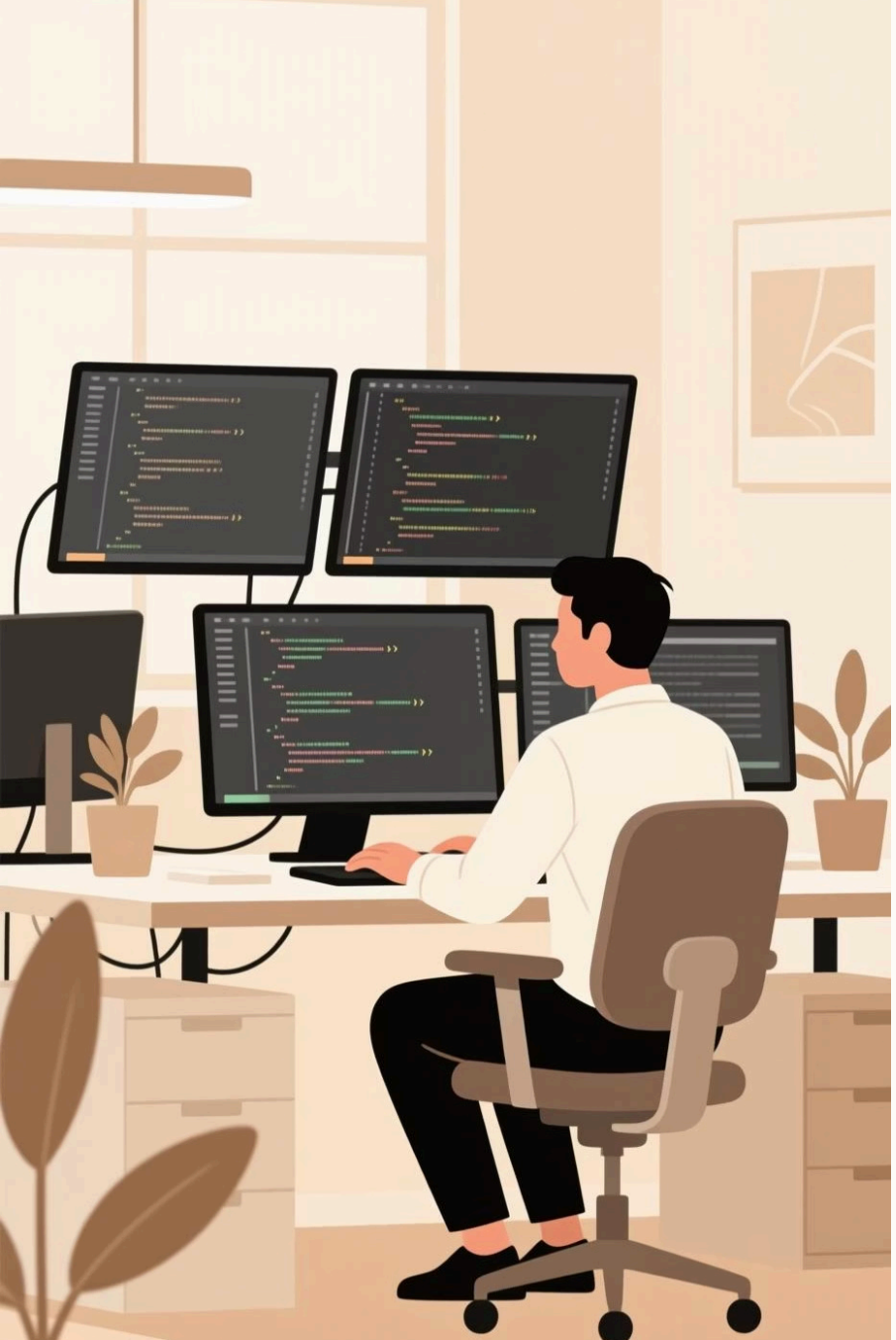
