Introdução aos Sistemas de Informação

Padrões GRASP

Prof. Dr. Valter Camargo

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

- Projeto Guiado por Responsabilidade – PGR
  - Responsabilidades estão relacionadas com as obrigações de um objeto em termos de seu papel
  - Responsabilidades de dois tipos:
    - Fazer
    - Conhecer

Padrões GRASP

- Fazer
  - Fazer algo propriamente dito, com criar objetos ou executar um cálculo
  - Iniciar uma ação entre objetos
  - Controlar e coordenar atividades em outros objetos
- Saber
  - Ter conhecimento sobre dados privados encapsulados
  - Conhecer objetos relacionados
  - Ter conhecimento sobre coisas que ele pode derivar ou calcular

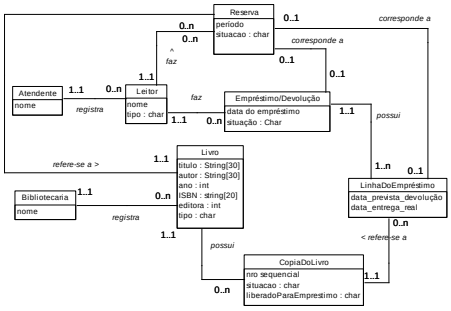
Padrões GRASP

- Principais
  - Especialista (Expert)
  - Criador (Creator)
  - Coesão Alta (High Cohesion)
  - Acoplamento Fraco (Low Coupling)
  - Controlador (Controller)

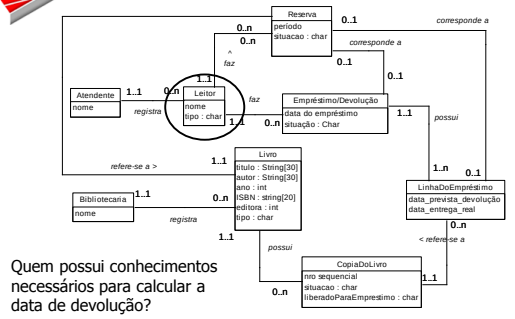
Padrão Especialista

- 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 (a classe que possui os atributos que detém o conhecimento necessário).
- **Exemplo**: no sistema de biblioteca (a seguir), quem seria o responsável por **calcular a data de devolução** de um livro? (onde colocar o método que calcula a data de devolução?)

Padrão Especialista

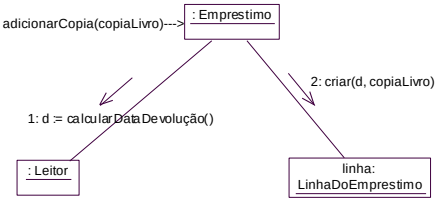


Padrão Especialista



Quem possui conhecimentos necessários para calcular a data de devolução?

Padrão Especialista



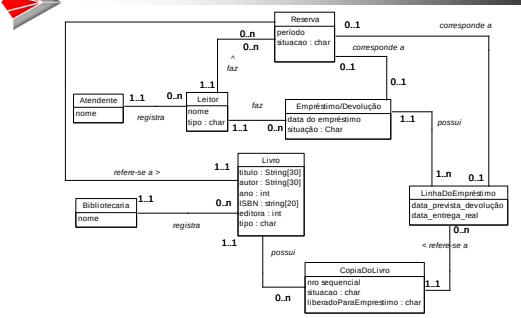
Padrão Especialista

- 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

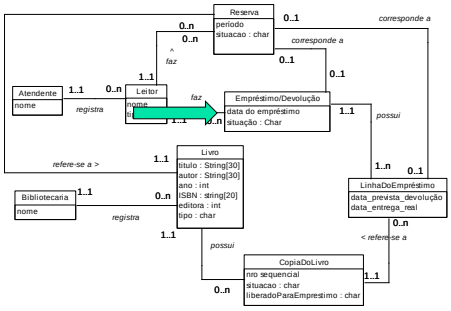
Padrão Criador

- **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?

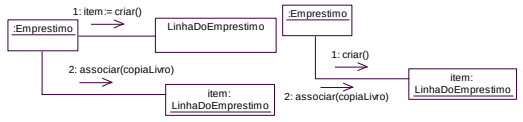
Padrão Criador



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.

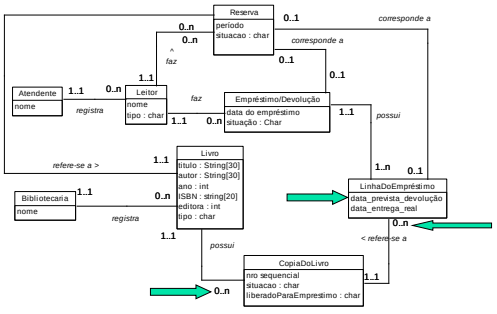
Padrão Acoplamento Fraco

- 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.

