

Padrões de Projeto

1 – Especialista (Expert)

- O primeiro padrão que deve ser considerado para resolver o problema de atribuição de **responsabilidade é o Especialista**, que diz que o primeiro candidato a receber a responsabilidade é aquele que possui a informação necessária para executar a tarefa.
- O padrão especialista é responsável por **atribuir responsabilidade à classe** que tem informação necessária para assumir aquela responsabilidade.



Padrões de Projeto

- A informação necessária para uma determinada classe geralmente está espalhada entre várias classes, e através deste padrão é possível descobrir esta informação, bem como no mesmo **caminho percorrido** para esta descoberta, conhecer outras informações especializadas das outras classes que estão relacionadas entre si.
- As consequências do uso deste padrão é que o encapsulamento é mantido, já que objetos usam sua própria **informação para cumprir responsabilidades**, leva ao acoplamento fraco entre objetos e à alta coesão já que objetos fazem tudo que é relacionado à sua própria informação.



Padrões de Projeto

Problema

- Qual é o princípio básico da atribuição de responsabilidades a objetos? Um modelo de Projeto pode definir dezenas ou centenas de classes de software e uma aplicação pode exigir a satisfação de centenas ou milhares de responsabilidades. Durante o projeto orientado a objetos, quando **as interações entre objetos são definidas**, fazemos escolhas de atribuição de responsabilidades a classes. Se bem feitas as escolhas tornam os sistemas fáceis de compreender, de manter e de estender, e há a possibilidade de reutilizar os componentes em aplicações futuras.



Padrões de Projeto

Solução

- Atribua a responsabilidade ao especialista: a classe que tem as informações necessárias para assumir a responsabilidade

Exemplo

- Em um sistema de vendas, alguma classe precisa conhecer o total geral de uma venda.



Padrões de Projeto

Comece atribuindo responsabilidades claramente definidas

- Quem deve ser o responsável por conhecer o total geral de uma venda?
- Segundo o **padrão especialista** devemos procurar a classe de objetos que tem a informação necessária para determinar o total.
- Chegamos agora a uma questão chave: olhamos no modelo de domínio ou no modelo de projeto para analisar as classes que tem **a informação necessária**? O modelo de domínio ilustra classes conceituais do domínio no mundo real; o modelo de projeto ilustra classes de software.



