Padrões GRASP

- O que são padrões?
  - Padrões de análise;
  - Padrões de projeto;
  - Padrões de processo;
- GRASP (*General Responsibility Assignment Software Patterns*)
  - Atribuição de responsabilidade a objetos
- Em um diagrama de colaboração... como distribuir as responsabilidades?

Padrões GRASP

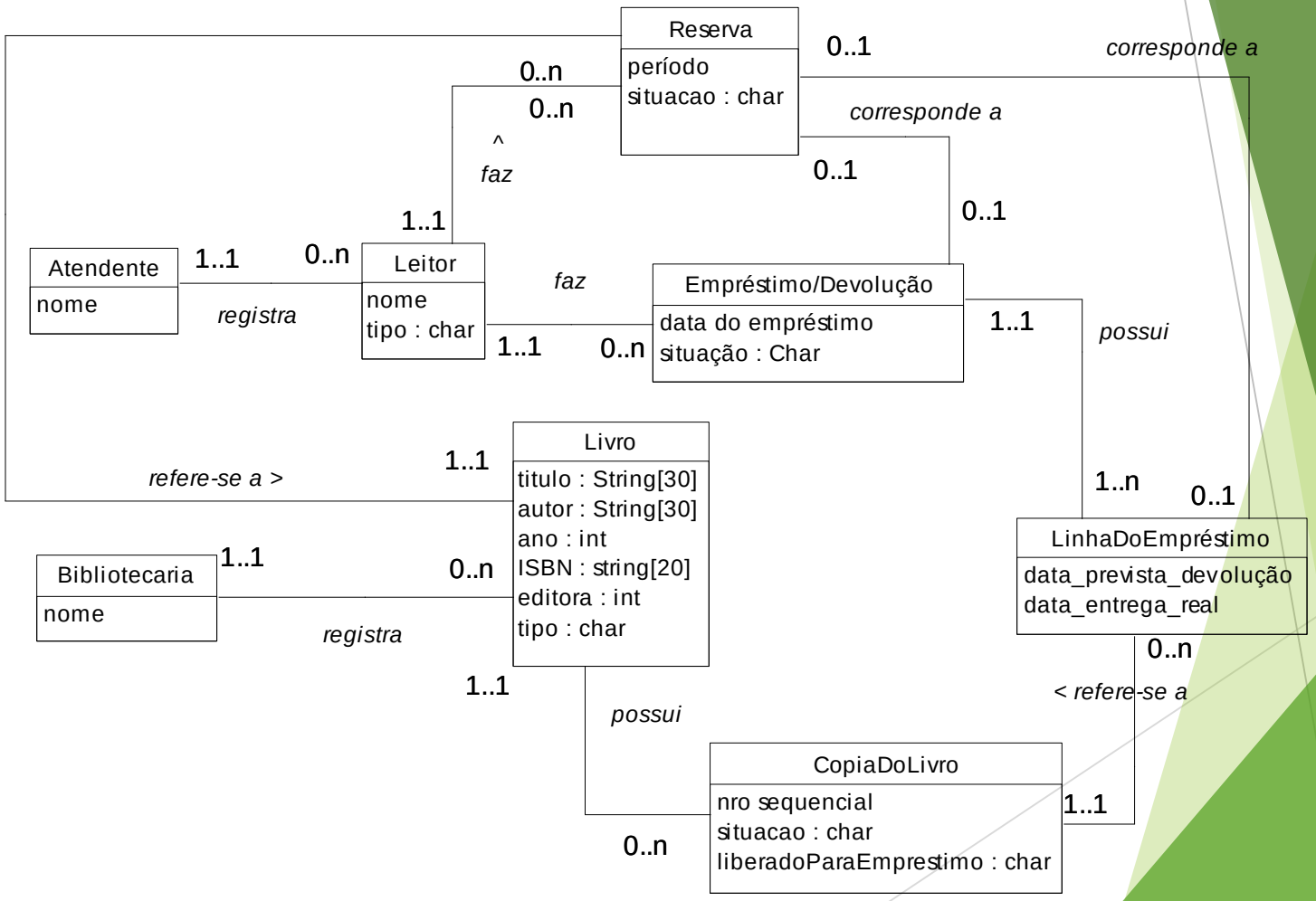
- Principais
  - Especialista (Expert)
  - Criador (Creator)
  - Acoplamento Fraco (Low Coupling)
  - Controlador (Controller)

