Padrões Básicos de Projeto Orientado a Objetos

Introdução

- Um sistema OO é composto de objetos que enviam mensagens uns para os outros
  - Uma mensagem é um método executado no contexto de um objeto
- Escolher como distribuir as responsabilidades entre objetos (ou classes) é crucial para um bom projeto
- Uma má distribuição leva a sistemas e componentes frágeis e difíceis de entender, manter, reusar e estender

Introdução

- Padrões de distribuição de responsabilidades
  - Padrões GRASP (General Responsibility Assignment Software Patterns).
  - Livro de Craig Larman
- Esses padrões representam princípios de um bom projeto OO.

Responsabilidades

- Responsabilidades são obrigações de um tipo ou de uma classe
- Obrigações de **fazer** algo
  - Fazer algo a si mesmo.
  - Iniciar ações em outros objetos.
  - Controlar ou coordenar atividades em outros objetos
- Obrigações de **conhecer** algo
  - Conhecer dados encapsulados.
  - Conhecer objetos relacionados.
  - Conhecer coisas que se pode calcular.

Responsabilidades

- Exemplos
  - Um objeto Venda tem a responsabilidade de criar itens da venda (fazer algo).
  - Um objeto Venda tem a responsabilidade de saber sua data (conhecer algo)

Responsabilidades

- Granularidade
  - Uma responsabilidade pode envolver um único método (ou poucos).
    - Exemplo: Criar um item de uma Venda.
  - Uma responsabilidade pode envolver dezenas de classes e métodos.
    - Exemplo: Responsabilidade de fornecer acesso a um BD.
  - Uma responsabilidade não é igual a um método. Mas métodos são usados para implementar responsabilidades.

O Padrão Expert

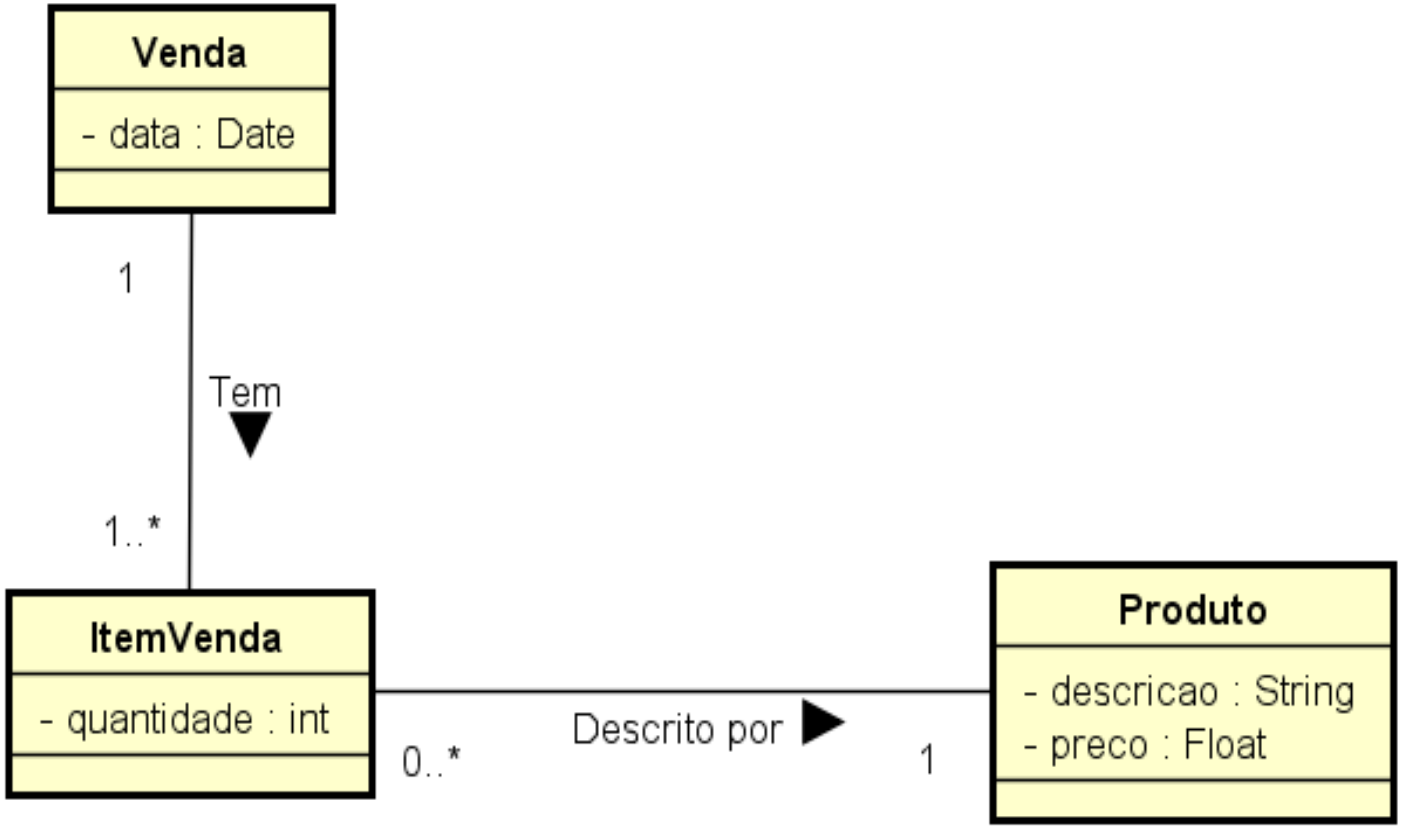
- Problema
  - Qual é o princípio mais fundamental para atribuir responsabilidades?
- Solução
  - Atribuir uma responsabilidade ao expert de informação, ou seja, a classe que possui a informação necessária para preencher a responsabilidade.

