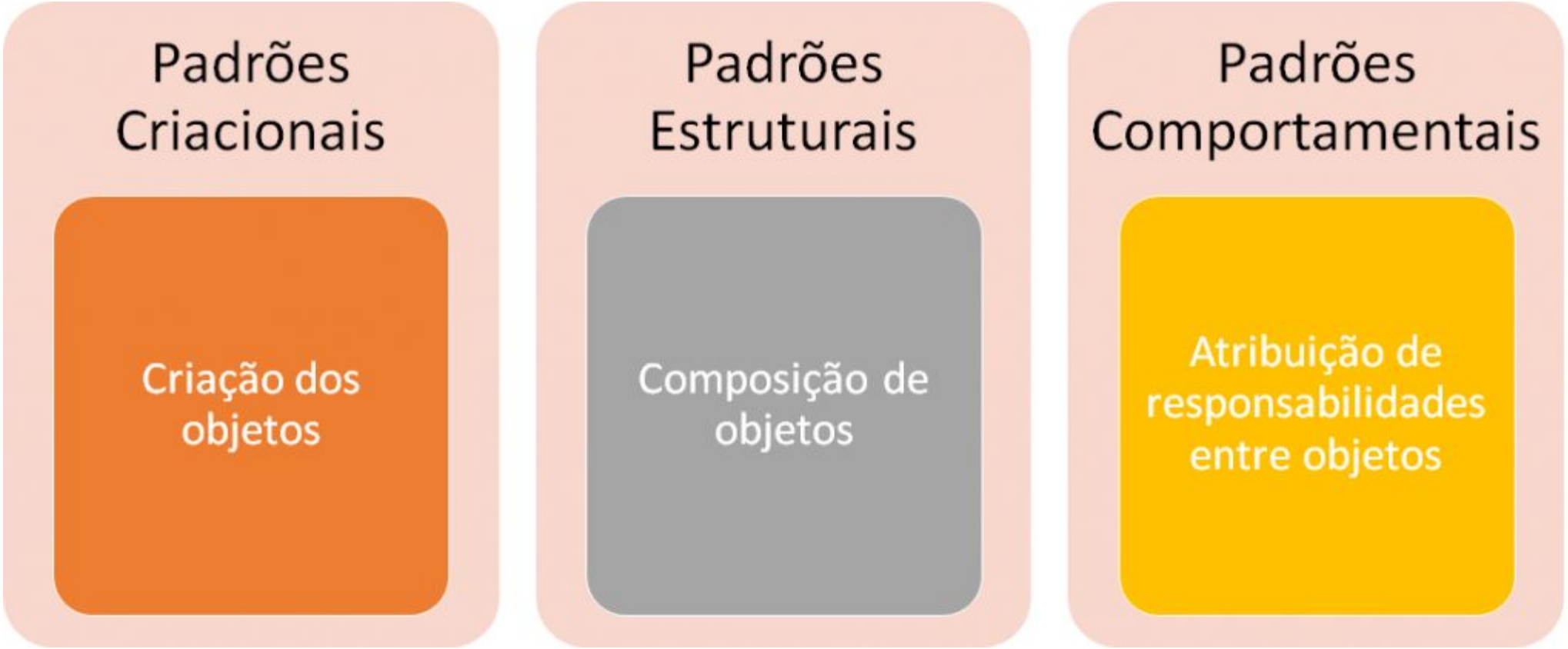
Limitações

- Requerem conhecimento prévio para aplicação correta
- Podem ser desnecessários para problemas simples
- Má implementação pode gerar código mais complexo

Como os Design Patterns se encaixam na Arquitetura de Software?

- Auxiliam na criação de sistemas modulares
- Reduzem o acoplamento entre componentes
- Melhoram a manutenção dos sistemas.

Categorias de Design Patterns



Categorias de Design Patterns

- Padrões Criacionais: *Focados na criação de objetos de forma eficiente e flexível.*

- Padrões Estruturais: *Auxiliam na organização de classes e objetos para formar estruturas maiores.*

- Padrões Comportamentais: *Definem como os objetos interagem e comunicam entre si.*

Padrões Estruturais

Padrões de projeto estruturais explicam como **compor** objetos e classes em **estruturas maiores**, mantendo essas estruturas **flexíveis** e **eficientes**.

Padrões Estruturais

- **Reutilização de código**: reduz duplicações e promove um design mais limpo.

- **Flexibilidade na arquitetura:** Permitem alterar a estrutura do sistema sem modificar muito o código existente, facilitando adaptações e extensões.

- **Facilidade de manutenção:** Ao organizar melhor as dependências entre classes e objetos, tornam o sistema mais compreensível e fácil de manter.

Padrões Estruturais

- ***Bridge (Ponte):*** Separa uma abstração da sua implementação, permitindo que ambas evoluem independentemente.

- ***Composite (Composto):*** permite que clientes tratem objetos individuais e composições de forma uniforme.

- ***Decorator (Decorador):*** Adiciona funcionalidades a objetos individualmente, de forma dinâmica, sem alterar a classe original ou afetar outros objetos.

Padrões Estruturais

- ***Facade (Fachada):*** Fornece uma interface simplificada para um conjunto complexo de classes, bibliotecas ou subsistemas.
- ***Flyweight (Peso-Mosca):*** compartilha objetos comuns de forma eficiente.
- ***Proxy:*** Fornece um substituto ou representante de outro objeto para controlar o acesso a ele.
- ***Adapter (Adaptador):*** Converte a interface de uma classe em outra interface esperada pelos clientes, permitindo que classes incompatíveis trabalhem juntas.