Padrão Especialista

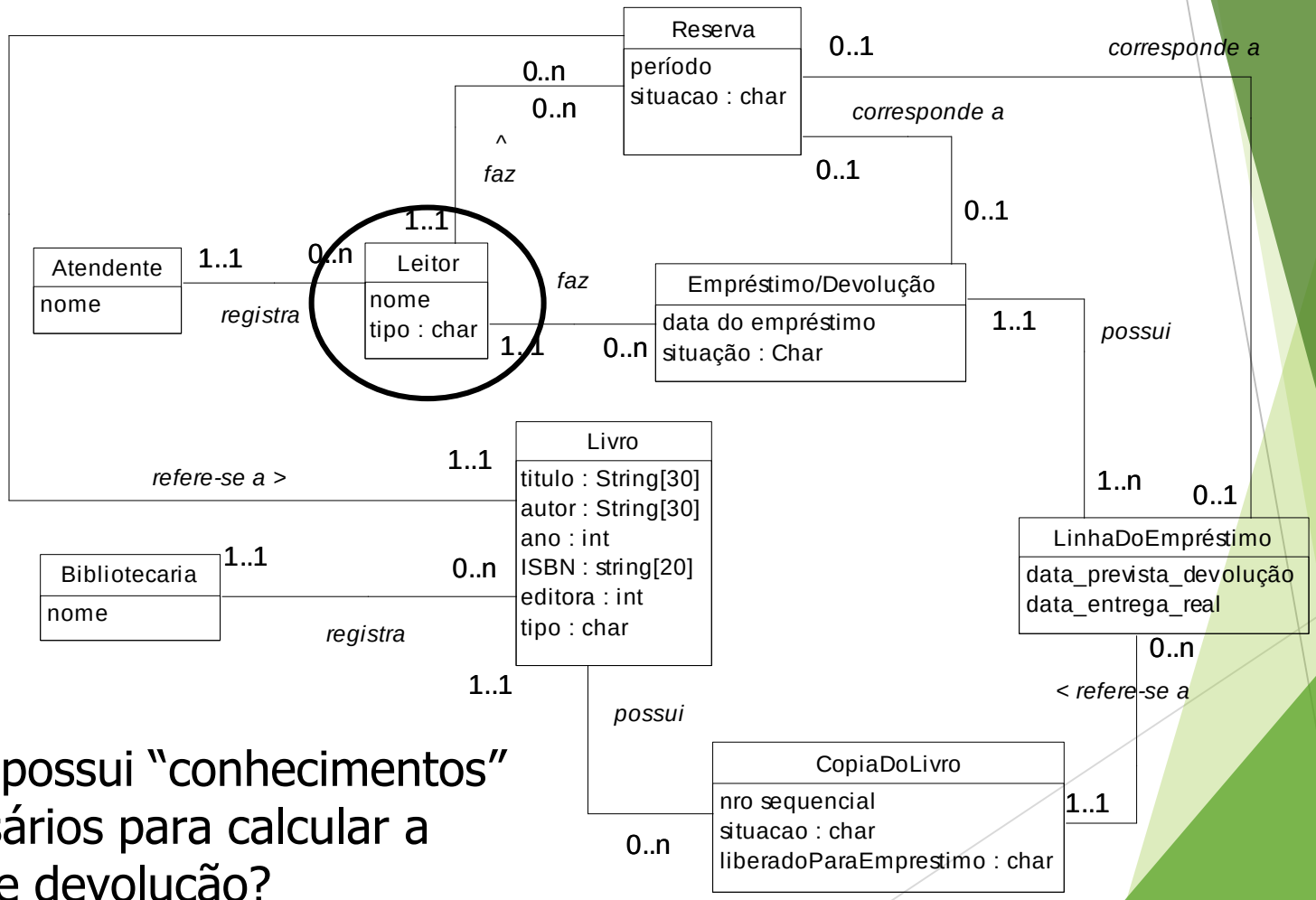
- É o primeiro a ser pensado.

- **Problema**: qual é o princípio mais básico de atribuição de responsabilidades a objetos ?
- **Solução**: Atribuir responsabilidade ao **especialista** da informação.
- **Exemplo**: no sistema de biblioteca (a seguir), a data de devolução de um livro é calculada de acordo da seguinte forma: 7 dias para alunos e 12 dias para docentes. Qual classe seria a responsável por **calcular a data de devolução** de um livro?

Padrão Especialista

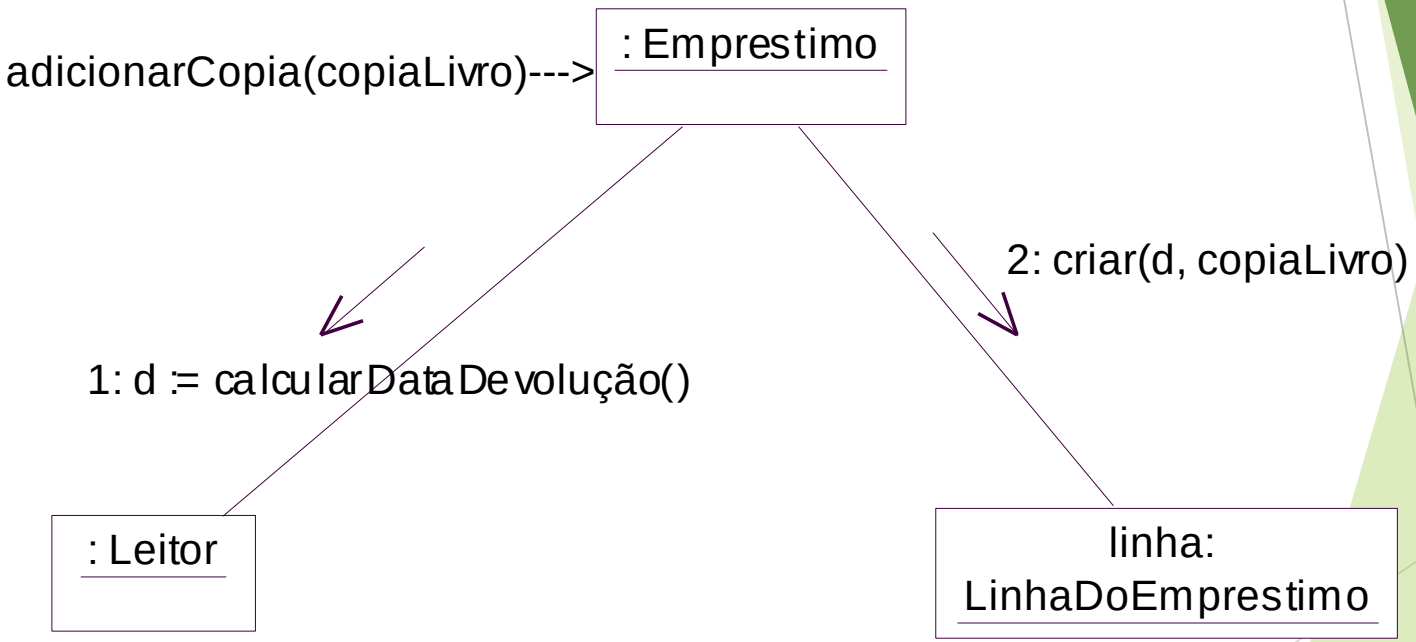


Padrão Especialista



Quem possui "conhecimentos" necessários para calcular a data de devolução?