O Padrão Expert

- Exemplo
  - Num sistema de Ponto de Venda (TPDV), alguma classe precisa saber o total de uma venda.
  - Podemos dizer isso sobre a forma de uma responsabilidade:
    - Quem deveria ser responsável pelo conhecimento do total de uma venda?
  - Pelo padrão Expert, escolhemos a classe que possui a informação necessária para determinar o total.

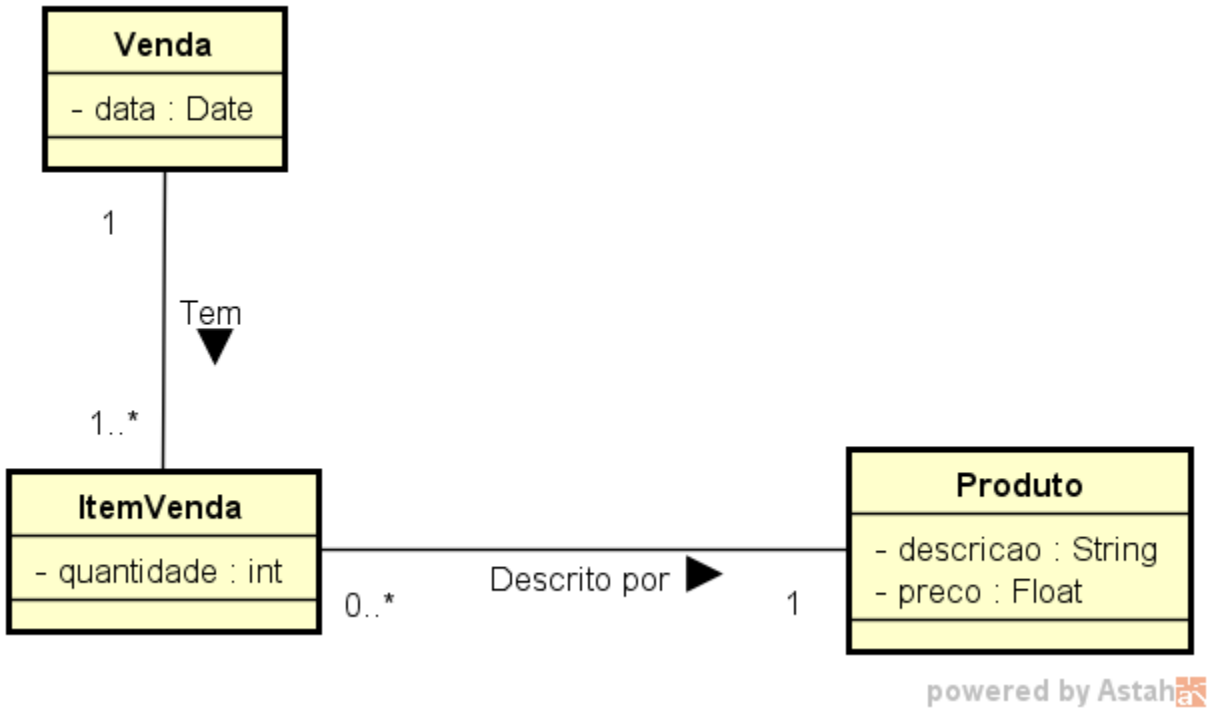
O Padrão Expert

- Considere esse modelo parcial:



O Padrão Expert

- Considere esse modelo parcial:

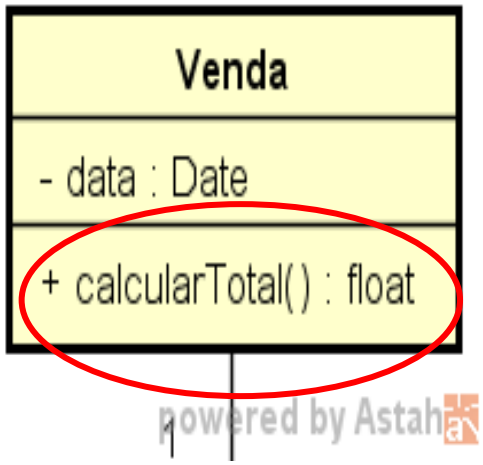


Precisamos conhecer (ter acesso a) todos ItemVenda.

O expert dessa informação é Venda.