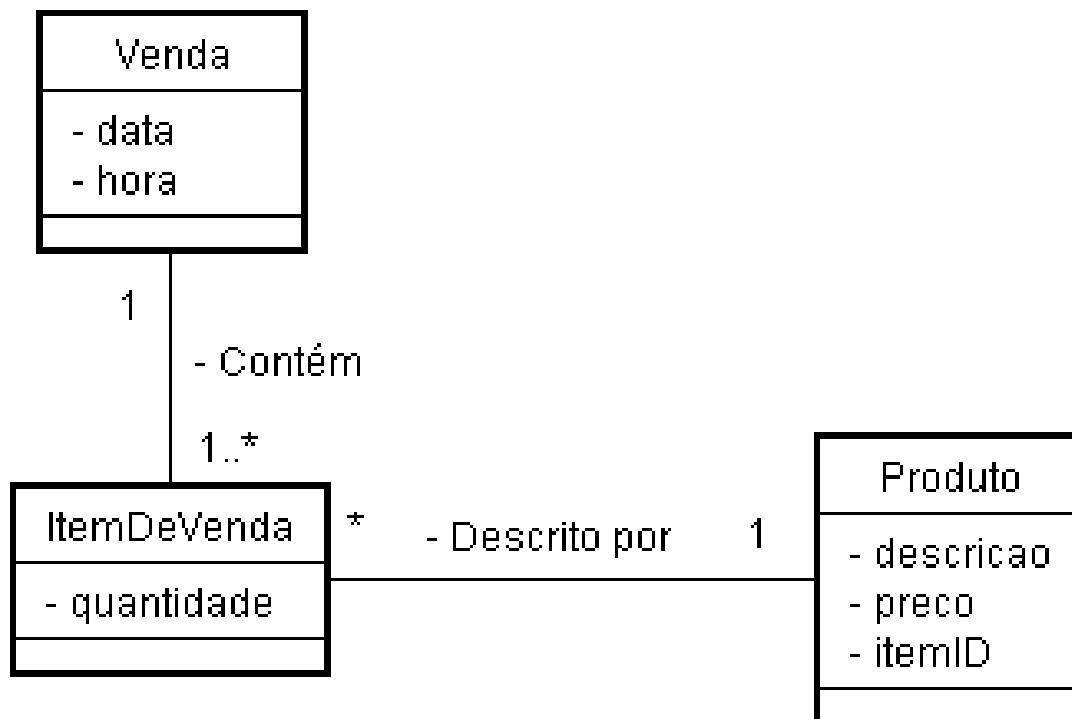
Padrões de Projeto

- Se houver classes relevantes no modelo de projeto, olhe lá primeiro.
- Caso contrário, olhe no modelo de domínio e tente usar suas representações para a criação das classes de projeto correspondentes.
- Considere que estamos apenas iniciando o projeto, por exemplo, e que não há um modelo de projeto mínimo. Devemos olhar o **modelo de domínio na busca de especialistas**; talvez a Venda do mundo real seja um deles. Então, adicionamos uma classe de software ao modelo de projeto chamada Venda, e atribuímos a ela a responsabilidade de conhecer o seu total, portanto ela é a especialista nessa informação.



Padrões de Projeto

- Considere o modelo de domínio.



Padrões de Projeto

- Qual informação é necessária para determinar o total geral da venda? Para isso é preciso conhecer todas as instâncias de ItemDeVenda de uma venda e a soma dos seus subtotais. Uma instância de Venda sabe essa informação. Logo, segundo o padrão especialista, Venda é uma classe adequada para receber essa responsabilidade.
- Como já mencionado, geralmente é durante a criação dos diagramas de **interação que as questões** sobre responsabilidades surgem, como por exemplo:
- Que informação é necessária para determinar o subtotal do item da venda?



Padrões de Projeto

- Para determinar o subtotal é necessário que a quantidade de itens foram vendidos seja conhecida. A classe ItemDeVenda é responsável por essa informação, logo ela é a especialista nessa informação.
- Para satisfazer a responsabilidade de conhecer e informar o subtotal o ItemDeVenda precisa saber o preço do produto. A classe Produto é a **especialista na informação necessária para fornecer esse preço, portanto deve-se** enviar uma mensagem para ela perguntado o seu preço.
- Para satisfazer a responsabilidade de conhecer e informar o total da venda, **três responsabilidades foram atribuídas** para três classes de objetos conforme tabela abaixo.



Padrões de Projeto

Classe do projeto	Responsabilidade
Venda	Sabe o total de venda.
Item de venda	Sabe o subtotal da linha de item.
Produto	Sabe o preço do produto.



Padrões de Projeto

Benefícios

- O encapsulamento de informações é mantido, uma vez que os objetos usam sua própria informação para executar tarefas, favorecendo o acoplamento fraco, que conduz a sistemas mais robustos e fáceis de manter.
- O comportamento está distribuído entre as classes que têm as informações necessárias; assim são estimuladas definições de classes “leves”, de maior coesão mais fáceis de usar.
- Favorece o reuso.



Padrões de Projeto

Contra indicações

- Contra indicado quando aumenta acoplamento e reduz coesão.
 - Ex: Quem é responsável por salvar um Empréstimo no banco de dados?



Padrões de Projeto

2 – Criador (Create)

- A criação de objetos é uma das atividades mais comuns em um sistema orientado a objetos. Desta forma, é útil ter um princípio geral para a atribuição de responsabilidades de criação. Com essas **responsabilidades bem atribuídas**, o projeto apresentará acoplamento fraco, mais clareza, encapsulamento e reutilização.
- Este padrão define qual classe deve ser responsável por criar **instâncias de outras classes**. A classe criadora deve ser aquela que possui a outra como parte dela ou que esteja fortemente associada a ela.