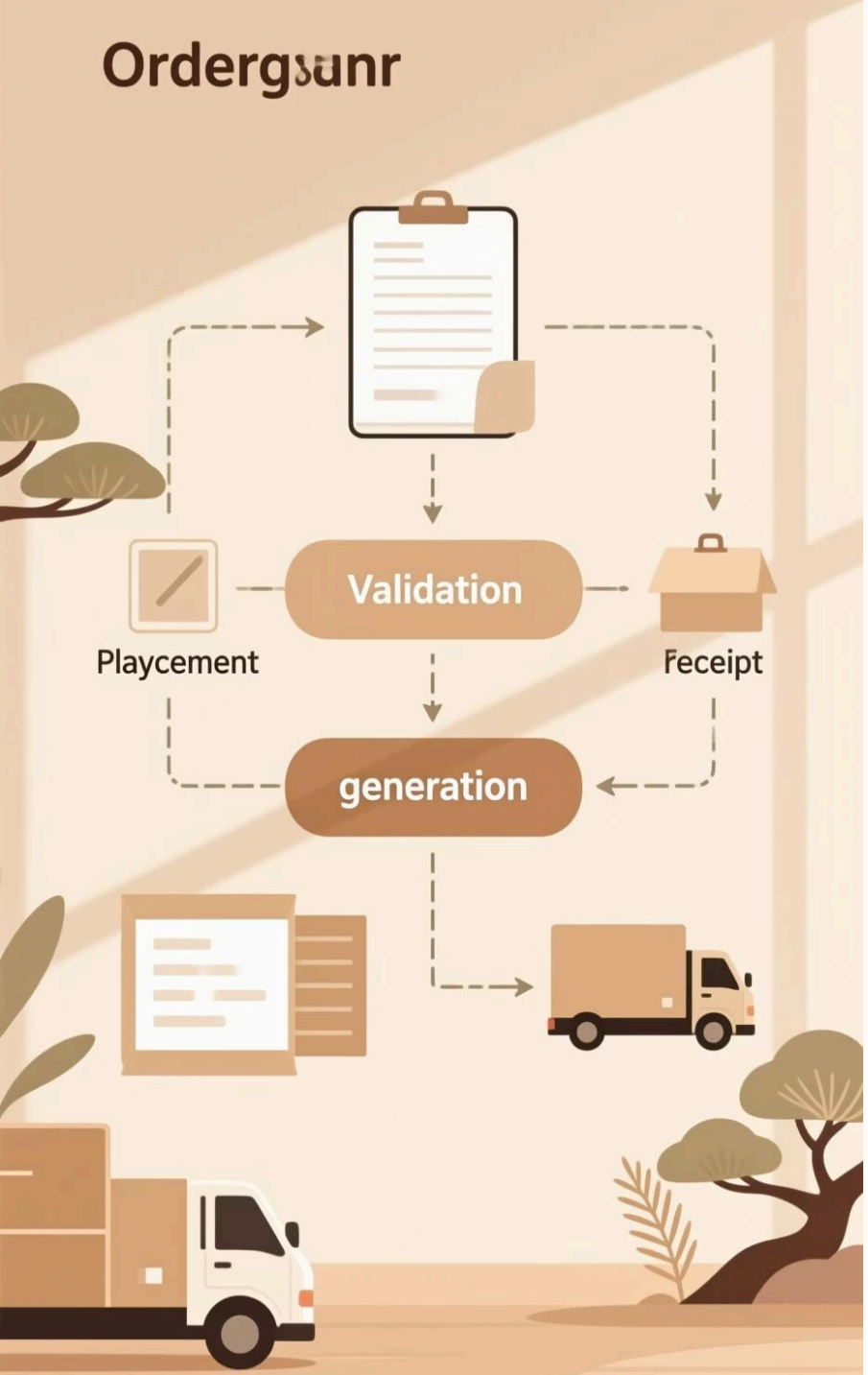