Padrão Especialista



Padrão Especialista

- Usando-se o padrão Especialista, consegue-se manter o encapsulamento, pois cada classe faz o que realmente tem **conhecimento** para fazer
- Favorece-se o acoplamento fraco e a alta coesão
- O comportamento fica distribuído entre as classes que têm a **informação** necessária, tornando as classes mais “leves”
- O reuso é favorecido, pois ao reutilizar uma classe sabe-se que ela oferece todo o comportamento inerente e esperado

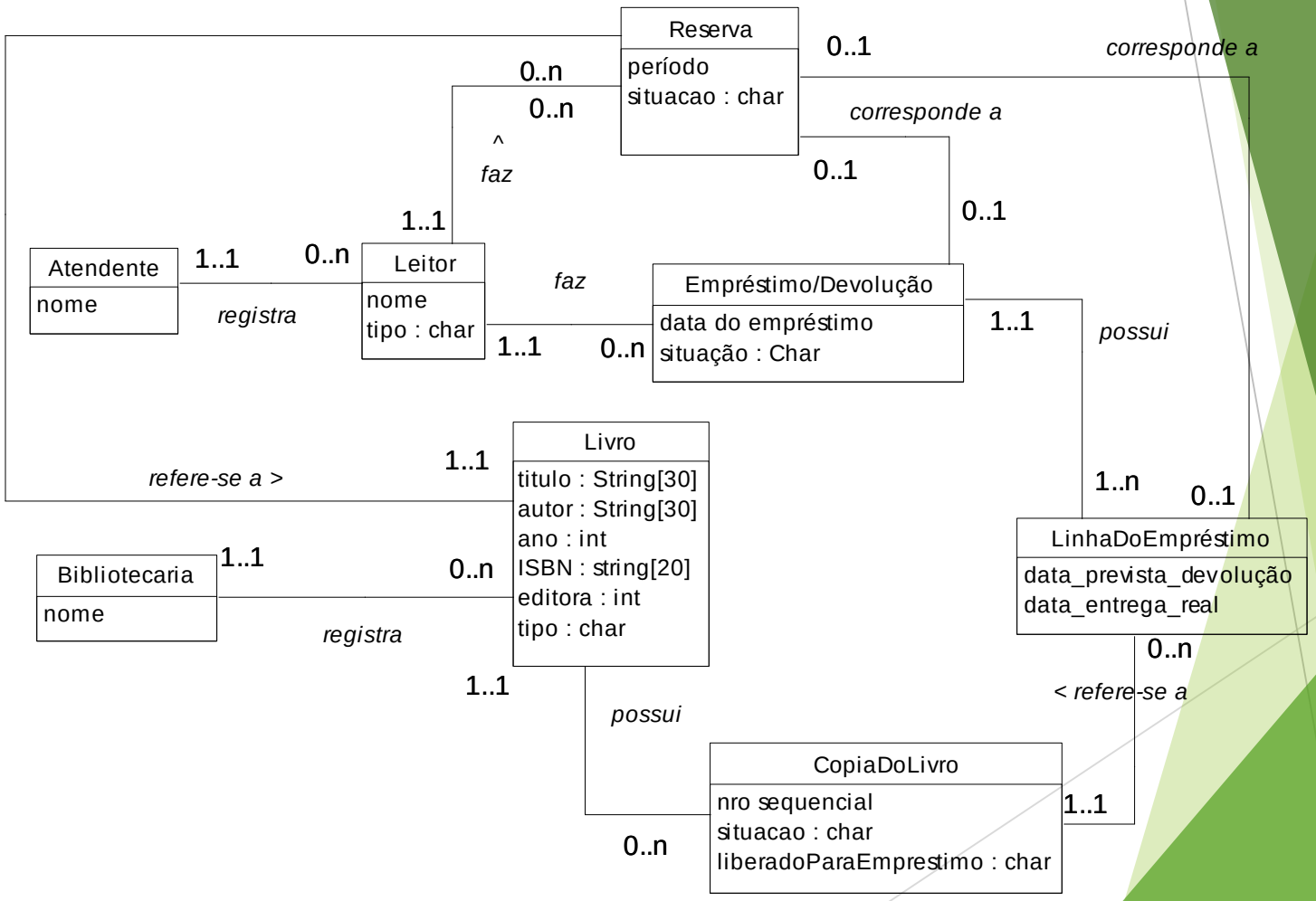
Padrão Criador

- **Problema**: Quem deveria ser responsável pela criação de uma nova instância de alguma classe ?