Padrões de Projeto

- O padrão criador guia a atribuição de responsabilidade relacionada com a criação de objetos uma tarefa muito comum. O **objetivo básico do padrão** criador é encontrar um criador que necessite ser conectado ao objeto criado em qualquer evento. Escolhe-lo como o criador garante um acoplamento fraco.



Padrões de Projeto

Problema

- Quem deve ser responsável pela criação de uma nova instância de uma classe?
- A criação de objetos é uma das atividades mais comuns em um sistema orientado a objetos. Consequentemente, é útil ter um princípio geral para a **atribuição de responsabilidades de criação**. Sendo essas responsabilidades bem atribuídas, o projeto apresentará acoplamento fraco, mais clareza, encapsulamento e reutilização.



Padrões de Projeto

Solução

- Atribua à classe B a responsabilidade de criar uma instância da classe A se uma das seguintes condições for verdadeira:
- B agrega objetos de A;
- B contém objetos de A;
- B registra instâncias de objetos de A;
- B usa, de maneira muito próxima, objetos de A;
- B tem os dados de iniciação que serão passados para A quando ele for criado (portanto, B é uma classe especialista com respeito à criação de A);



Padrões de Projeto

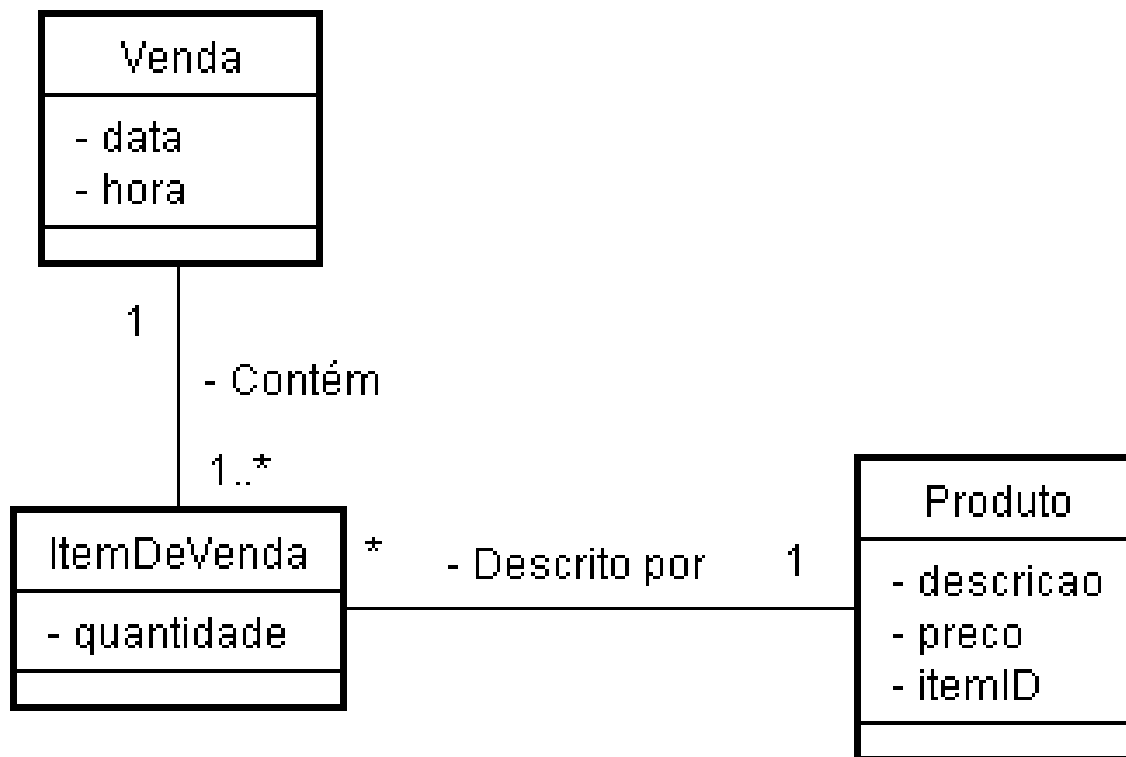
- B é um criador de objetos de A.
- Se mais de uma opção for aplicável, prefira uma classe B que agrega ou contém a classe A.