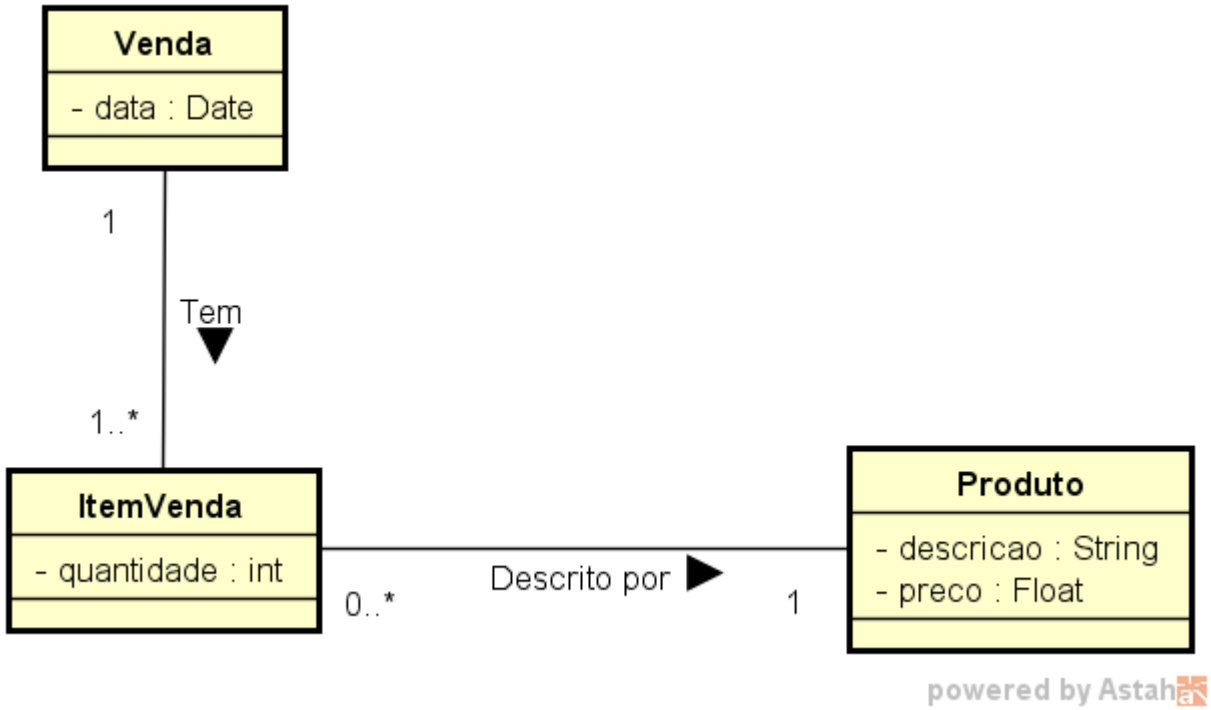
O Padrão Expert

- Então, a responsabilidade é da classe Venda



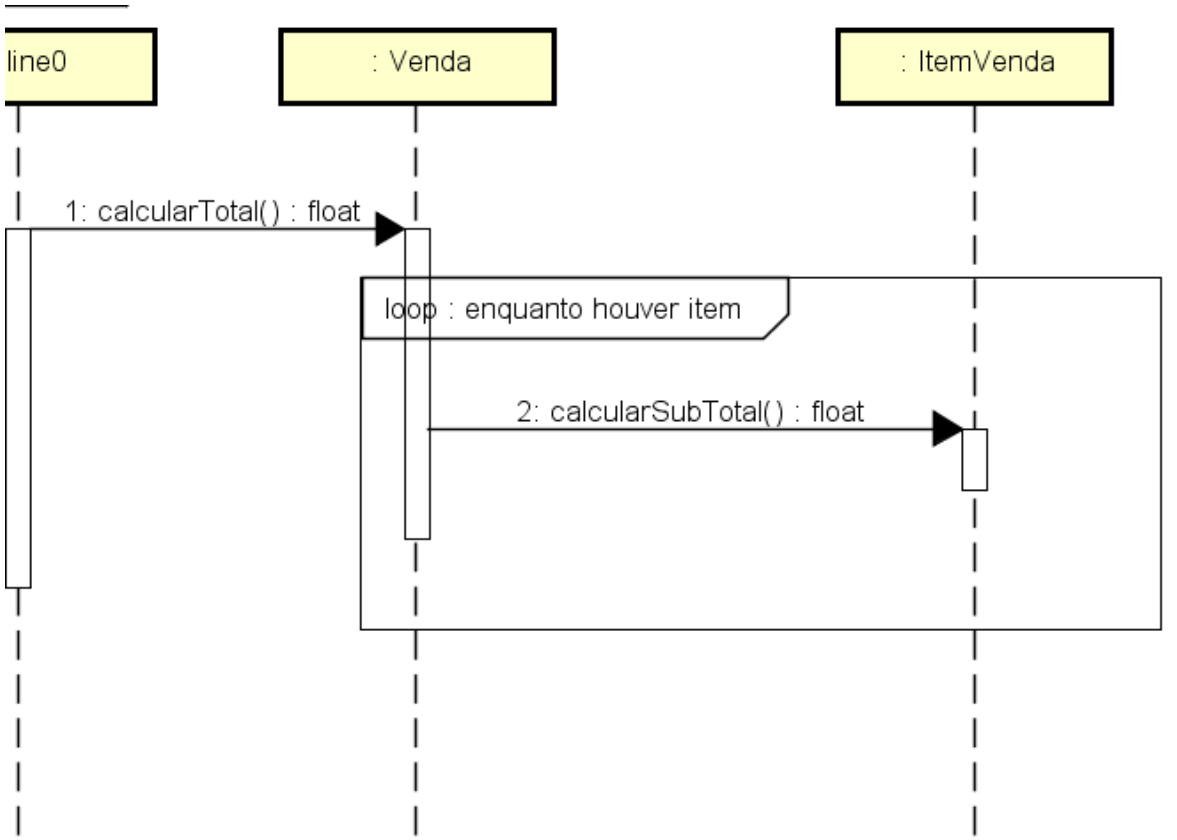
O Padrão Expert

- Para calcular o total da venda, é necessário saber o subtotal de item da venda, que é quantidade vezes o preço do produto.



