Design Patterns

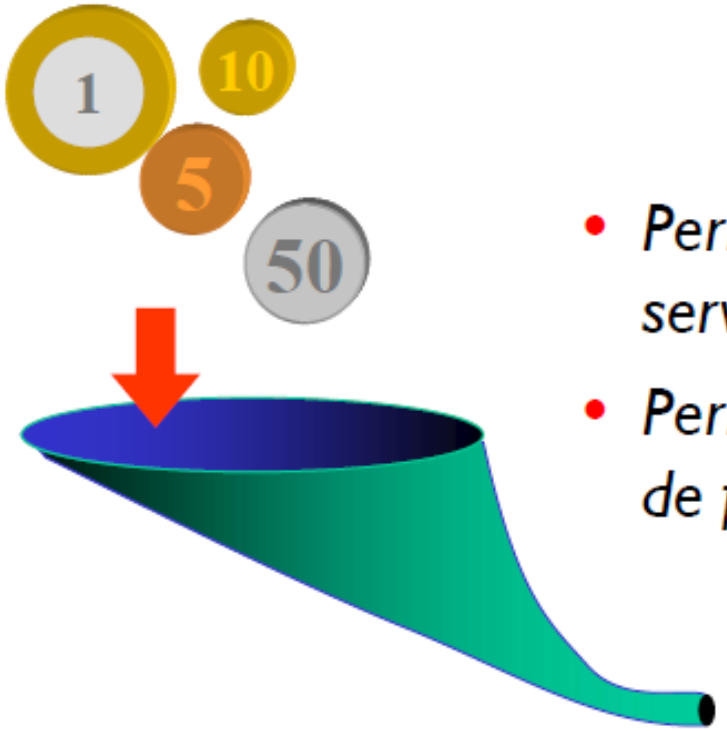
Classificação (GoF)

		Propósito		
		1. Criação	2. Estrutura	3. Comportamento
Escopo	Classe	Factory Method	Class Adapter	Interpreter Template Method
	Objeto	Abstract Factory Builder Prototype Singleton	Object Adapter Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Chain of Responsability

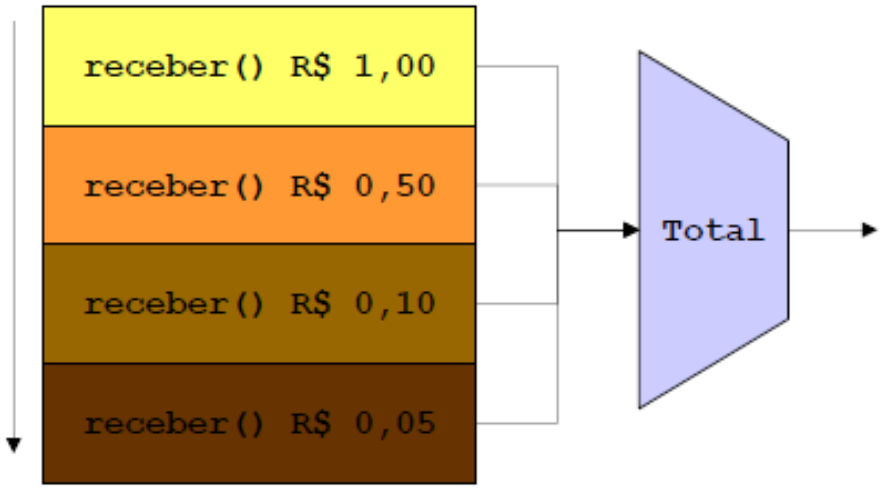
"Evita acoplar o remetente de uma requisição ao seu destinatário ao dar a mais de um objeto a chance de servir a requisição. Compõe os objetos em cascata e passa a requisição pela corrente atéque um objeto a sirva." [GoF]