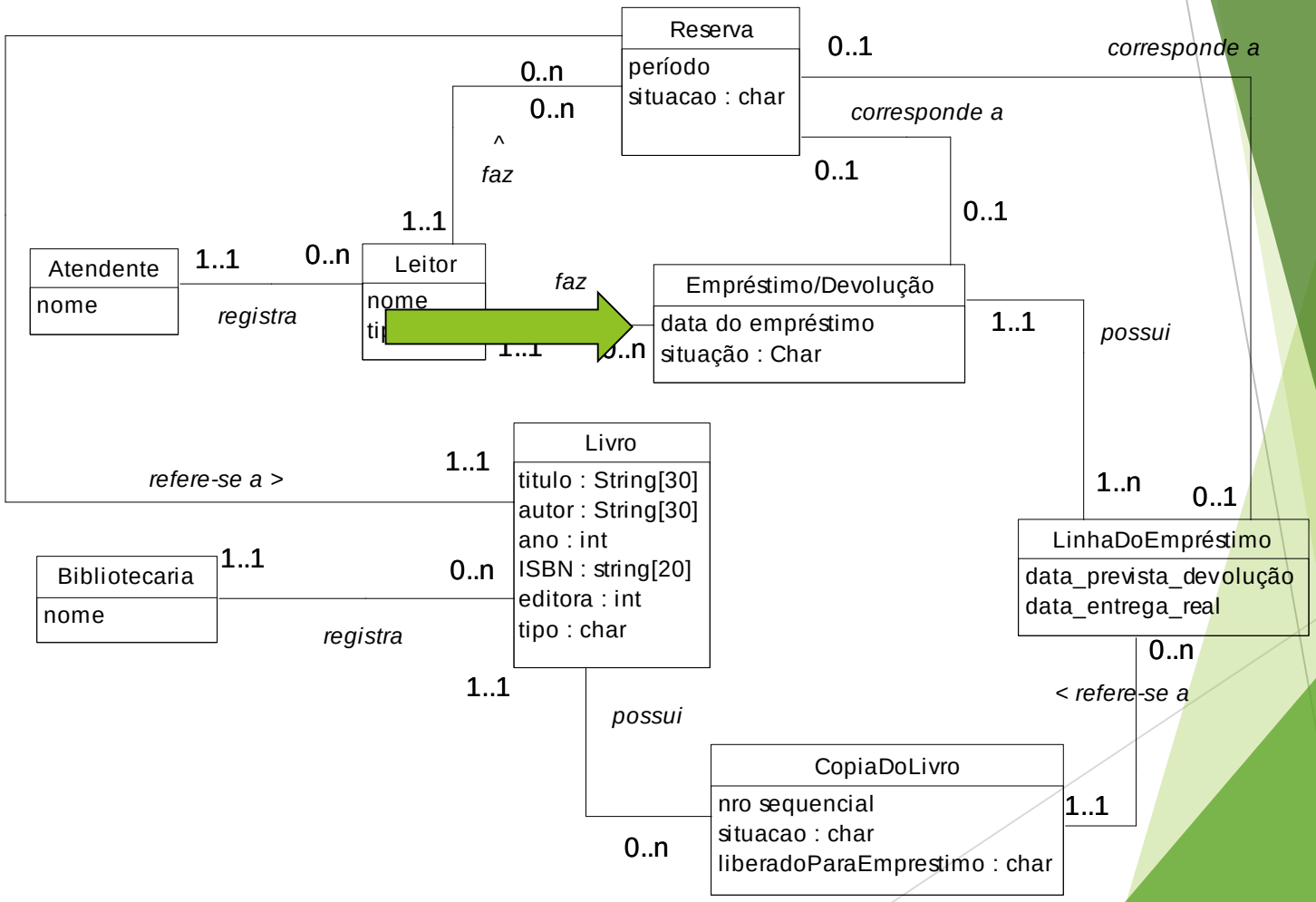
- **Solução**: atribua à classe B a responsabilidade de criar uma nova instância da classe A se uma das seguintes condições for verdadeira:
- B agrega objetos de A
- B registra objetos de A
- B usa objetos de A

- **Exemplo**: No sistema da Biblioteca, quem é responsável pela criação de uma LinhaDoEmprestimo ?

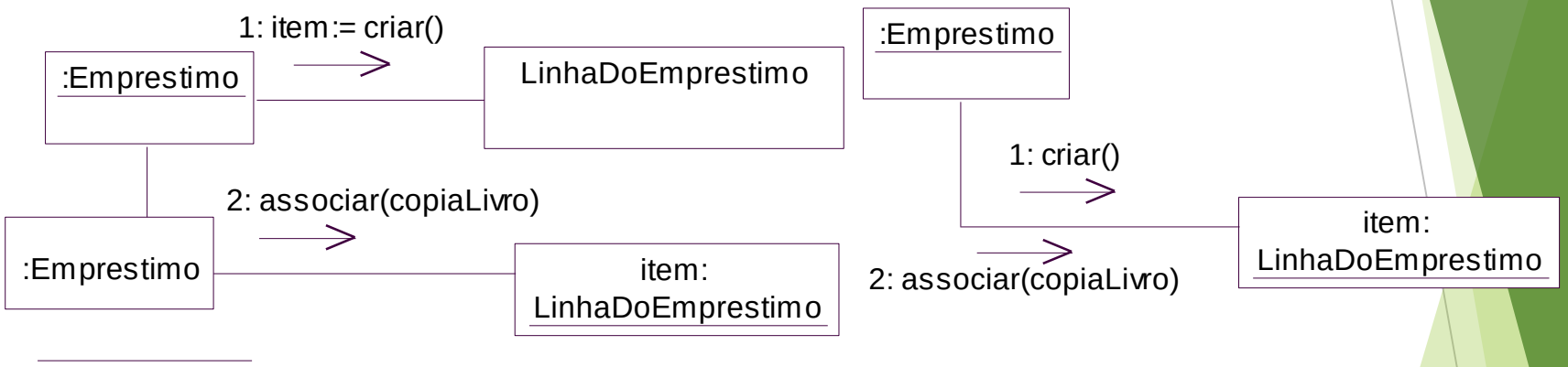
Padrão Criador



Padrão Criador



Padrão Criador



(a) Cria e depois usa o objeto

(b) cria e já usa o próprio objeto

Simplificação para a criação de objetos.