Exemplo

- No sistema de vendas, quem deve ser responsável por criar uma instância de ItemDeVenda?
- Segundo o padrão criador, devemos procurar uma classe que agregue as instâncias de ItemDeVenda. Considere o modelo abaixo:



Padrões de Projeto



Padrões de Projeto

- Uma vez que uma Venda contém muitos objetos ItemDeVenda, o padrão Criador sugere que Venda seja um bom candidato para ter a responsabilidade pela criação das instâncias de ItemDeVenda.
- Essa **responsabilidade requer que um método** de instânciação de ItemDeVenda seja definido em Venda.
- Mais uma vez essas responsabilidades foram consideradas e atribuídas durante a produção dos diagramas de interação. A **seção de métodos** de um diagrama de classes pode então resumir os resultados da atribuição de responsabilidades, realizadas concretamente como métodos.



Padrões de Projeto

Benefícios

- Favorece o acoplamento fraco, implicando em menor dependência para a manutenção e maiores oportunidades de reutilização. O acoplamento não é aumentado pois a classe criada provavelmente já é visível para a classe criadora, por causa das associações existentes que motivaram sua escolha como criador.



Padrões de Projeto

Contra indicações

- Muitas vezes, a criação é uma tarefa complexa, que exige o uso de instâncias recicladas por motivos de desempenho, eventualmente criando uma instância de uma família de **classes similares com base no valor de alguma propriedade externa**, e assim por diante. Nesses casos, é aconselhável delegar a criação definido pelo padrão Factory Method da GoF .



Padrões de Projeto

3 – Acoplamento Fraco (Low Coupling)

- Acoplamento é a medida de quão fortemente um elemento está conectado a, tem conhecimento de ou depende de outros elementos.
- O acoplamento fraco é um princípio que deve ser levado em conta em **todas as decisões do projeto**, é um objetivo subjacente que deve estar sempre em mente. É um princípio de avaliação que o projetista deve aplicar durante a avaliação das decisões de um projeto.



Padrões de Projeto

- Uma classe com acoplamento fraco significa que ela não depende de muitas outras.
- Uma classe com o acoplamento alto (ou forte) é mais difícil de reutilizar, pois seu uso **depende da reutilização de outras classes** da qual ela depende e também é mais sensível a mudanças nas classes associadas a ela.
- O padrão Acoplamento Fraco favorece o projeto de classes que são mais independentes, o que reduz o impacto de mudanças.



Padrões de Projeto

- Esse padrão não pode ser considerado isolado de outros padrões, como o **Especialista e o Coesão Alta**, mas, em vez disso, precisa ser incluído como um entre vários princípios de projetos que influenciam uma escolha na atribuição de uma responsabilidade.
- Não há medida absoluta que determine quando o acoplamento é muito forte. O desenvolvedor deve avaliar o grau **atual de acoplamento e julgar** se o seu aumento trará problemas. Em geral, as classes que são genéricas por natureza, e com alta probabilidade de reutilização, deveriam ter acoplamento especialmente fraco.



Padrões de Projeto

Problema

- Como favorecer a dependência baixa, o pequeno impacto à mudança e aumentar a reutilização?
- O acoplamento mede o quanto um elemento está conectado a, tem conhecimento de ou depende de outros elementos.
- Um elemento com acoplamento fraco (ou baixo) não **depende de outros elementos** que incluem classes, subsistemas, sistemas, etc.