Problema

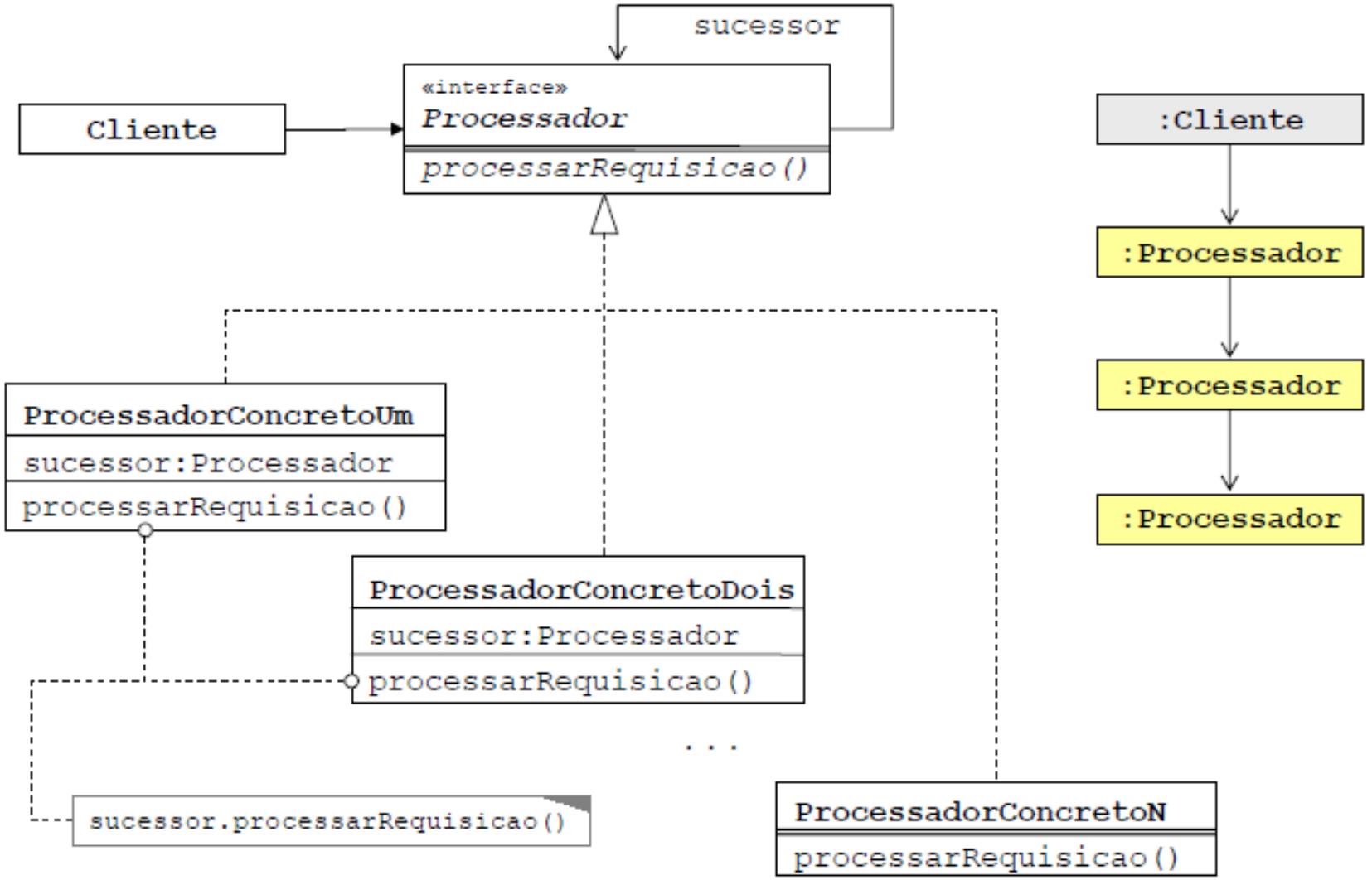


- *Permitir que vários objetos possam servir a uma requisição ou repassá-la*
- *Permitir divisão de responsabilidades de forma transparente*

*Um objeto pode ser uma **folha** ou uma **composição** de outros objetos*



Estrutura


sucessor
«interface»
Processador
processarRequisicao()
Cliente
ProcessadorConcretoUm
sucessor:Processador
processarRequisicao()
ProcessadorConcretoDois
sucessor:Processador
processarRequisicao()
...
ProcessadorConcretoN
processarRequisicao()
sucessor.processarRequisicao()
:Cliente
:Processador
:Processador
:Processador

Estratégia

- *Pode-se implementar um padrão de várias formas diferentes. Cada forma é chamada de estratégia*
- *Chain of Responsibility pode ser implementada com estratégias que permitem maior ou menor acoplamento entre os participantes*
  - *Usando um mediador: só o mediador sabe quem é o próximo participante da cadeia*
  - *Usando delegação: cada participante conhece o seu sucessor*

Prós e Contras

- ✓ Você pode controlar a ordem de tratamento dos pedidos.
- ✓ *Princípio de responsabilidade única*. Você pode desacoplar classes que invocam operações de classes que realizam operações.
- ✓ *Princípio aberto/fechado*. Você pode introduzir novos handlers na aplicação sem quebrar o código cliente existente.

- ✗ Alguns pedidos podem acabar sem tratamento.

Prática e aplicação em Java

- Considere uma aplicação que deva permitir fazer pedidos de compras. Um pedido tem uma identificação única, um tamanho (quantidade de itens), um valor e uma data.
- Todas os pedidos podem sofrer descontos sucessivos quem devem seguir uma ordem.
  - Pedidos maiores de 10 itens = 8% de desconto
  - Pedidos acima de $1000,00 = 10% de desconto
  - Pedidos feitos às segundas feiras = 5 % de desconto

Prática e aplicação em Java

- Se o pedido atender às 3 condições de desconto, aplicar:
  - 8% de desconto ➔ 10% de desconto sobre o valor ➔ 5 % de desconto sobre o valor
- Se o pedido atender às condições de tamanho e valor
  - 8% de desconto ➔ 10% de desconto sobre o valor
- Se o pedido atender às condições de valor e data
  - 10% de desconto ➔ 5 % de desconto sobre o valor
- Se o pedido atender às condições de tamanho e data
  - 8% de desconto ➔ 5 % de desconto sobre o valor
- Se o pedido atender às condições de tamanho
  - 8% de desconto
- Se o pedido atender às condições de valor
  - 10% de desconto
- Se o pedido atender às condições de data
  - 5 % de desconto sobre o valor
- Se não atender a nenhum condição
  - Sem desconto

Exercício

- Considere uma aplicação que vai cadastrar pontuação de candidatos em um concurso público. O candidato recebe uma pontuação equivalente aos acertos na avaliação, mas podem ser consideradas pontuações acrescidas.
  - Ter ensino médio completo em instituição pública (os 3 anos) : booleano
  - Ter ensino superior completo : booleano
  - Ter X anos de registro em entidade de classe : int
  - Ter X anos de experiência na área do concurso : int

  - Ensino médio completo ➔ Acresce 5 pontos
  - Ensino superior completo ➔ Acresce 10 pontos
  - Registro em entidade de classe ➔ Acresce 1 ponto por ano de registro
  - Experiência na área ➔ Acresce 2 pontos por ano de experiência

- Criar uma aplicação Java que receba Candidatos, com todos os dados, e exiba sua pontuação final.