Padrão Acoplamento Fraco

- Acoplamento é a dependência entre elementos (por exemplo, classes ou subsistemas).
  - Associações e dependências
- Em geral, o acoplamento resulta da colaboração entre esses elementos para atender a uma responsabilidade.
- Na OO o acoplamento mede o quanto um objeto está conectado a, tem conhecimento de, ou depende de outros objetos.
- Pode-se dizer que o acoplamento é fraco (ou baixo) se um objeto não depende de muitos outros e que o acoplamento é forte (ou alto) se um objeto depende de muitos outros.

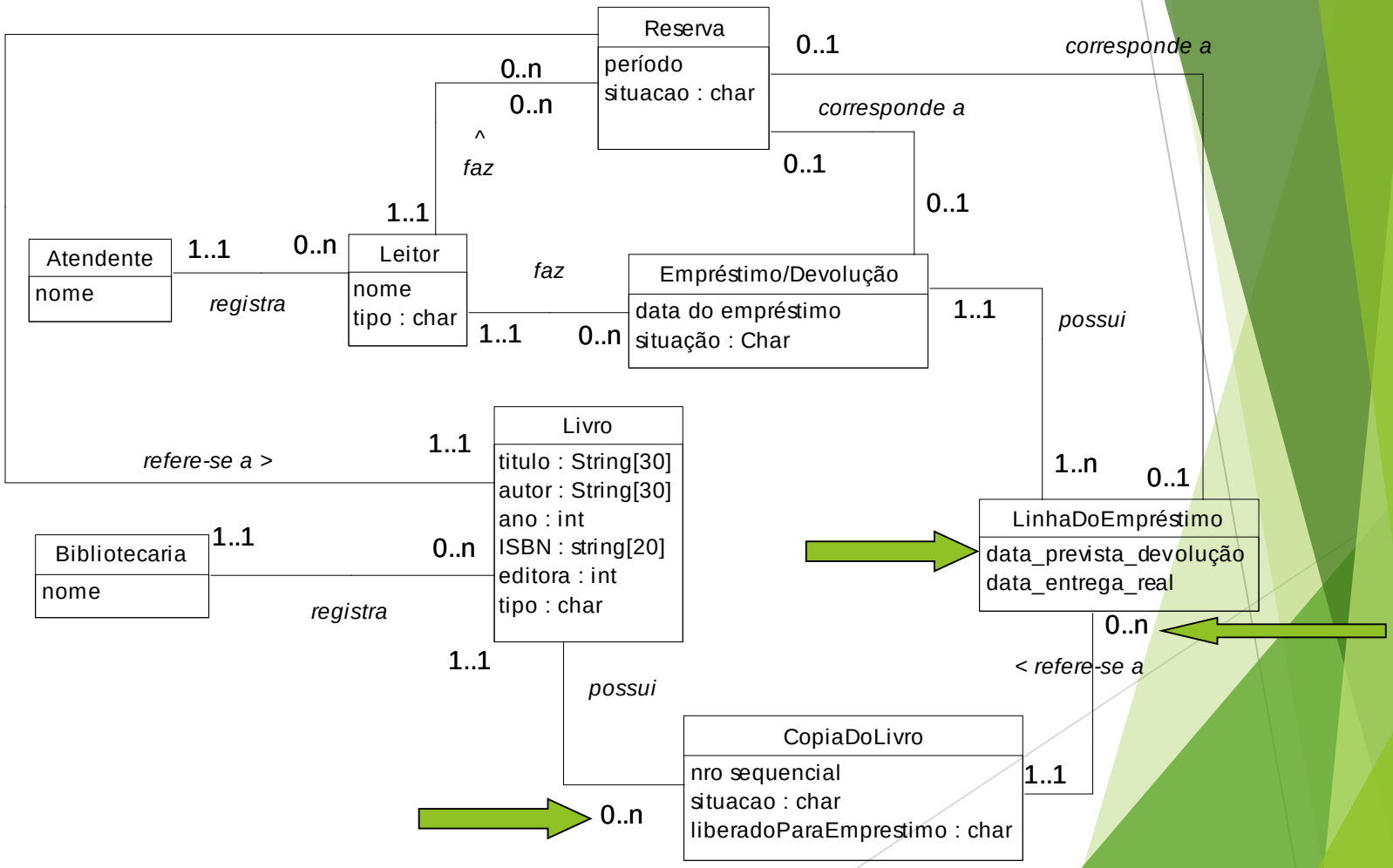
Padrão Acoplamento Fraco

- O acoplamento alto pode causar vários problemas:
  - Mudanças em classes interdependentes forçam mudanças locais, ou seja, se uma classe A está acoplada às classes B e C, mudanças em B e C podem exigir que A seja modificada para preservar seu comportamento.
  - Quando uma classe está conectada a muitas outras, para entendê-la é necessário entender também essas outras, o que dificulta a compreensão do objetivo de cada classe.
  - Dificuldade em reutilizar a classe, pois todas as classes acopladas também precisam ser incorporadas para reuso.