Padrões de Projeto

- Uma classe com acoplamento forte (ou alto) depende de muitas outras. Essas classes podem ser indesejáveis, pois algumas tem os seguintes problemas:
 - As mudanças em classes relacionadas formam mudanças locais;
 - São difíceis de compreender isoladamente;
 - São de difíceis reutilização, porque isso exige a presença das classes da quais ela depende.



Padrões de Projeto

Solução

- Atribuir responsabilidades de maneira que o acoplamento permaneça fraco.

Exemplo

- Considere o seguinte diagrama de classes parcial do estudo de caso do sistema de vendas.



Padrões de Projeto

- Considere que necessitamos instanciar Pagamento e associá-lo à Venda.
- Que classe deve ser responsável por isso?
- Uma vez que um Registro registra um Pagamento no domínio do mundo real, o **padrão Criador sugere Registro** como um candidato a criar o Pagamento. Então a instância de Registro pode enviar uma mensagem para Venda, passando junto o novo Pagamento como parâmetro.
- Analisaremos duas soluções para o problema:




Padrões de Projeto

Registro cria pagamento.

- Esta atribuição de responsabilidades acopla a classe Registro ao conhecimento da classe Pagamento posteriormente passando pagamento como parâmetro à Venda.

Venda cria pagamento.

- Esta solução alternativa atribui responsabilidades acoplando a classe Venda ao conhecimento da classe Pagamento.
- Qual das soluções de projeto, baseado na atribuição das responsabilidades, sustenta melhor o Acoplamento Fraco? 

Padrões de Projeto

- Em ambos os casos, consideraremos que Venda pode ser acoplada ao conhecimento sobre um Pagamento. No primeiro caso, no qual o Registro cria o Pagamento, acrescenta um acoplamento de Registro a Pagamento, enquanto no segundo caso, no qual a Venda cria um Pagamento, não aumenta o acoplamento. Sob o ponto de vista exclusivo do acoplamento, **o segundo caso é preferível**, pois mantém um acoplamento mais fraco.



Padrões de Projeto

Benefícios

- Atribuir uma responsabilidade de maneira que o acoplamento permaneça fraco traz como benefício:
- O elemento não será afetado por mudanças em outros;
- É simples de entender isoladamente;
- É conveniente para reutilização.



Padrões de Projeto

Contra indicações

- Não deve-se exagerar ao projetar visando o mais baixo acoplamento, essa prática exige bom senso. Aconselha-se que o projetista avalie suas decisões de reduzir o acoplamento e a encapsular coisas concentrando-se nos pontos passíveis de evolução e alta instabilidade.



Padrões de Projeto

4 – Coesão Alta (High Cohesion)

- Coesão é a medida de quão fortemente relacionadas e focalizadas são as responsabilidades de uma classe.
- Assim como o Acoplamento Fraco, a Coesão Alta é um princípio que devemos ter em mente durante todas as decisões do projeto, **é um objetivo subjacente a ser continuamente considerado**. É um padrão de avaliação que o projetista aplica quando avalia todas as decisões do projeto.



Padrões de Projeto

- Uma classe com baixa coesão acaba se responsabilizando por mais coisas do que ela realmente deveria ser responsável. Classes assim acabam oferecendo dificuldade **para o seu entendimento e manutenção**, sendo assim, mais difíceis de reutilizar por não terem o foco definido
- Como regra prática, uma classe com coesão alta tem um número relativamente pequeno de métodos, com funcionalidades **altamente relacionadas**, e não executa muito trabalho. Se a tarefa for grande, ela irá colaborar com outros objetos para dividir o esforço.



Padrões de Projeto

- O nível de coesão não pode ser considerado isoladamente de outras responsabilidades e de outros princípios, como o **Especialista** e o **Acoplamento Fraco**. Má coesão geralmente traz mau acoplamento, e vice-versa. São interdependentes..