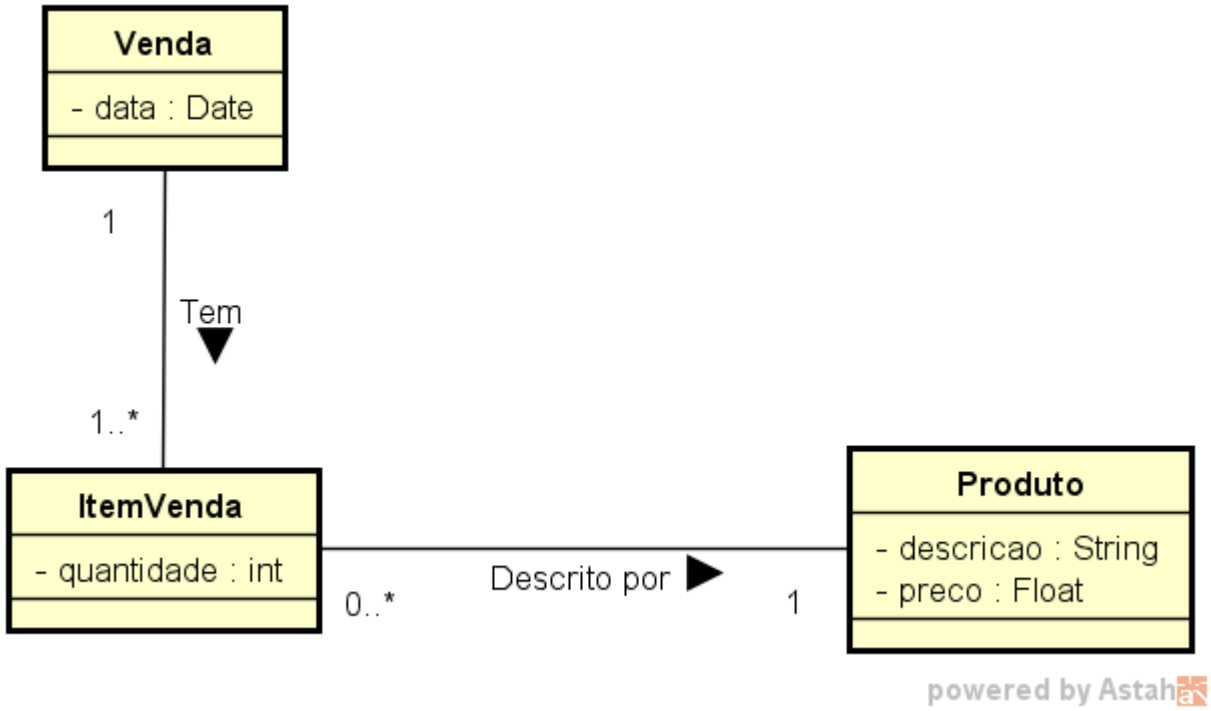
O Padrão Expert

- Quem tem essa informação é ItemVenda (expert)
  - Então é responsabilidade de fornecer o subtotal é dessa classe