Padrão Acoplamento Fraco

- **Problema**: como favorecer a baixa dependência e aumentar a reutilização ?
- **Solução**: Atribuir responsabilidades de maneira que o acoplamento permaneça baixo.
- **Exemplo**: No sistema de biblioteca, suponha que queremos **realizar a devolução da cópia do livro**. Que classe deve ser responsável por essa tarefa?
- Analise as alternativas a seguir:

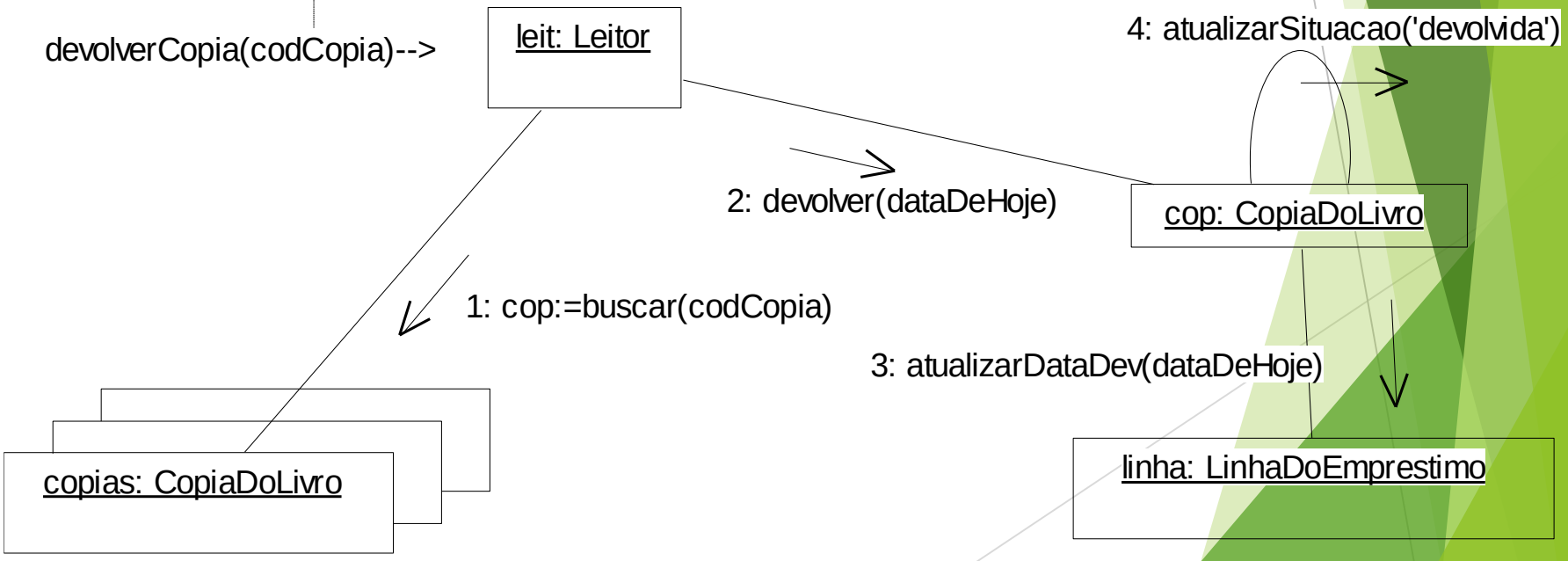
Padrão Acoplamento Fraco



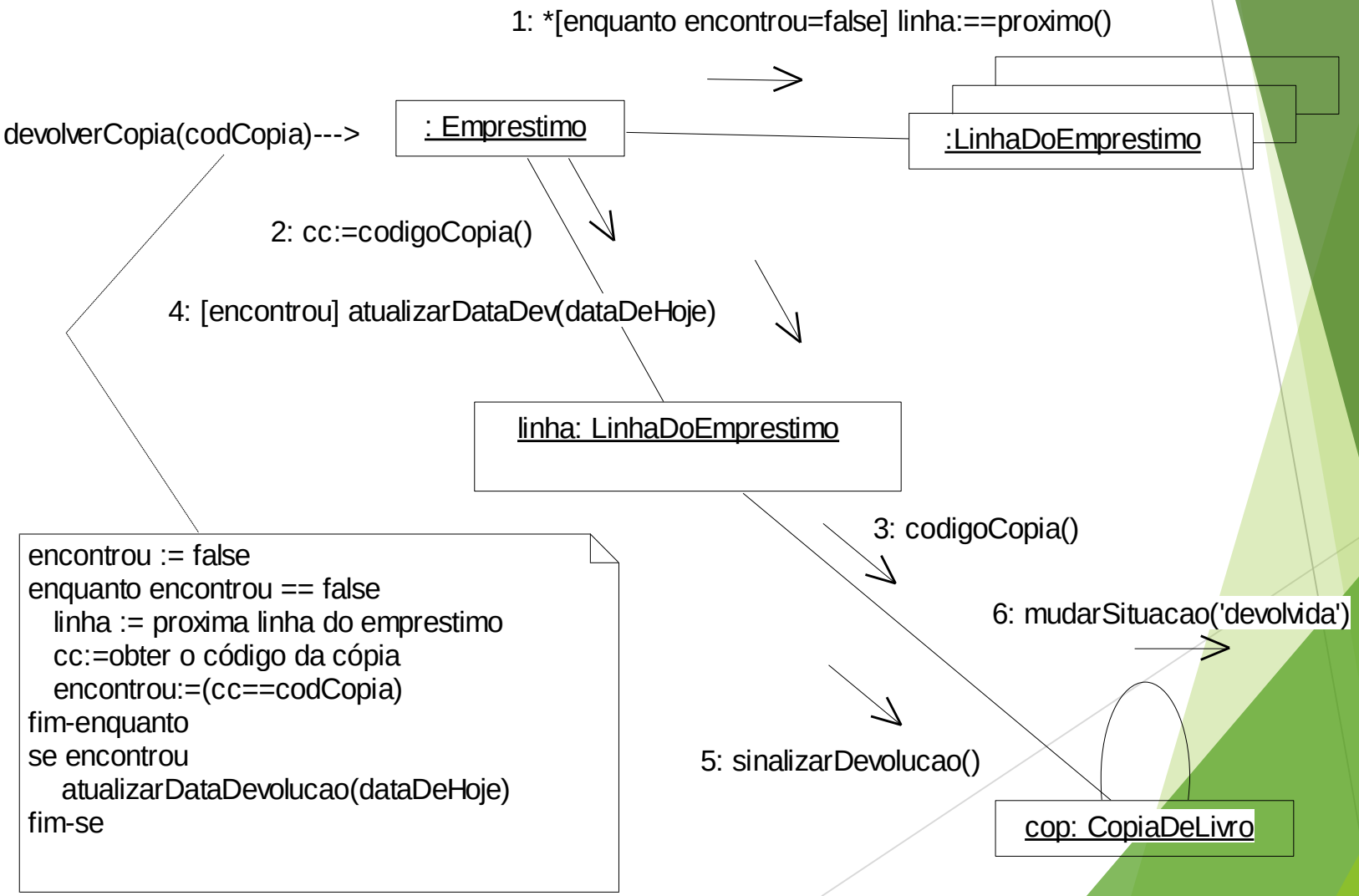
Padrão Acoplamento Fraco
Projeto 1

cop:=copias.buscar(codCopia)
cop.devolver(dataDeHoje)

Essa solução aumenta o acoplamento inicialmente concebido no Modelo Conceitual pois inclui uma associação entre Leitor e CopiaDoLivro.
Além disso, a mensagem enviada de CopiaDoLivro para LinhaDoEmprestimo, que havia sido concebida no sentido inverso (de LinhaDoEmprestimo para CopiaDoLivro), também aumenta o acoplamento.



Padrão Acoplamento Fraco
Projeto 2

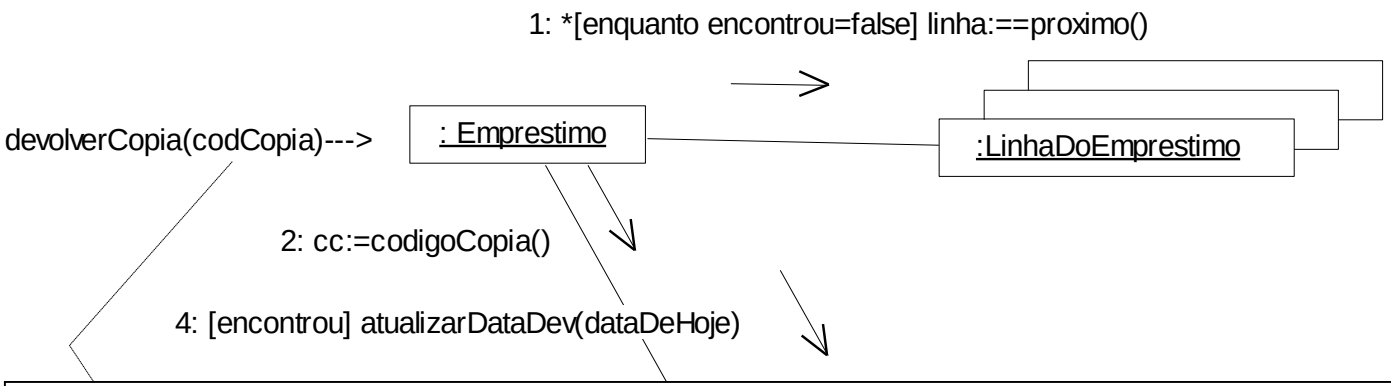


Padrão Acoplamento Fraco

