Aula 09 - Design Patterns

Disciplina: Arquitetura de Software

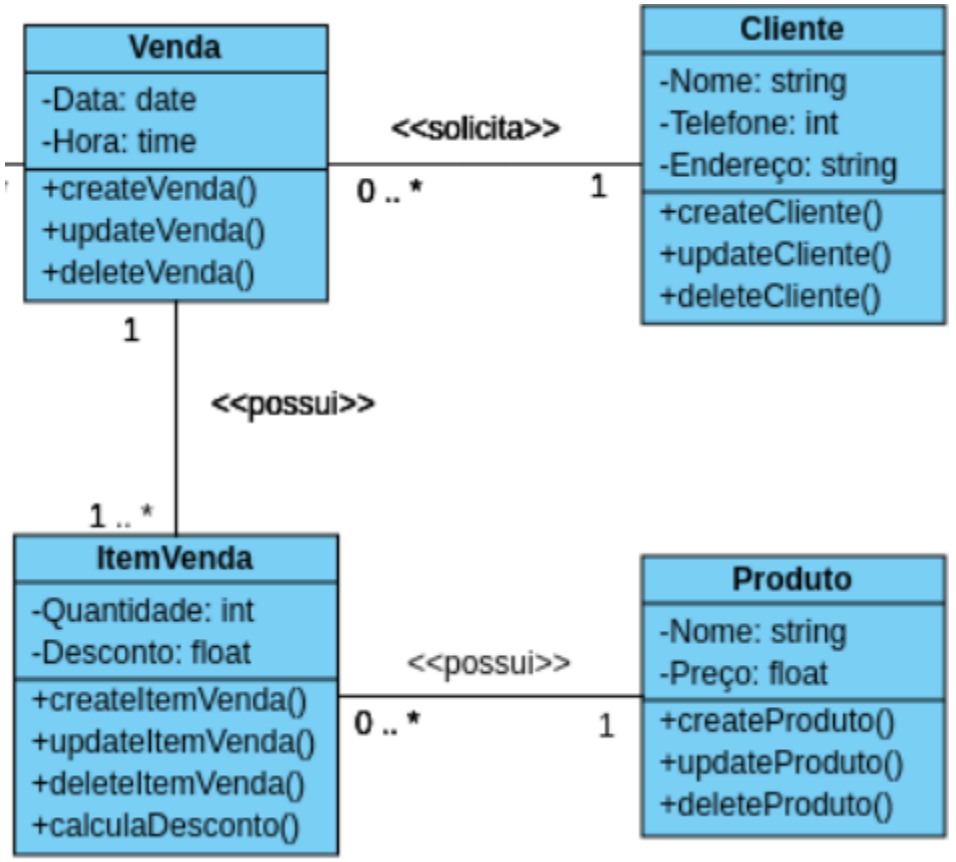
Prof. Me. João Paulo Biazotto

UniCesumar

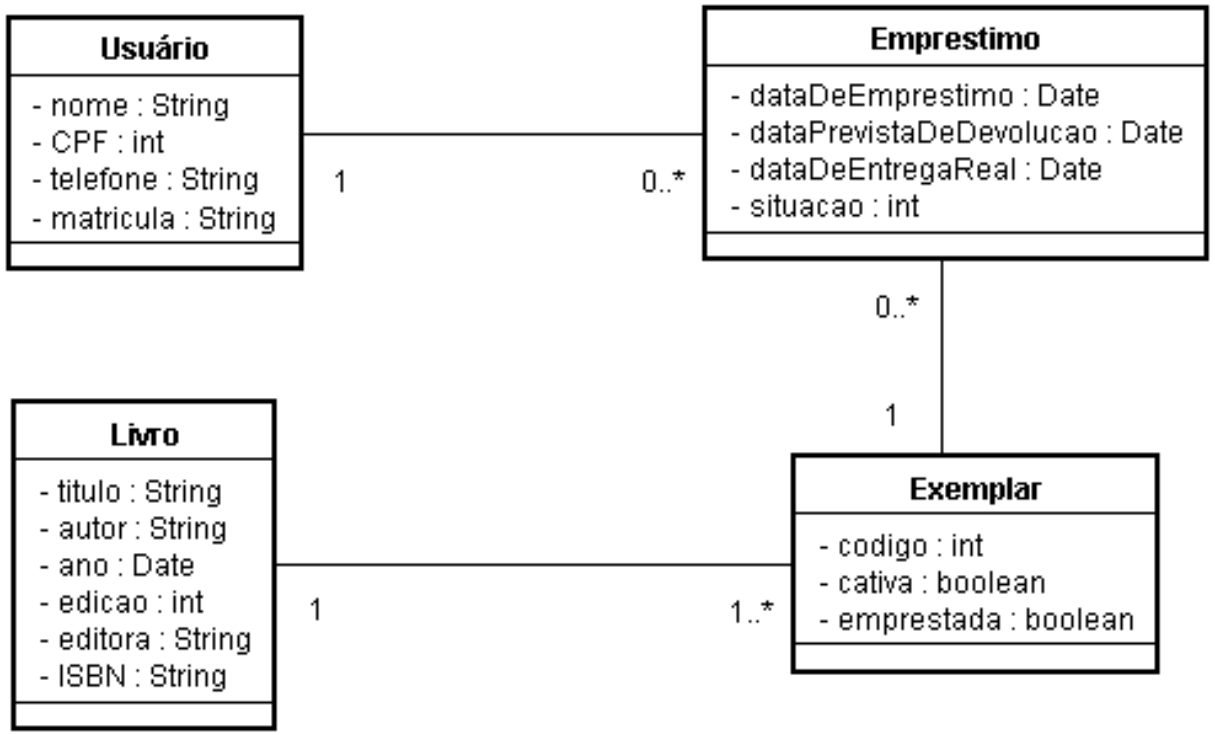
