

Fase 2: Documento de Design — Design Orientado a Objetos

1. Contrato da Classe Base: Pedido

A classe Pedido define o ritual fixo de processamento, garantindo uma sequência padronizada para todos os tipos de pedido. O método principal é Processar(), que orquestra o fluxo comum:

Método Público

Processar(): string

Responsabilidade:

Executa o fluxo completo de um pedido:

1. Validar() – Verifica se há itens e dados mínimos necessários.
2. CalcularTotal() – Calcula o valor total combinando o subtotal definido pelo tipo de pedido com as políticas plugáveis (frete, embalagem, seguro e promoção).
3. EmitirRecibo(total) – Gera o documento de saída no formato apropriado (NF-e ou Commercial Invoice).

Retorna: recibo textual coerente com o pedido processado.

Métodos Protegidos Virtuais (Ganchos de Especialização)

protected virtual void Validar()

protected abstract decimal CalcularSubtotal()

protected abstract string EmitirRecibo(decimal total)

Descrição dos Ganchos:

- Validar(): validações básicas (itens, cliente, endereço). As derivadas podem fortalecer as validações, nunca enfraquecê-las.
- CalcularSubtotal(): calcula o subtotal base — varia conforme o tipo de pedido (nacional ou internacional).
- EmitirRecibo(total): formata o recibo de acordo com o tipo e contexto fiscal.

2. Regras do LSP (Princípio da Substituição de Liskov)

Para garantir correção comportamental entre a classe base e as derivadas:

Regra 1 — Substituibilidade

Qualquer código cliente que use `Pedido p = new PedidoNacional(...); p.Processar();` ou `Pedido p = new PedidoInternacional(...); p.Processar();` deve funcionar corretamente sem precisar conhecer o tipo concreto nem realizar casts. O comportamento deve respeitar o mesmo contrato da base Pedido.

Regra 2 — Invariantes Preservados

As validações fundamentais da classe base (`Validar()`) não podem ser enfraquecidas nas subclasses. Elas podem apenas adicionar restrições, garantindo que os pedidos derivados mantenham as garantias mínimas de consistência.

Regra 3 — Contratos de Saída Equivalentes

O método `Processar()` deve sempre:

- Retornar um recibo coerente com o total calculado.
- Manter o formato esperado (string representando o documento fiscal).
- Não introduzir novas exceções que não existam na base.

Assim, todos os tipos de pedido são intercambiáveis para o cliente, respeitando o contrato da abstração.

3. Eixos Plugáveis (Composição via Delegates/Estratégias)

As políticas de frete, embalagem, seguro e promoção são componentes independentes e substituíveis. Cada uma segue a assinatura: decimal -> decimal, ou seja, recebem o valor corrente e retornam o valor ajustado.

Eixo	Delegate (assinatura)	Função/Papel
Frete	Frete: decimal -> decimal	Aplica o custo de frete ao valor atual do pedido (fixo ou percentual).
Embalagem	Embalagem: decimal -> decimal	Acrescenta o custo de embalagem ou proteção ao total.
Seguro	Seguro: decimal -> decimal	Aplica o prêmio ou percentual de seguro sobre o valor corrente.
Promoção	Promocao: decimal -> decimal	Aplica descontos, cupons ou reduções promocionais no valor final.

Resumo

- O método `Processar()` garante um template fixo de execução.
- As variações são isoladas por herança controlada (LSP).
- As políticas adicionais usam composição via delegates, garantindo flexibilidade e baixo acoplamento.