Padrões de Projeto

Problema

- Como manter a complexidade sob controle?
- Em termos de projeto orientado a objetos, a coesão mede o quanto as responsabilidades de um elemento são fortemente relacionadas. Um elemento com **responsabilidades altamente relacionadas** e que não executa um grande volume de trabalho, tem uma alta coesão. Esses elementos incluem classes, subsistemas e outros.



Padrões de Projeto

- Uma classe com baixa coesão faz muitas coisas não relacionadas ou executa muitas tarefas. Tais classes são indesejáveis, pois sofrem dos seguintes problemas:
- Difíceis de compreender
- Difíceis de reutilizar
- Difíceis de manter
- Delicadas, constantemente afetadas pelas mudanças.
- As classes de coesão baixa geralmente representam um grau de abstração muito alto e de grande granularidade, ou então **assumiram responsabilidades** que deveriam ter sido delegadas a outros objetos.



Padrões de Projeto

Solução

- Atribuir uma responsabilidade de forma que a coesão permaneça alta.

Exemplo

- O mesmo exemplo usado no padrão Acoplamento Fraco pode ser usado para análise da Coesão Alta.
- Considere que necessitamos criar uma instância de Pagamento (em dinheiro) e associá-la à Venda.
- Que classe deve ser responsável por isso?