Processamento de Pedidos Orientado a Objetos

Dominando Composição, Herança e o Princípio de Substituição de Liskov



O Problema: O Ritual de Processamento de Pedidos

Todo pedido segue uma sequência fixa: validação, cálculo do subtotal e emissão de recibo. No entanto, pedidos domésticos e internacionais aplicam diferentes regras fiscais, moedas e formatos de documento.

01

Validar

Verificar a integridade do pedido e campos obrigatórios

02

Calcular Subtotal

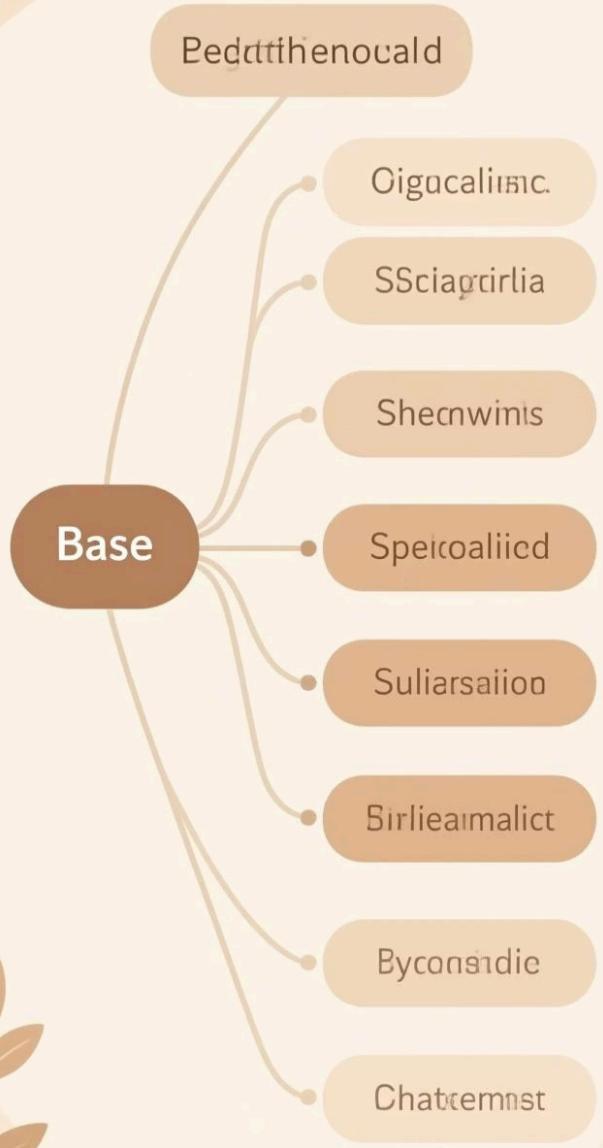
Aplicar preços base e regras específicas do domínio

03

Emitir Recibo

Gerar nota fiscal ou fatura comercial

O desafio: como reutilizar o ritual, especializando o comportamento por tipo de pedido, sem criar hierarquias de subclasse frágeis.



Estratégia de Design: Herança & Composição

Utilizamos **herança controlada** para o ritual e **composição com delegados** para preocupações transversais (crosscutting concerns). Uma classe base concreta Pedido expõe o orquestrador Processar(). Hooks virtuais (Validar, CalcularSubtotal, EmitirRecibo) permitem que as subclasses especializem o comportamento. Políticas plugáveis — frete, promoções, embalagem — são injetadas como delegados.

Herança (Ritual)

- Classe base define a orquestração
- Hooks virtuais protegidos para variação
- Subclasses sobrescrevem sem quebrar o fluxo

Composição (Políticas)

- Delegados injetados na construção
- Sem explosão combinatória de subclasses
- Políticas independentes e reutilizáveis

Princípio de Substituição de Liskov em Ação

Qualquer cliente que chame Processar() em uma referência de Pedido funciona identicamente com PedidoNacional ou PedidoInternacional—sem casting, sem condicionais. Os contratos são preservados: as regras de validação se fortalecem, mas nunca enfraquecem, os recibos de saída permanecem equivalentes e as exceções continuam previsíveis.

