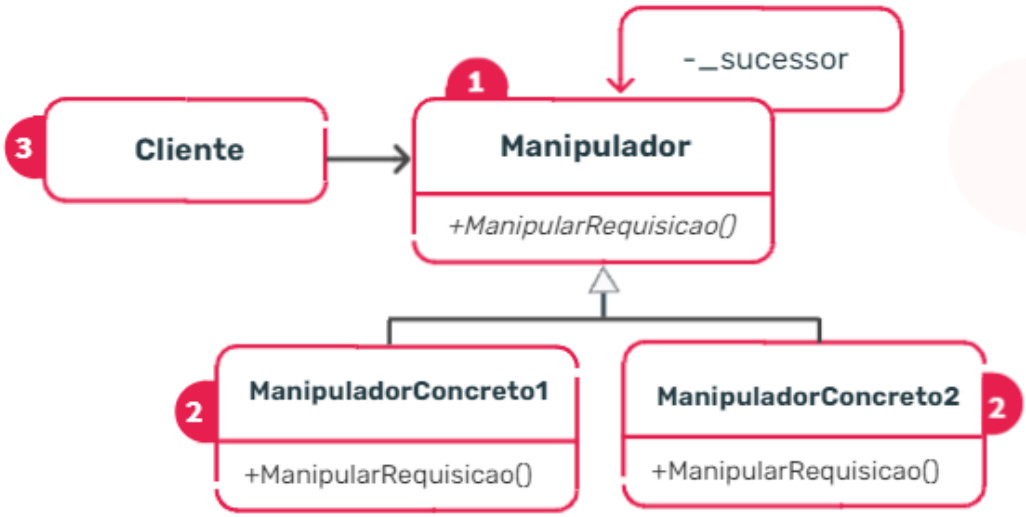
Como implementar o Chain of Responsibility pattern?

Vamos conhecer os passos clicando em cada número.



1

Criaremos uma classe abstrata que se encarregará de armazenar uma referência ao próximo manipulador da corrente e um método para enviar a solicitação do cliente ao próximo elemento da corrente.
Manipuladores concretos serão capazes de usar esse comportamento invocando o método pai.

2

Uma por uma, vamos criar subclasses manipuladoras concretas e implementaremos os métodos. Cada manipulador deve tomar duas decisões ao receber uma solicitação:

- Processar a solicitação ou;
- Passar a solicitação para o próximo link da corrente.

3

O cliente pode ativar qualquer manipulador na corrente, não apenas o primeiro. A solicitação será passada pela corrente até que algum manipulador se recuse a transmiti-la ou até que atinja o final da corrente. Devido à natureza dinâmica da rede, o cliente deve estar pronto para lidar com os seguintes cenários:

- A corrente pode consistir em um único elo.
- Algumas solicitações podem não chegar ao fim da corrente.
- Outros podem chegar ao fim da corrente sem ser gerenciados.

Conclusão

Portanto, podemos concluir que é benéfico usar o **Chain of Responsibility pattern** quando esperamos que nosso aplicativo processe diferentes tipos de solicitações e responda com resultados diferentes, mas não sabemos com antecedência quais são essas solicitações.