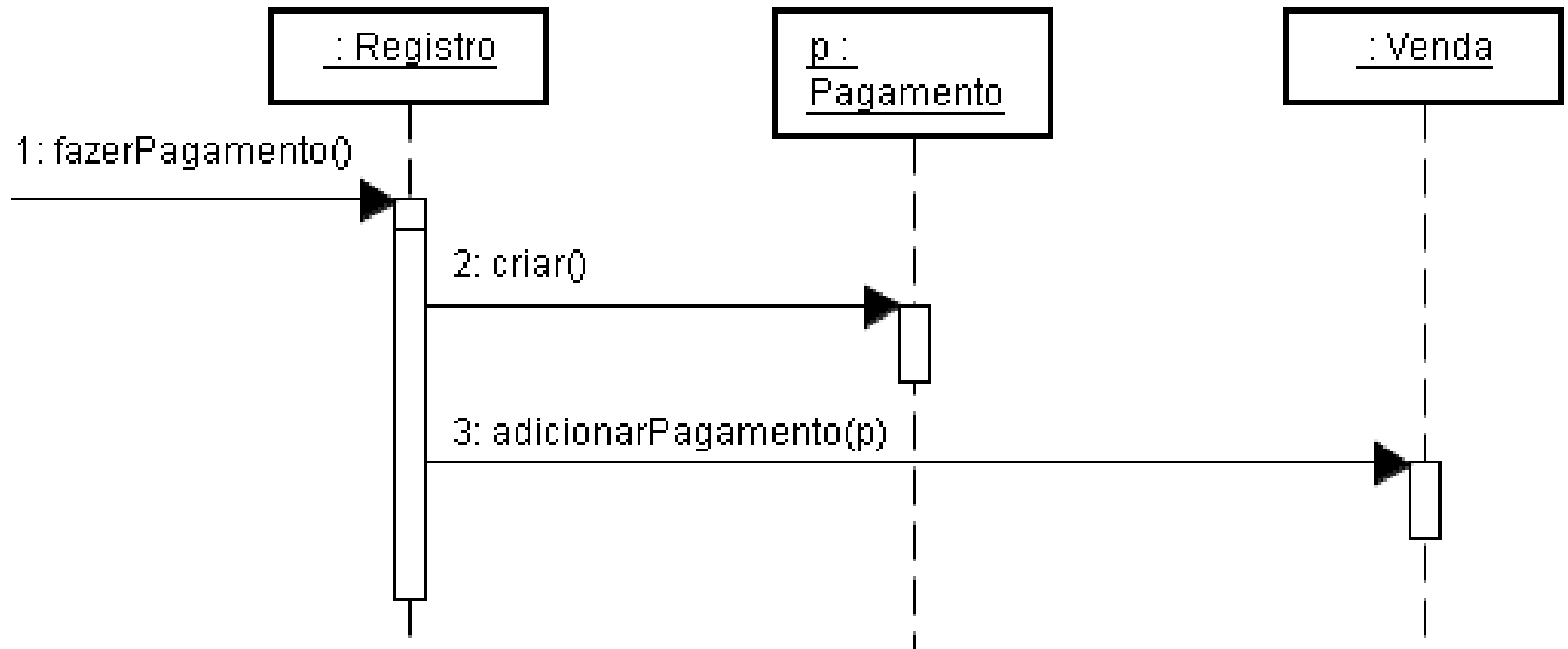


Padrões de Projeto

- Uma vez que Registro registra um Pagamento no domínio do mundo real, o padrão Criador sugere Registro como um candidato para criar o Pagamento. A **instância de Registro deve** então mandar uma mensagem para Venda, passando como um parâmetro o novo Pagamento.
- O modelo abaixo (Registro cria Pagamento) ilustra o que acabamos de definir.



Padrões de Projeto



Padrões de Projeto

- Essa atribuição entrega a responsabilidade de criar um pagamento à Registro. Este toma parte na responsabilidade de executar a operação do sistema fazerPagamento.
- Neste exemplo isolado, isso é aceitável. Entretanto, se continuarmos a torna a **classe Registro responsável** por tomar parte, ainda que em pequena escala, do trabalho relacionado com mais operações do sistema, ele ficará progressivamente sobrecarregado de tarefas e perderá sua coesão.



Padrões de Projeto

- Imagine que existam 70 operações no sistema, todas recebidas por Registro. Se ela fizesse o trabalho relacionado com cada uma, se tornaria um objeto inchado, sem coesão.
- O problema não está no fato de que essa tarefa de criação de pagamento **torna o Registro sem coesão**. Entretanto, como parte de um quadro de atribuição de responsabilidade mais amplo, ele pode sugerir uma tendência para uma coesão baixa.