Problemática e Motivação

- 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

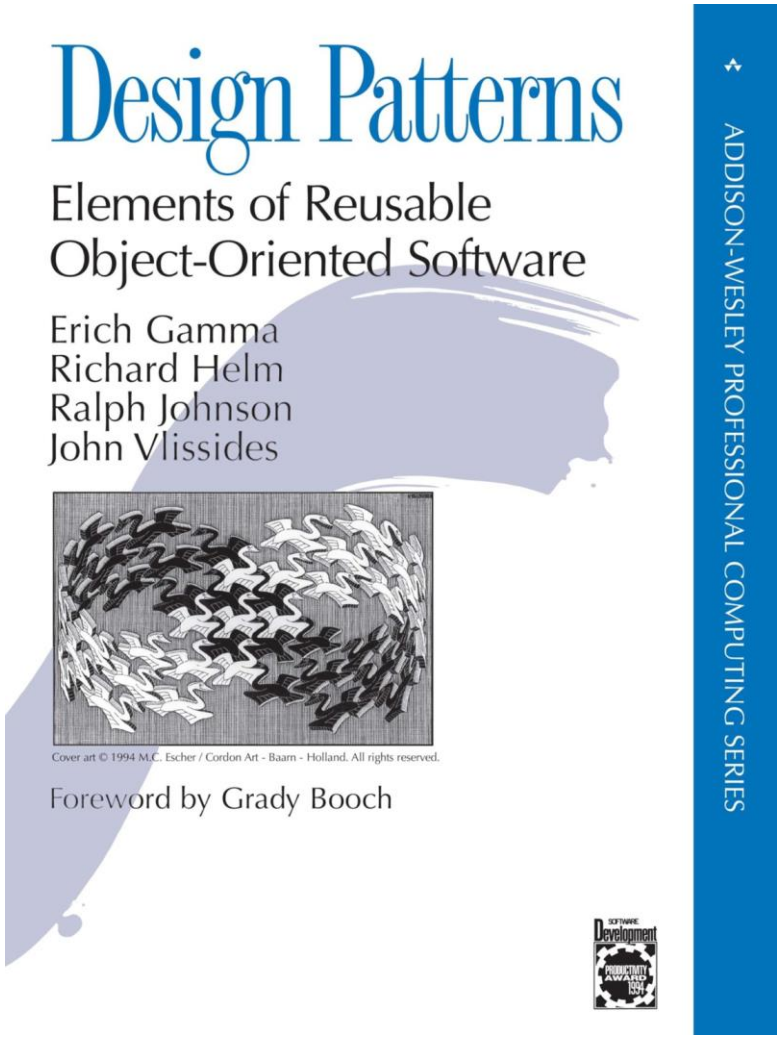
- Precisamos sempre implementar uma nova solução para os mesmos problemas?