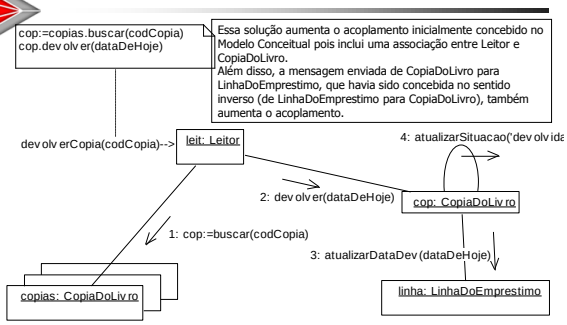
Padrão Acoplamento Fraco

- **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:

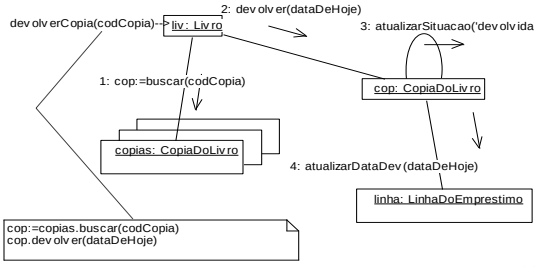
Padrão Acoplamento Fraco



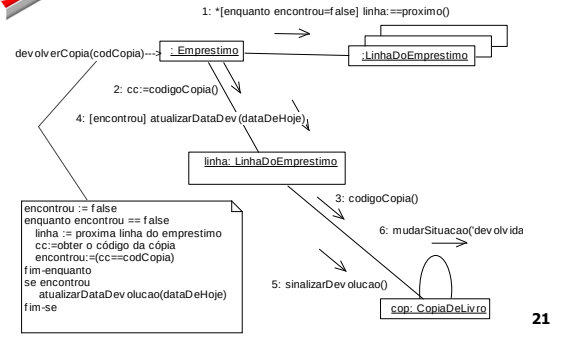
Padrão Acoplamento Fraco
Projeto 1



Padrão Acoplamento Fraco
Projeto 2



Padrão Acoplamento Fraco
Projeto 3

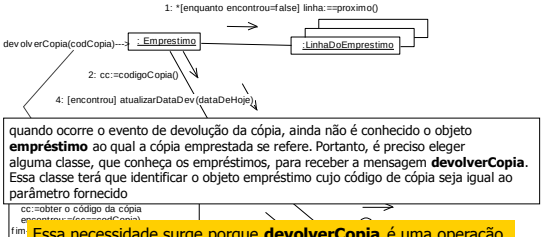


Padrão Acoplamento Fraco

- 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?

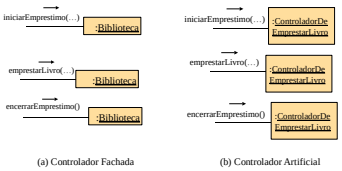


Essa necessidade surge porque **devolverCopia** é uma operação do sistema, responsável por tratar os eventos que ocorrem em cada caso de uso

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?

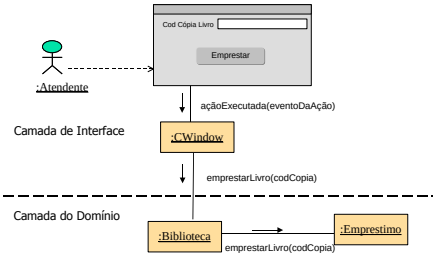
Padrão Controlador



(a) Controlador Fachada (b) Controlador Artificial

25

Padrão Controlador



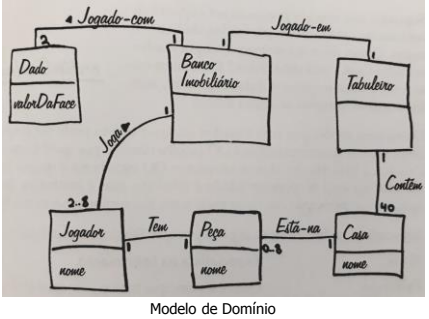
26

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.