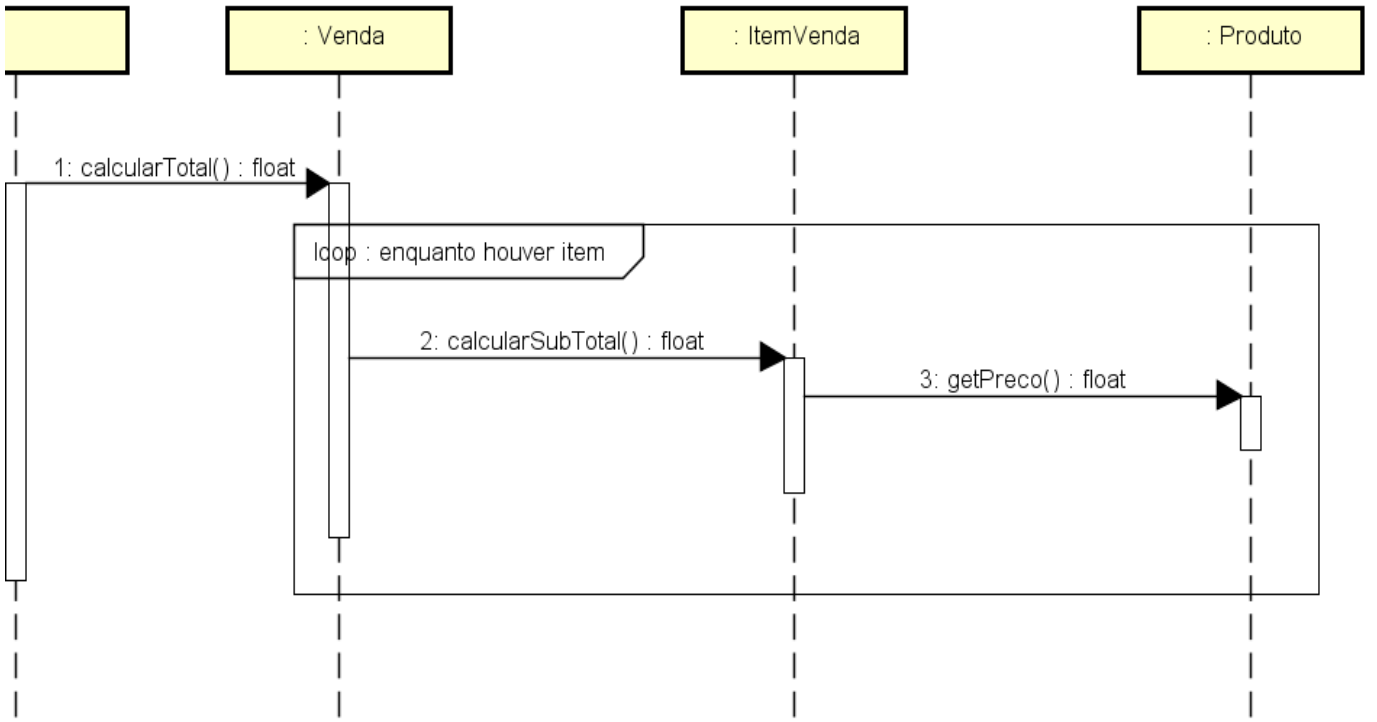
O Padrão Expert

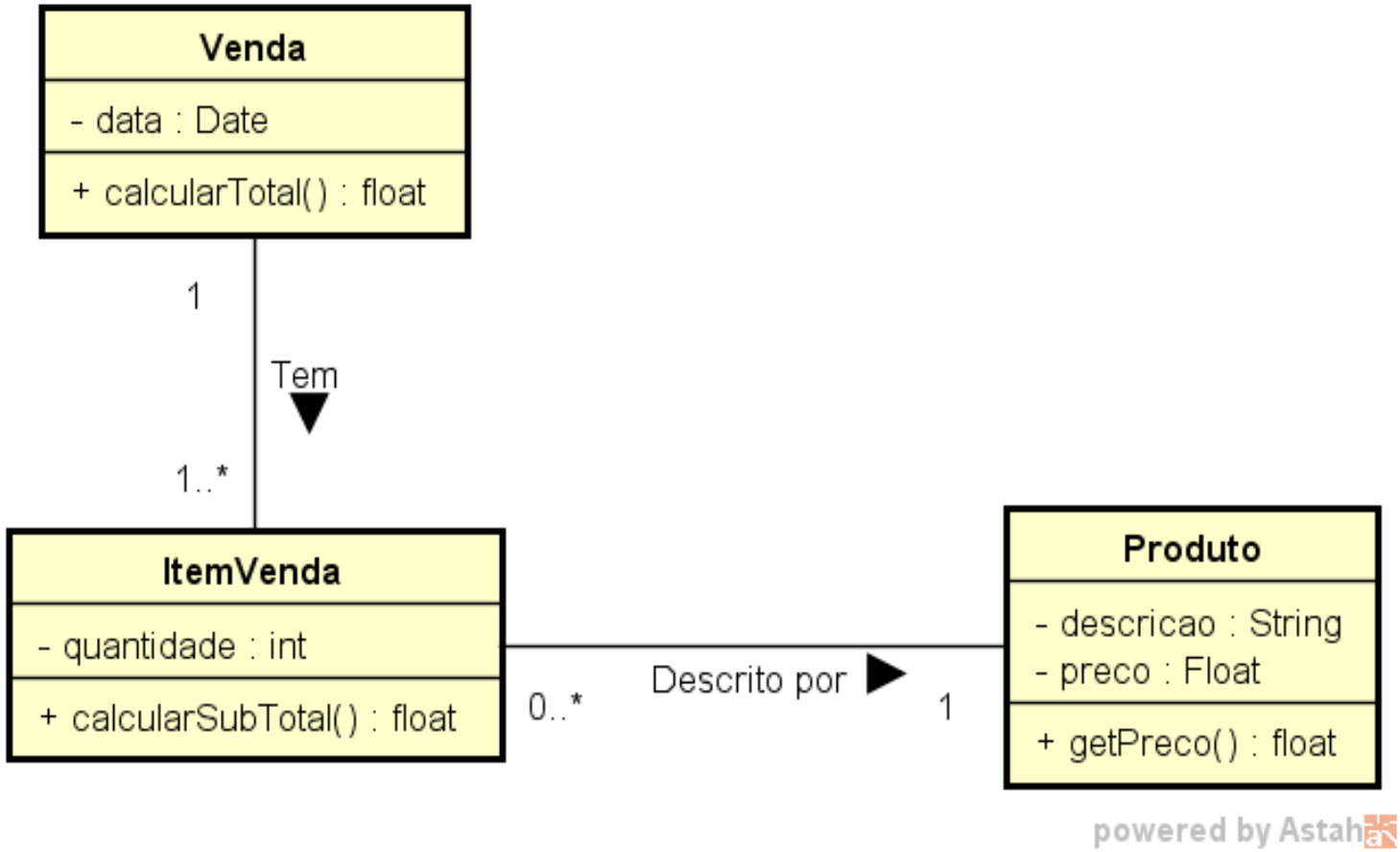
- Quem deve ter a responsabilidade de fornecer o preço do produto?



O Padrão Expert

- Quem tem essa informação é Produto(expert)
  - Então é responsabilidade de fornecer o preço é dessa classe




Venda
- data : Date
+ calcularTotal( ) : float
1
Tem
1..*
ItemVenda
- quantidade : int
+ calcularSubTotal( ) : float
0..*
Descrito por
1
Produto
- descricao : String
- preco : Float
+ getPreco( ) : float
powered by Astah