Substituibilidade

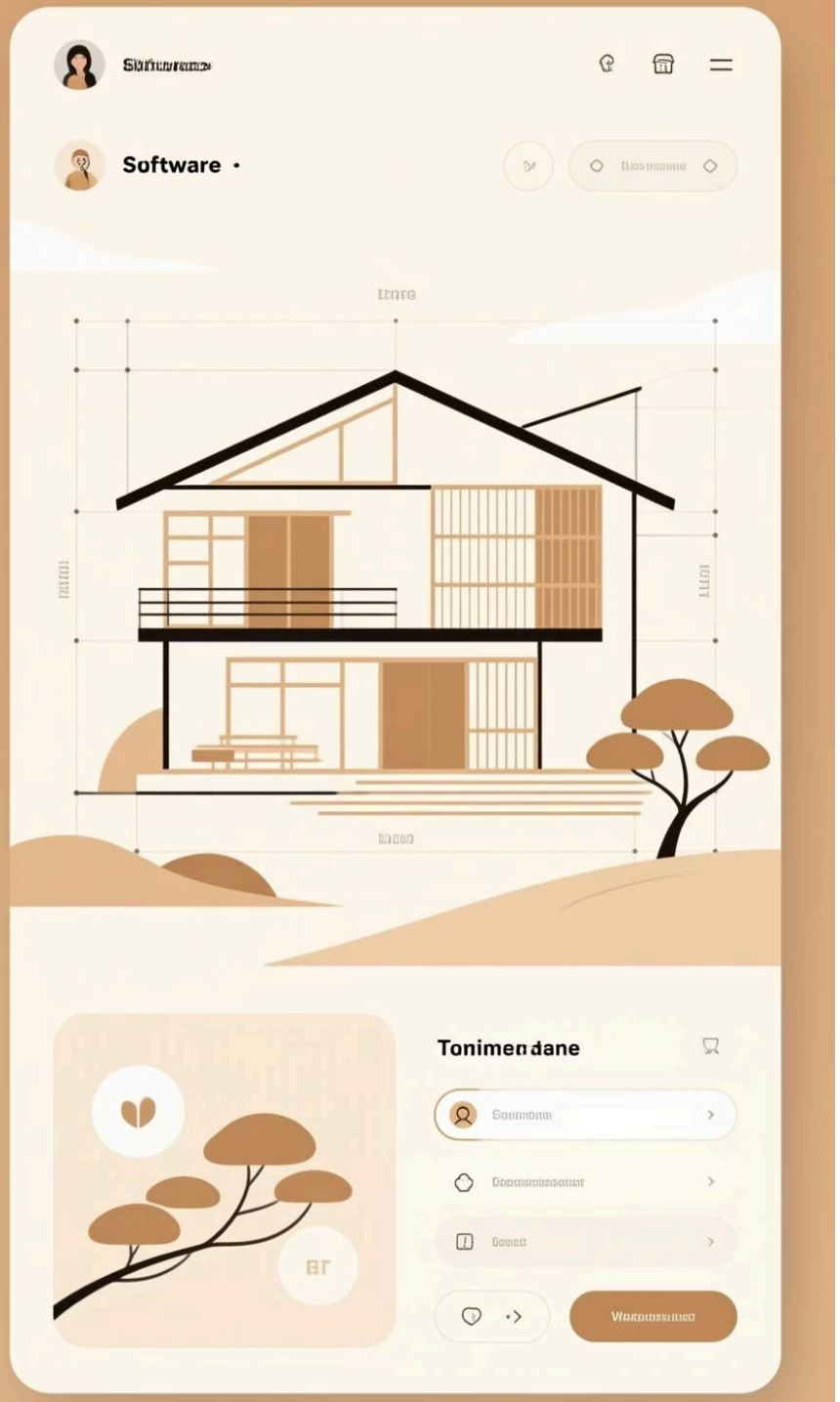
O código cliente permanece agnóstico ao tipo de pedido e delega o comportamento para a subclasse de forma transparente.

Integridade do Contrato

Invariantes da classe base não podem ser enfraquecidos; as subclasses apenas fortalecem a validação e as regras.

Políticas Plugáveis

Delegados de frete e promoção são trocados em tempo de execução sem criar novas subclasses ou quebrar o código existente.



Principais Aprendizados e Evolução

Este design obteve sucesso ao separar o ritual invariante do comportamento variável. A herança lida com o fluxo do processo principal; a composição isola políticas independentes. Evolução futura: migrar delegates para interfaces formais quando a complexidade justificar, possibilitando uma composição de políticas mais rica e contratos explícitos.

✓ O Que Funcionou

Sem explosão de subclasses, substituição compatível com LSP e reuso independente de políticas.

→ Próximo Passo

Formalizar assinaturas de delegates como interfaces para contratos mais robustos e composição de middleware.