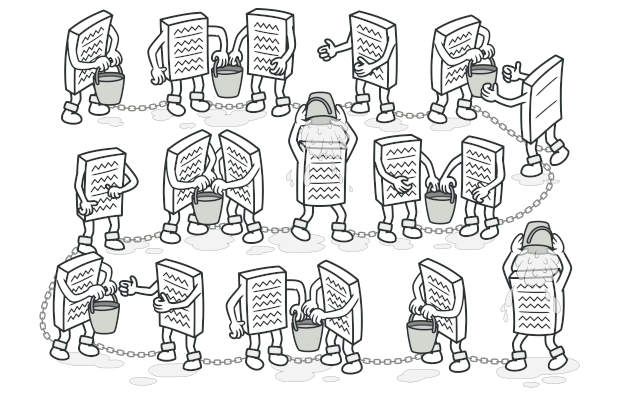
**ENGENHARIA DE SOFTWARE** 7º PERÍODO

Gilmar Luiz de Borba

O Padrão de Projeto Chain of Responsibility: Uma Jornada pela Fluidez de Responsabilidades

**\_\_\_**

Thiago Henrique Gomes da Silva



# INTRODUÇÃO

Olá a todos! Hoje, vamos explorar um dos padrões de projeto comportamental mais versáteis e poderosos: Chain of Responsibility. Este padrão nos permite criar cadeias de objetos que processam uma requisição, dando a cada objeto a chance de manipular a requisição, antes de passá-la para o próximo objeto na cadeia.

## Teoria

Imagine um sistema de aprovação de empréstimos. Cada solicitação passa por uma série de verificadores: análise de crédito, verificação de renda, avaliação de risco. Cada verificador tem sua responsabilidade específica e pode aprovar ou rejeitar a solicitação.

O padrão Chain of Responsibility permite modelar esse cenário de forma elegante e flexível. Ele define:

### Manipulador (Handler) Uma interface que define o método para lidar com uma requisição.

### Concreto Manipulador (Concrete Handler) Classes que implementam a interface Manipulador e realizam o processamento específico da requisição.

### Cadeia de Responsabilidade (Chain of Responsibility) A estrutura que conecta os manipuladores, permitindo que a requisição seja passada ao longo da cadeia.

## Exemplo em Java

[chainOfResponsibility](https://drive.google.com/drive/folders/18AzN6GrR5Qe9fYKVBQYQr2QxvKgeo99N?usp=drive_link)

## Problema que o pattern resolve:

O pattern Chain of Responsibility resolve o problema de desacoplar o emissor de uma solicitação dos seus processadores, permitindo que diferentes verificadores sejam adicionados ou removidos da cadeia sem afetar o emissor da solicitação. Isso torna o código mais flexível, modular e fácil de manter.

Além disso, o pattern Chain of Responsibility também permite que a responsabilidade seja delegada de forma transparente, sem que o emissor precise saber qual verificador irá processar a solicitação.



Exemplo: No código acima, a solicitação de empréstimo é enviada para o **VerificadorAnaliseCredito**, que decide se a aprova ou a rejeita. Se for aprovada, a solicitação é passada para o próximo verificador (**VerificadorVerificacaoRenda**) e assim por diante. Essa cadeia de responsabilidades pode ser modificada facilmente, adicionando ou removendo verificadores conforme necessário.

## Benefícios do Chain of Responsibility

* Flexibilidade: Permite adicionar ou remover manipuladores facilmente, sem afetar o código dos outros manipuladores.
* Reuso: Os manipuladores podem ser reutilizados em diferentes cadeias de responsabilidade.
* Desacoplamento: Os manipuladores não precisam conhecer a estrutura da cadeia, apenas o próximo manipulador.
* Manutenção: Facilita a manutenção e atualização do sistema, pois a responsabilidade é bem delimitada.

## SOLID

O padrão Chain of Responsibility, como implementado no código, demonstra as seguintes boas práticas dos princípios SOLID:

**SRP (Single Responsibility Principle):** Cada classe (VerificadorBase, VerificadorAnaliseCredito, VerificadorVerificacaoRenda, VerificadorAvaliacaoRisco) tem uma única responsabilidade bem definida. Cada verificador é responsável por uma etapa específica do processo de aprovação de empréstimos (análise de crédito, verificação de renda, avaliação de risco).

**OCP (Open/Closed Principle):** O sistema é aberto para extensão, mas fechado para modificação. Podemos adicionar novos verificadores à cadeia sem precisar modificar as classes existentes.

**LSP (Liskov Substitution Principle):** As subclasses de VerificadorBase (os verificadores específicos) podem ser substituídas por qualquer outra classe que implemente a interface Verificador, garantindo que o comportamento geral da cadeia seja mantido.

**ISP (Interface Segregation Principle):** A interface Verificador define apenas os métodos essenciais para o padrão, evitando a necessidade de métodos irrelevantes.

## Contra-Indicações (Anti-Patterns)

* Cadeias muito longas: Pode tornar o sistema difícil de entender e manter.
* Processamento complexo: Se a lógica de cada manipulador for muito complexa, pode complicar o código.
* Dificuldade em rastrear a requisição: Pode ser difícil identificar qual manipulador processou a requisição.

## Conclusão

O padrão Chain of Responsibility é uma ferramenta poderosa para lidar com o fluxo de requisições em sistemas complexos, oferecendo flexibilidade, desacoplamento e manutenção simplificada. No entanto, é importante ter cuidado com o design da cadeia para evitar anti-patterns que podem tornar o sistema difícil de entender e manter.

Espero que este seminário tenha sido útil para vocês!