O Padrão Creator

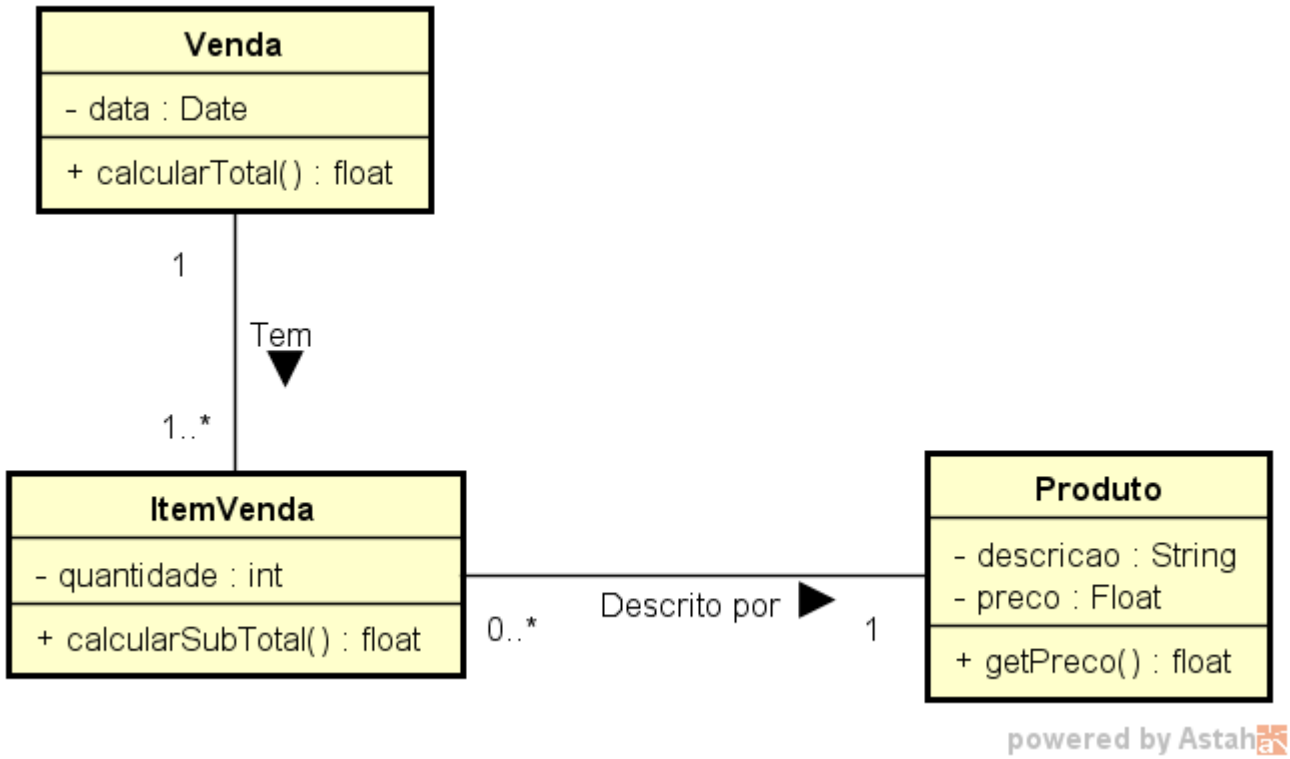
- Problema
  - Quem deve criar novas instâncias de uma classe?
- Solução
  - Atribua à classe B a responsabilidade de criar instâncias da classe A se uma das seguintes condições for verdadeira:
    - B agrega objetos da classe A
    - B contém objetos da classe A
    - B registra instâncias da classe A
    - B usa de maneira muito próxima objetos da classe A
    - B tem dados usados para inicializar A

O Padrão Creator

- Se mais de uma opção for aplicável, prefira uma classe B que agregue ou contenha objetos da classe A