Padrões Comportamentais

- ***Facade (Fachada):*** Fornece uma interface simplificada para um conjunto complexo de classes, bibliotecas ou subsistemas.
- ***Flyweight (Peso-Mosca):*** compartilha objetos comuns de forma eficiente.
- ***Proxy:*** Fornece um substituto ou representante de outro objeto para controlar o acesso a ele.
- ***Adapter (Adaptador):*** Converte a interface de uma classe em outra interface esperada pelos clientes, permitindo que classes incompatíveis trabalhem juntas.

Problemática

- Um **aplicativo** baixa os dados das ações de várias fontes em formato XML e, em seguida, exibe **gráficos e diagramas.**

- Em determinado momento, você decide melhorar o **aplicativo** com uma **biblioteca** de análise inteligente de terceiros.

- Mas há um problema: a biblioteca de análise só funciona com dados no formato **JSON**.

Problemática e Motivação

- Existem duas soluções:
  - A) Alterar a API para fornecer JSON;
    -
  - B) Alterar a biblioteca para aceitar arquivos XML; ou
    -
  - C) ...

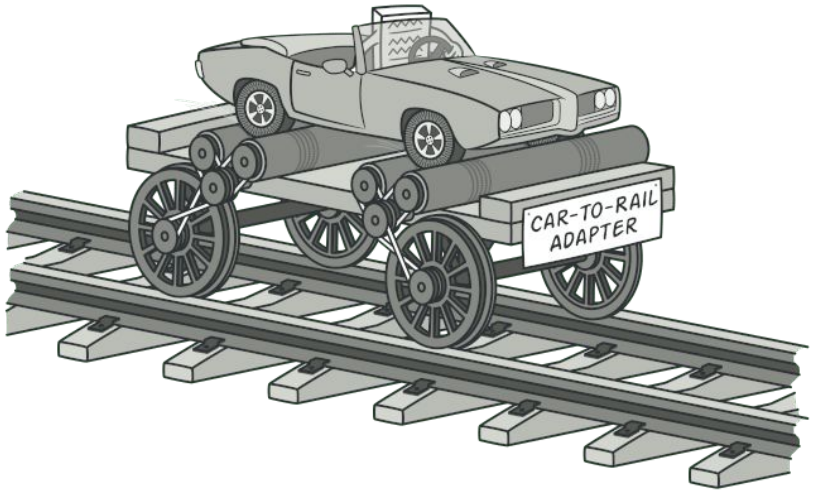
Problemática

- Algumas partes do nosso aplicativo dependem de informações em formato incompatível.

- O que fazer?

Padrão Adapter

- Permitir que duas **interfaces incompatíveis** trabalhem juntas, atuando como uma ponte entre elas.
- Quando você quer usar uma classe existente, mas sua interface não é compatível com o que o código cliente espera.

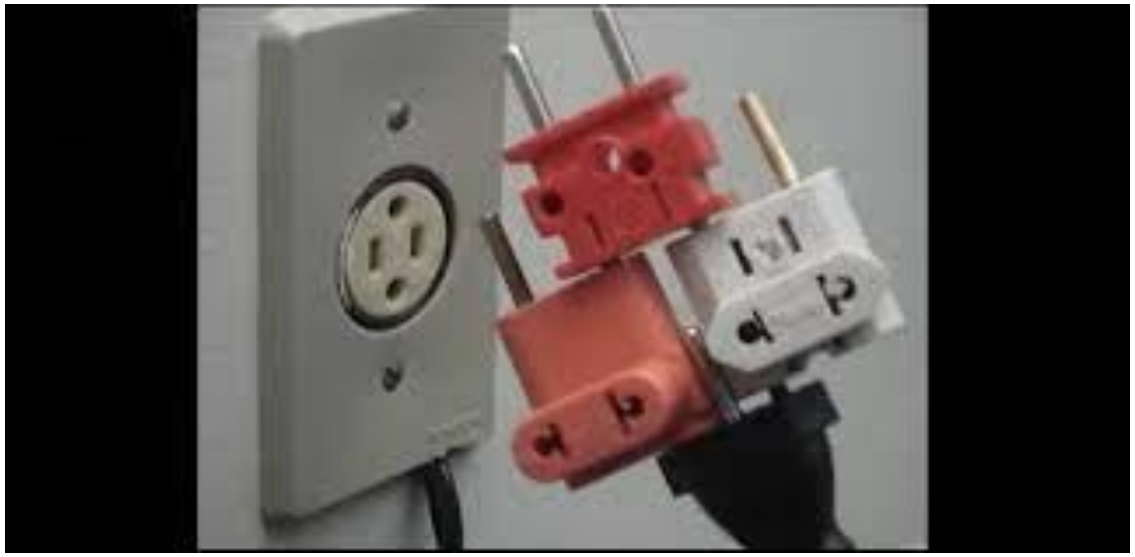


Exemplos de utilização

Exemplos de utilização

Exemplos de utilização

Exemplos de utilização



Elementos do Padrão Adapter

- ***Client:*** classe que necessita do recurso;
- ***Client Interface:*** Interface que define como o cliente precisa utilizar um recurso;
- ***Service:*** recurso a ser utilizado, mas que não é compatível com a interface do cliente;
- ***Adapter:*** interface que comunica o serviço com o cliente.

Padrão Adapter

- Converter tipos de dados (XML, JSON);
- Adapter tipos de login (Google, Facebook)
- Adaptar bibliotecas (Google Places)

Bora programar: Implementação do Padrão Strategy

DÚVIDAS?

UniCesumar