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

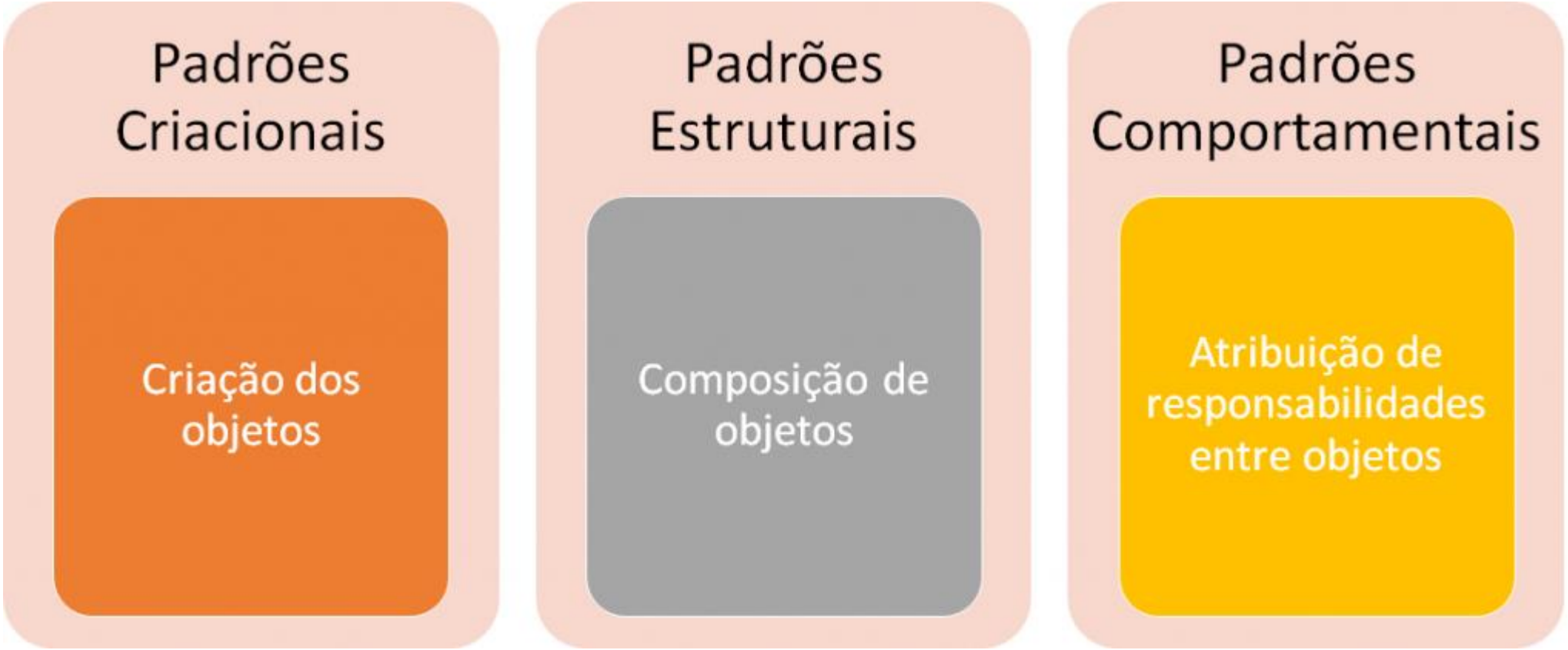
Limitações

- 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 criacionais

Os padrões de projeto criacionais **abstraem o processo de instanciação**. Eles ajudam a tornar um sistema independente de como seus objetos são criados, compostos e representados.