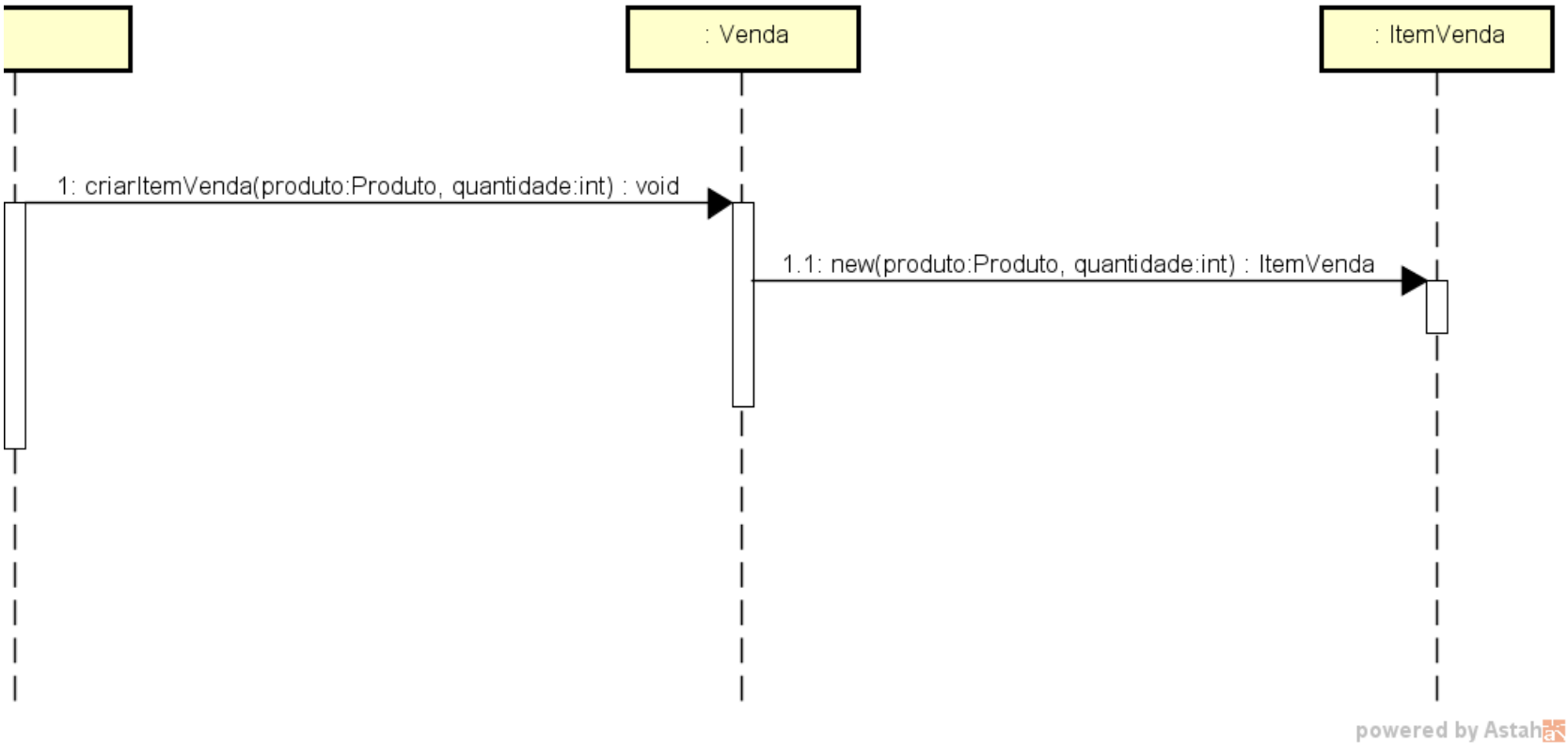
O Padrão Creator

- No nosso exemplo de TPDV, quem deve ser responsável por criar uma instância de ItemVenda?



O Padrão Creator

- Venda agrega objetos de ItemVenda
  - É um bom candidato para ter a responsabilidade de criar instâncias de ItemVenda

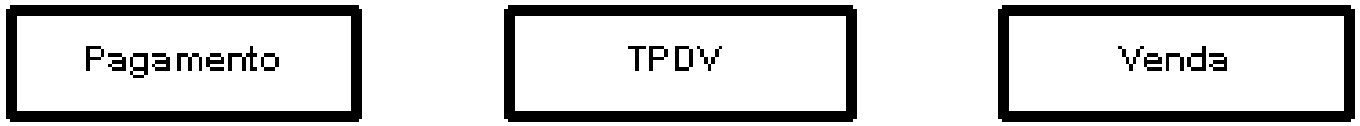


O Padrão: Baixo Acoplamento

- Problema
  - Como minimizar dependências entre as classes?
- Solução
  - Atribuir responsabilidades de forma a minimizar o acoplamento;
- Acoplamento
  - É uma medida de quão fortemente uma classe está conectada, possui conhecimento ou depende de outras classes;
  - Com fraco acoplamento, uma classe não é dependente de muitas outras classes.
  - Com uma classe possuindo forte acoplamento, temos os seguintes problemas:
    - Mudanças em uma classe relacionada força mudanças locais à classe
    - A classe é mais difícil de entender isoladamente
    - A classe é mais difícil de ser reusada, já que depende da presença de outras classes

O Padrão: Baixo Acoplamento

- Considere as seguintes classes do exemplo de Ponto de Venda:

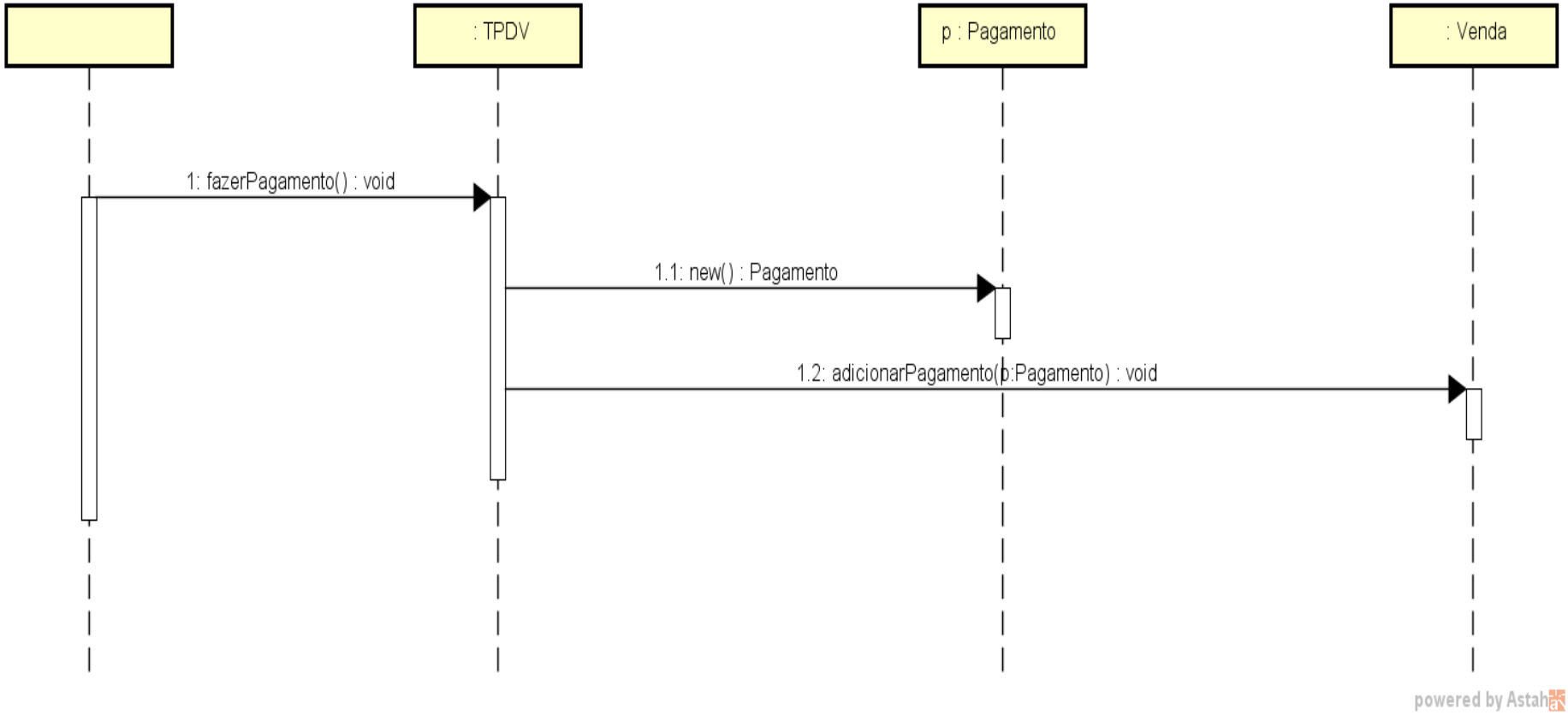


- Suponha que precisamos criar uma instância de Pagamento e associa-lá a Venda. Qual classe deveria ser responsável por isso?

O Padrão: Baixo Acoplamento

- Uma vez que TPDV *registra* pagamentos no mundo real, a classe TPDV é uma candidato a criar a instância de Pagamento.

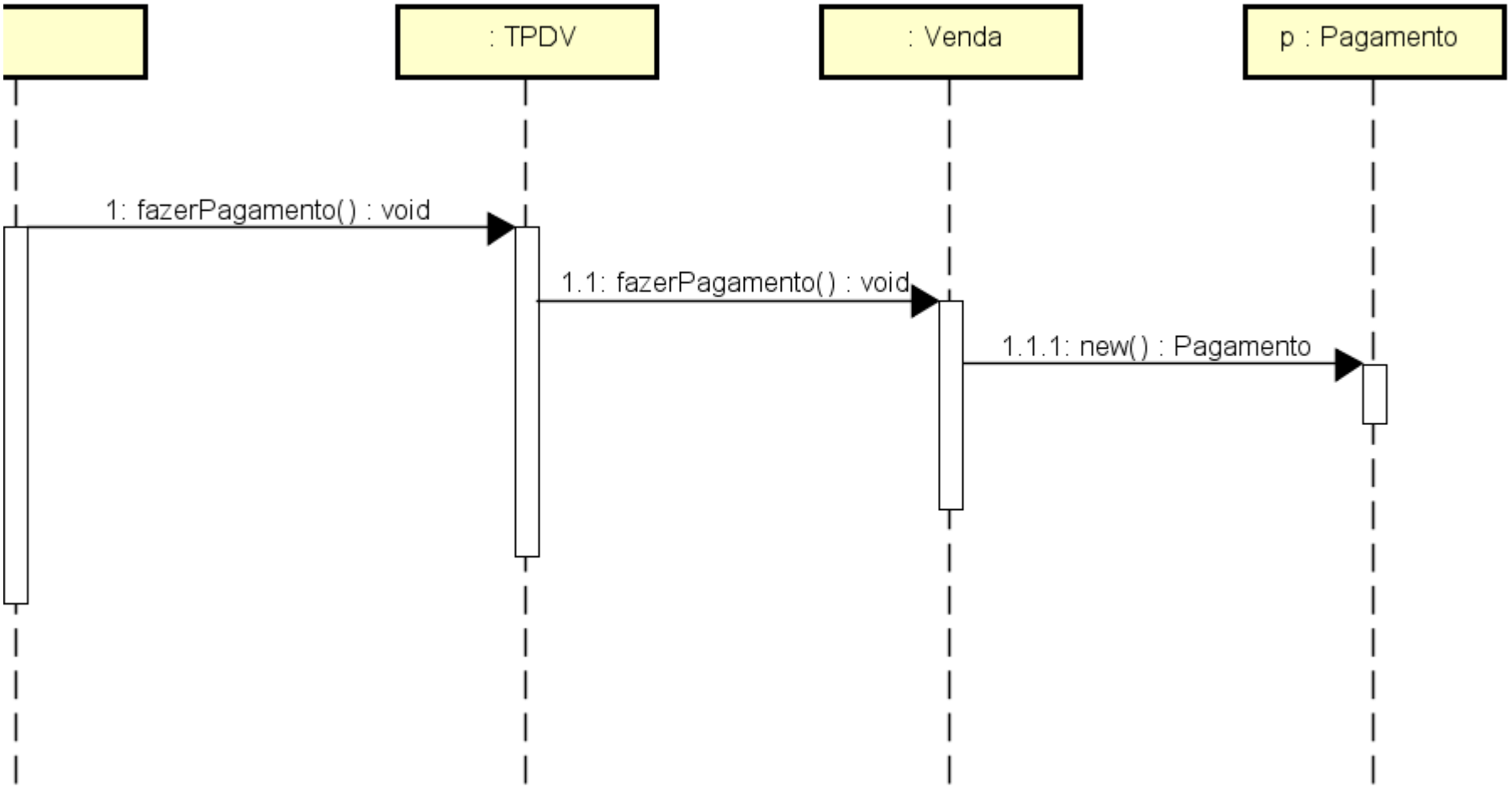
- Então poderíamos ter a seguinte solução:



- Essa atribuição de responsabilidade **acopla** TPDV a Pagamento.

O Padrão: Baixo Acoplamento

- Um solução de projeto alternativa poderia ser:



O Padrão: Baixo Acoplamento

- Supondo que em ambos os casos Pagamento deve no final ser associado à Venda, então a primeira alternativa cria uma acoplamento a mais entre TPDV e Pagamento.

Referências Bibliográficas

- Craig Larman. Applying UML and Patterns.
- Craig Larman. Utilizando UML e Padrões.