- O extremo de acoplamento fraco não é desejável, por ferir os princípios da tecnologia de objetos, que é o de comunicação por mensagens.
- Se uma classe não se comunica com outras, ela acaba ficando com excesso de responsabilidades (ver padrão Coesão Alta), o que também é indesejável. Isso leva a projetos pobres: objetos inchados e complexos, responsáveis por muito trabalho.

Padrão Controlador

Ainda tem algum problema?



quando ocorre o evento de devolução da cópia, ainda não é conhecido o objeto **empréstimo** ao qual a cópia emprestada se refere. Portanto, é preciso eleger alguma classe, que conheça os empréstimos, para receber a mensagem **devolverCopia**. Essa classe terá que identificar o objeto empréstimo cujo código de cópia seja igual ao parâmetro fornecido

cc:=obter o código da cópia

Essa necessidade surge porque **devolverCopia** é uma operação do sistema, responsável por tratar os eventos que ocorrem em cada caso de uso

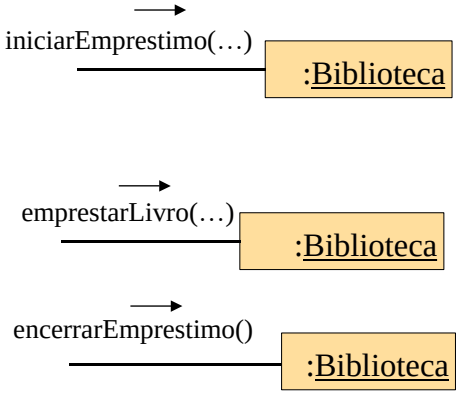
Padrão Controlador

- **Problema**: Quem deve ser responsável por tratar um evento do sistema ?