Padrões criacionais

Existem dois temas recorrentes nesses padrões. Primeiro, todos eles **encapsulam o conhecimento** sobre quais classes concretas o sistema utiliza. Segundo, eles **ocultam como as instâncias** dessas classes são **criadas e combinadas.**

Padrões criacionais

Consequentemente, os padrões criacionais oferecem grande **flexibilidade** em relação ao *que* é criado, *quem* o cria, *como* é criado e *quando*. Eles permitem configurar um sistema com objetos que podem variar amplamente em *estrutura* e *funcionalidade*.

Padrões criacionais

- **Singleton:** *garante que uma classe tenha apenas uma instância;*
- **Factory Method:** *define uma interface para criar objetos*;
- **Abstract Factory:** *define uma interface para criar famílias de objetos*;
- **Builder:** *permite a construção de um objeto complexo passo a passo*;
- **Prototype:** *cria objetos a partir de um protótipo*.

Padrões criacionais

- **Singleton:** *garante que uma classe tenha apenas uma instância;*
- **Factory Method:** *define uma interface para criar objetos*;
- **Abstract Factory:** *define uma interface para criar famílias de objetos*;
- **Builder:** *permite a construção de um objeto complexo passo a passo*;
- **Prototype:** *cria objetos a partir de um protótipo*.

Problemática

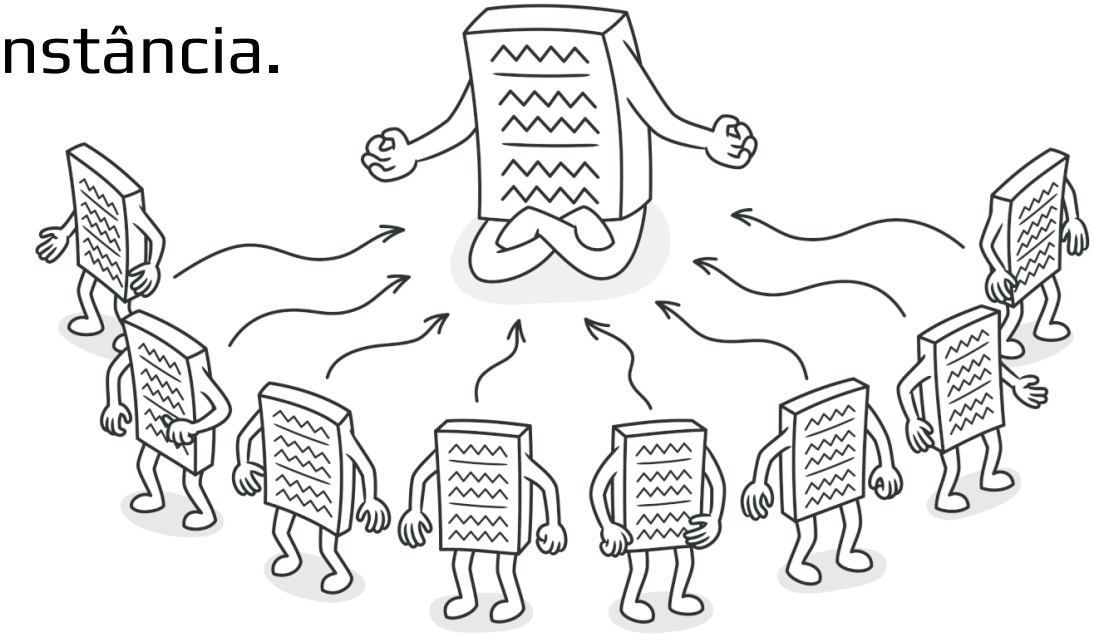
- Um sistema possui um banco de dados, no qual vários métodos podem escrever;
- Problemas:
  - Vários métodos abrem uma conexão com banco de dados -> Problemas de performance
  - Vários métodos tentam ler/escrever dados ao mesmo tempo no banco -> Problemas de integridade;

Problemática

- O problema aqui é que vários métodos estão compartilhando o mesmo recurso, e precisamos limitar isso;
- O que fazer?