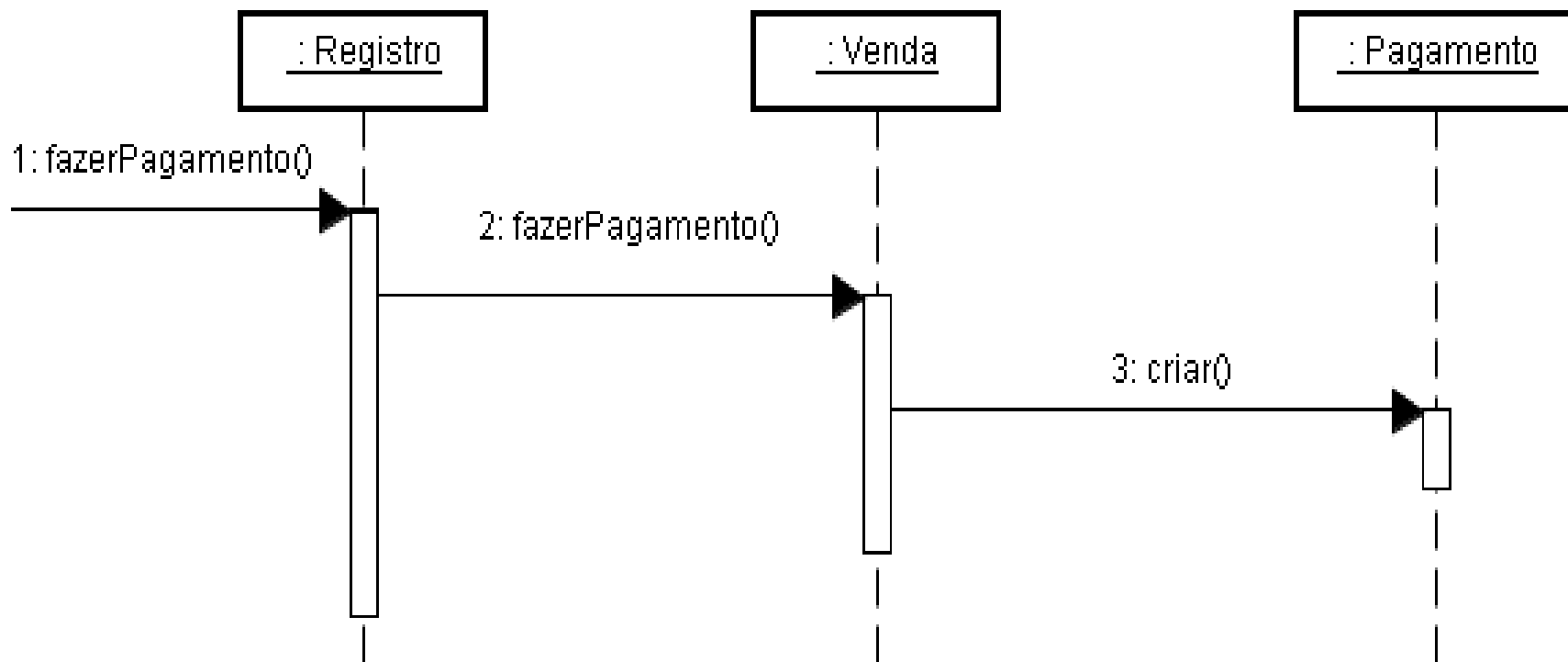


Padrões de Projeto

- E, mais importante em termos de desenvolvimento de habilidades de projetista de objetos, independentemente da escolha final, é **que o desenvolvedor** saiba considerar o impacto sobre a coesão.
- O próximo modelo (venda cria Pagamento) delega a criação do pagamento para a Venda o que favorece uma coesão mais alta no Registro. Uma vez que este modelo favorece uma alta **coesão como um acoplamento fraco** ele é preferível.



Padrões de Projeto



Padrões de Projeto

Benefícios

- Atribuir uma responsabilidade de maneira que a coesão permaneça alta traz como benefício:
- Mais clareza e facilidade de compreensão no projeto;
- Simplificação da manutenção e do acréscimo de melhorias;
- Favorecimento do acoplamento fraco.



Padrões de Projeto

Contra indicações

- Existem alguns poucos casos nos quais se justifica uma coesão mais baixa. Um deles é o agrupamento de responsabilidades ou de código em uma classe ou componente para simplificar a manutenção, claro que tal agrupamento também pode piorar a manutenção dependendo da forma que for executado.



Padrões de Projeto

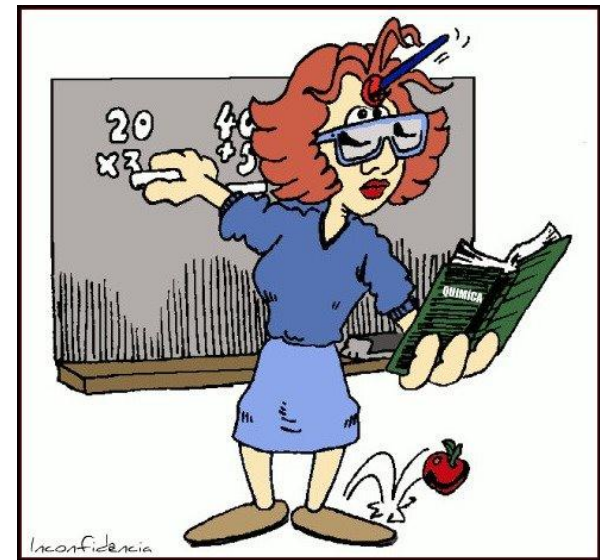
- Quando existem **objetos servidores distribuídos**. Por causa de sobrecarga e das implicações de desempenho associadas com objetos e comunicações remotas, algumas vezes, é melhor criar uma quantidade menor de objetos servidores maiores, menos coesos, que forneçam uma interface para muitas operações.



Padrões de Projeto



**Não durma
no ponto.**



**Exercício de
fixação.**