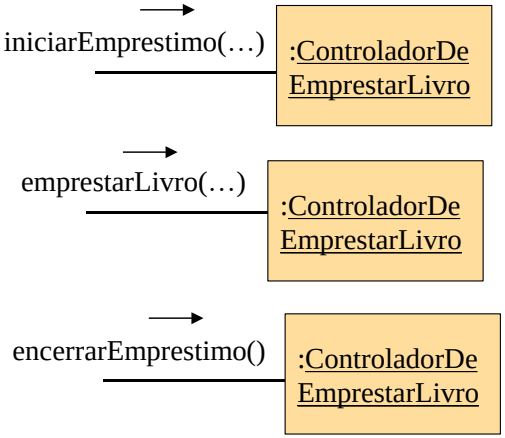
- **Solução**: A responsabilidade de receber ou tratar as mensagens de eventos (operações) do sistema pode ser atribuída a uma classe que:
  - represente todo o sistema, um dispositivo ou um subsistema – chamado de **controlador fachada** - OU
  - represente um cenário de um caso de uso dentro do qual ocorra o evento, chamado de **controlador artificial**, por exemplo um TratadorDe<NomeDoCasoDeUso> ou ControladorDe<NomeDoCasoDeUso>

- **Exemplo**: quem vai tratar os eventos do sistema de biblioteca?

Padrão Controlador

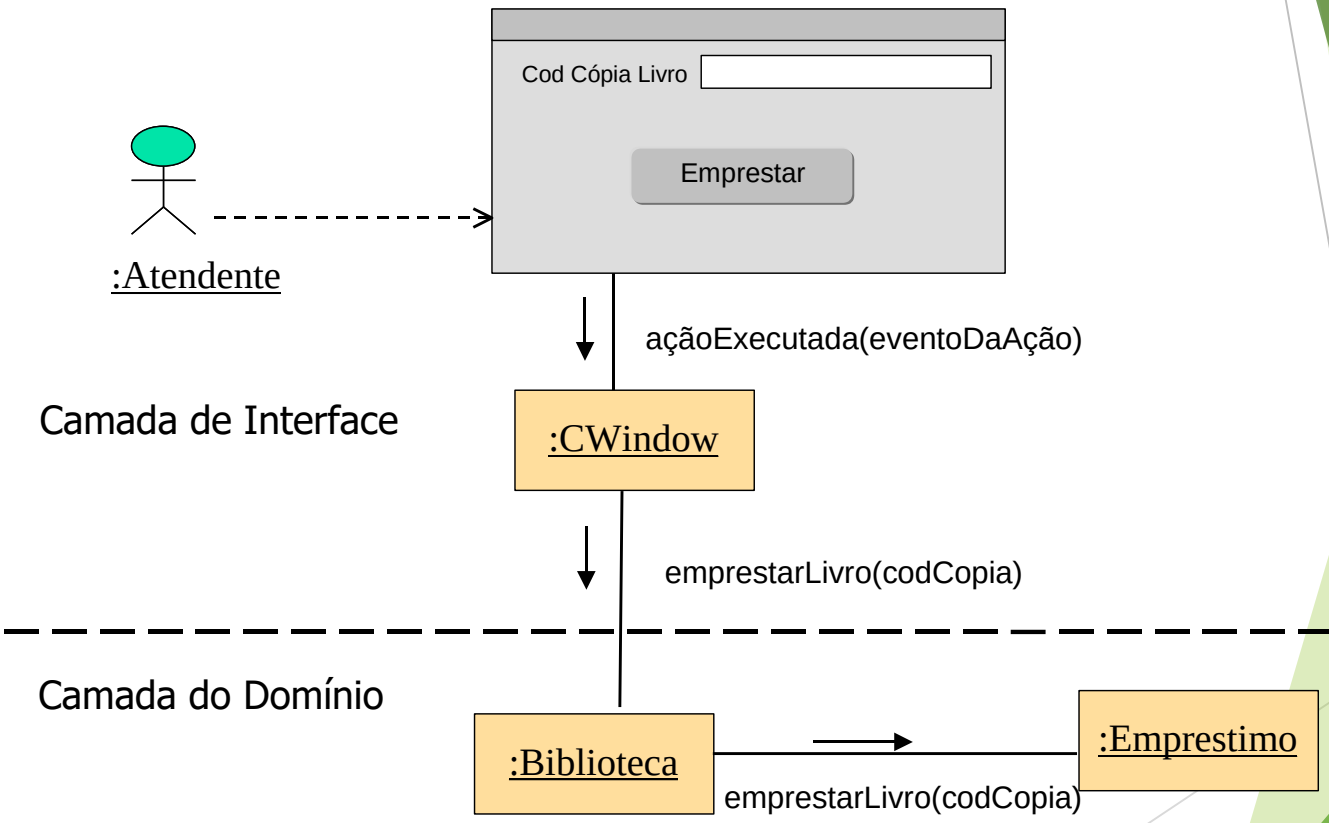


(a) Controlador Fachada



(b) Controlador Artificial

Padrão Controlador



Padrão Controlador

- **Benefícios:**
  - Com o uso do padrão Controlador, obtem-se um aumento das possibilidades de reutilização de classes e do uso de interfaces “plugáveis”. Além disso, pode-se usufruir do conhecimento do estado do caso de uso, já que o controlador pode armazenar o estado do caso de uso, garantindo a sequência correta de execução das operações.

Referências Bibliográficas

- Utilizando UML e Padrões – Craig Larman, Bookman Editora, 2003 (tradução)
- Guia de Consulta Rápida UML – Douglas Marcos da Silva, Novatec Editora, 2001
- UML – Booch, Rumbaugh, Jacobson, Editora Campus, 1999.
- Desenvolvendo Software Com UML 2.0 – Ernani Medeiros, Pearson, 2004