27

Discussão/Exemplo Banco Imobiliário

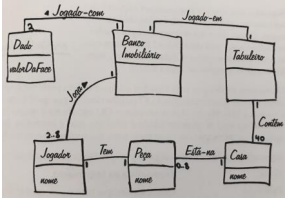


Modelo de Domínio

28

Padrão Criador

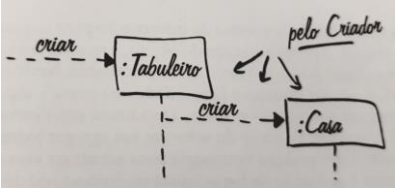
- Problema: quem cria objetos do tipo Casa?
  - Observe que qualquer objeto pode criar uma Casa, mas qual seria a escolha da maioria dos desenvolvedores OO? Por quê?



29

Padrão Criador

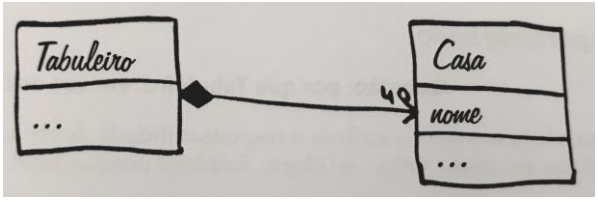
- Problema: quem cria objetos do tipo Casa?



30

Padrão Criador

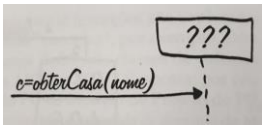
- Evolução do Diagrama de Classe de Projeto



31

Padrão Especialista

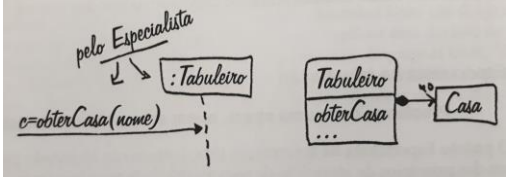
- Problema: dada uma chave, quem sabe sobre um objeto Casa?
  - Suponha que objetos precisem referenciar uma Casa em particular, dado o seu nome. Quem deve ser o responsável por conhecer uma Casa, dada uma chave?



32

Padrão Especialista

- Problema: dada uma chave, quem sabe sobre um objeto Casa?



33

Padrão Acoplamento Baixo

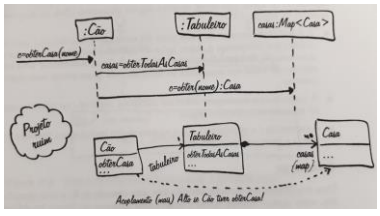
- Questão: por que Tabuleiro em vez de Cão?
  - Observe que qualquer classe poderia oferecer informações sobre uma Casa particular. "Cão" aqui representa uma classe arbitrária qualquer.

34

Padrão Acoplamento Baixo

- Questão: por que Tabuleiro em vez de Cão?
  - Observe que qualquer classe poderia oferecer informações sobre uma Casa particular. "Cão" aqui representa uma classe arbitrária qualquer.

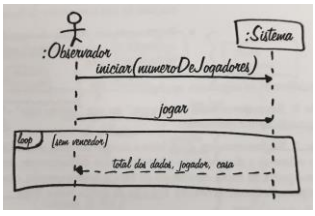
Qual o problema dessa solução?



35

Padrão Controlador

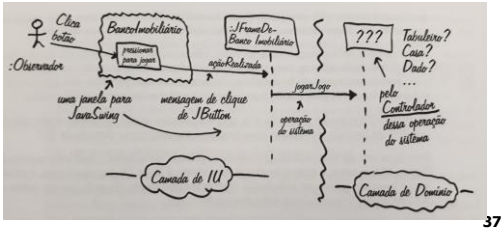
- Questão: qual o primeiro objeto , depois ou além da camada de IU, que deve receber a mensagem da camada de IU?



36

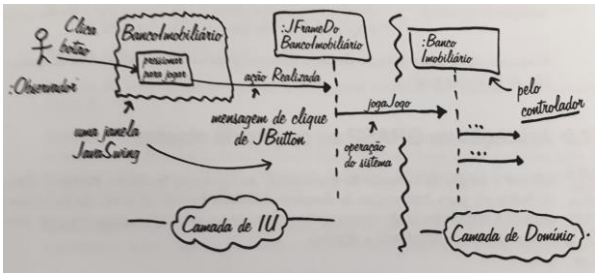
Padrão Controlador

- Questão: qual o primeiro objeto , depois ou além da camada de IU, que deve receber a mensagem da camada de IU?



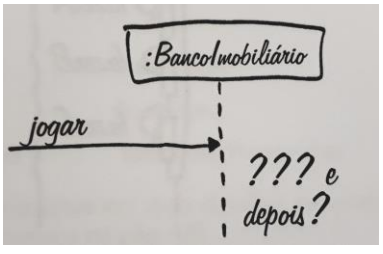
37

Padrão Controlador



38

Padrão Controlador



39

Padrão Coesão Alta