Padrão Singleton

O **Singleton** é um padrão de projeto criacional que permite a você **garantir que uma classe** tenha **apenas uma instância**, enquanto provê um ponto de acesso global para essa instância.



Padrão Singleton

Útil em situações onde você deseja **compartilhar o mesmo objeto** em todo o sistema, como gerenciadores de **log**, **conexões de banco de dados**, caches, configurações globais, entre outros.

Bora programar: Implementação do Padrão Singleton

Atividade

Baseado na implementação vista em sala, e tendo como base o projeto que vem sendo desenvolvido para a Escola de T.I, implemente um logger no seu projeto. Esse logger deve registar em arquivo todas as ações que um usuário fizer com o sistema.

Padrões criacionais

- ~~**Singleton:** *garante que uma classe tenha apenas uma instância;*~~
- **Factory Method:** *define uma interface para criar objetos*;
- **Abstract Factory:** *define uma interface para criar famílias de objetos*;
- **Builder:** *permite a construção de um objeto complexo passo a passo*;
- **Prototype:** *cria objetos a partir de um protótipo*.

Problemática

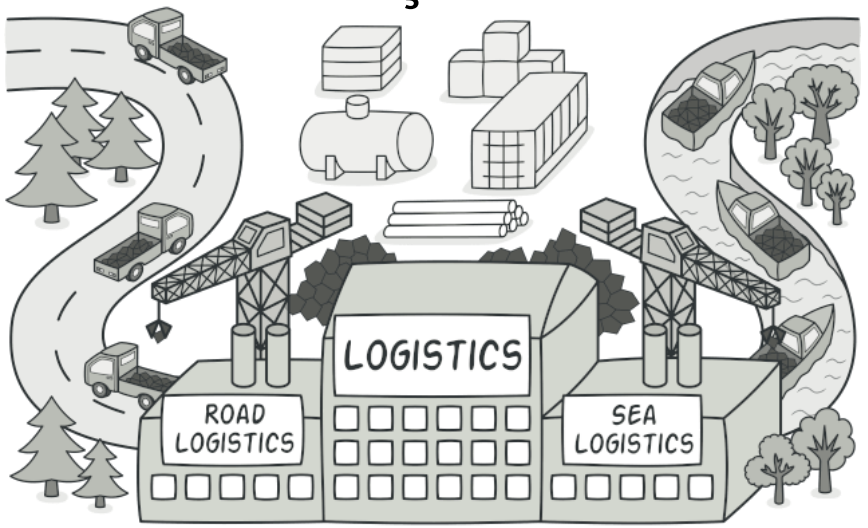
- Você está criando um leitor de documentos;
- A princípio, o leitor trabalha apenas com documentos do tipo txt;
- Ao longo do tempo, novos tipos de documentos precisam ser suportados, porém não sabemos exatamente quais tipos;

Problemática

- O problema aqui é a cada novo tipo de documento que precisa ser suportado, várias partes do código precisam ser alterados;
- O que fazer?

Padrão Factory Method

O **Factory Method** define uma interface para a criação de um objeto, mas permite que as **subclasses decidam qual classe instanciar.** Esse padrão permite que uma classe delegue a instanciação às suas subclasses.



Padrão Factory Method

- Uma classe não pode antecipar a **classe dos objetos** que deve criar.
- Uma classe deseja que suas **subclasses especifiquem os objetos** que ela cria.
- As classes delegam a responsabilidade para uma entre várias **subclasses auxiliares**, e você quer localizar o conhecimento sobre qual subclasse auxiliar é a responsável.

Bora programar: Implementação do Padrão Factory Method

Atividade

Baseado na implementação vista em sala, e tendo como base o projeto que vem sendo desenvolvido para a Escola de T.I, utilize o design pattern Factory Method no seu projeto. Tente escolher uma aplicação simples, direta e pequena para entender como o padrão pode ser usado.

DÚVIDAS?