

Modulo II

Padrões GRASP

Professores

Eduardo Bezerra – edubezerra@gmail.com

Ismael H F Santos – ismael@tecgraf.puc-rio.br

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

1

Ementa

- **Padrões de Projeto GRASP**
 - Padrões GRASP
 - High Coesion
 - Low Coupling
 - Expert
 - Creator
 - Controller
 - Outros Padrões

Julho 06

Prof(s). Eduardo Bezerra & Ismael H. F. Santos

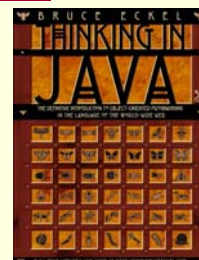
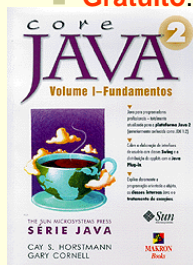
2

Bibliografia

- *Craig Larman, Utilizando UML e Padrões, Ed Bookman*
- *Eric Gamma, et ali, Padrões de Projeto, Ed Bookman*
- *Martin Fowler, Analysis Patterns - Reusable Object Models, Addison-Wesley, 1997*
- *Martin Fowler, Refatoração - Aperfeiçoando o projeto de código existente, Ed Bookman*

Livros

- **Core Java 2**, Cay S. Horstmann, Gary Cornell
 - Volume 1 (Fundamentos)
 - Volume 2 (Características Avançadas)
- **Java: Como Programar**, Deitel & Deitel
- **Thinking in Patterns with JAVA**, Bruce Eckel
 - **Gratuito.** <http://www.mindview.net/Books/TIJ/>



POO-Java

Padrões
GRASP



Julho 06

Prof(s). Eduardo Bezerra & Ismael H. F. Santos

5

Padrões GRASP

- GRASP: **G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns.
- Padrões de análise catalogados por Craig Larman.
- Indicam como atribuir **responsabilidades** a classes da melhor forma possível.
- Úteis na construção de
 - diagramas de interações
 - diagramas de classes
- Alguns padrões GRASP: Expert, Creator, High Coesion, Low Coupling, Controller.

Julho 06

Prof(s). Eduardo Bezerra & Ismael H. F. Santos

6

POO-Java

High Coesion



High Coesion (Coesão Alta)

- A **coesão** é uma medida do quão fortemente relacionadas e focalizadas são as responsabilidades de uma classe.
- Uma classe com baixa coesão:
 - faz muitas coisas não-relacionadas
 - executa trabalho demais.
- **Classes não coesas** são:
 - difíceis de compreender
 - difíceis de reutilizar
 - difíceis de manter
 - sensíveis a mudanças.

High Coesion (Coesão Alta)

- É extremamente importante assegurar que as responsabilidades atribuídas a cada classe sejam altamente relacionadas.
- Em um bom projeto OO, cada classe não deve fazer muito trabalho.
 - cada classe deve capturar apenas uma abstração.
- Como perceber que a coesão de uma classe está baixa?
 - Quando alguns atributos começam a depender de outros.
 - Quando há subgrupos de atributos correlacionados na classe.

Julho 06

Prof(s). Eduardo Bezerra & Ismael H. F. Santos

9

POO-Java

Low Coupling



Julho 06

Prof(s). Eduardo Bezerra & Ismael H. F. Santos

10

Low Coupling (Acoplamento Fraco)

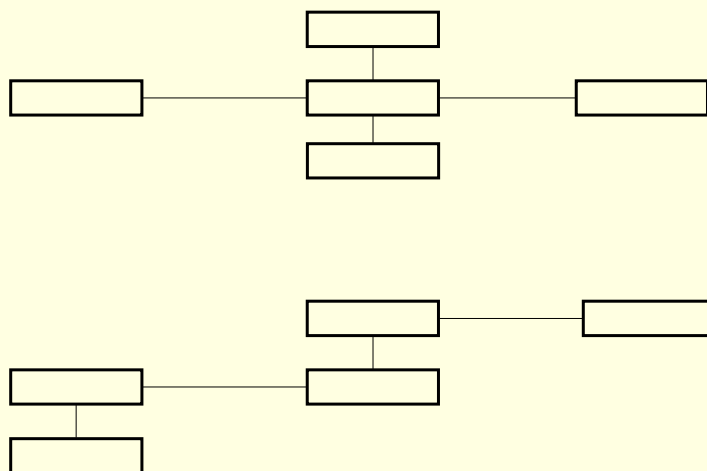
- O **acoplamento** é uma medida de quão fortemente uma classe está conectada a outras classes, tem conhecimento das mesmas ou depende delas.
- Uma classe com baixo (fraco) acoplamento não depende de muitas outras.
- Uma classe com acoplamento forte é:
 - mais difícil de compreender isoladamente
 - mais difícil de reutilizar (seu uso depende da reutilização das outras classes da qual ela depende)
 - sensível a mudanças nas classes associadas.
- Sempre que possível, evite que o envio de mensagens implique na criação de associações redundantes no modelo.

Julho 06

Prof(s). Eduardo Bezerra & Ismael H. F. Santos

11

Low Coupling (Acoplamento Fraco)



Julho 06

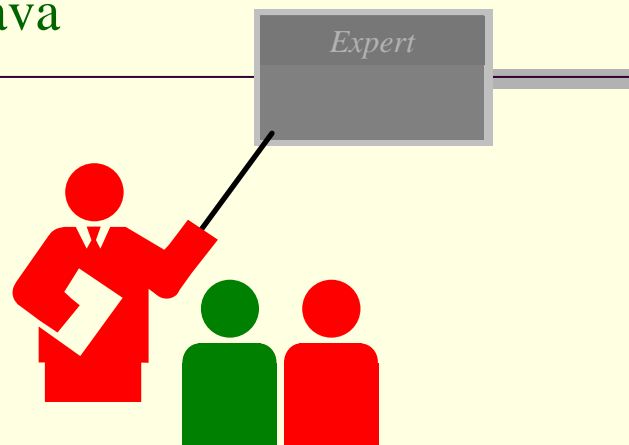
Prof(s). Eduardo Bezerra & Ismael H. F. Santos

12

Low Coupling (Acoplamento Fraco)

- **No slide anterior:**
 - Que configuração de classes é melhor?
 - Por que?
- **Aspectos gerais:**
 - Qual a relação do conceito de acoplamento com os objetos de controle em um caso de uso?
 - Quais propriedades de um produto de software estão relacionadas com esse conceito de acoplamento?

POO-Java

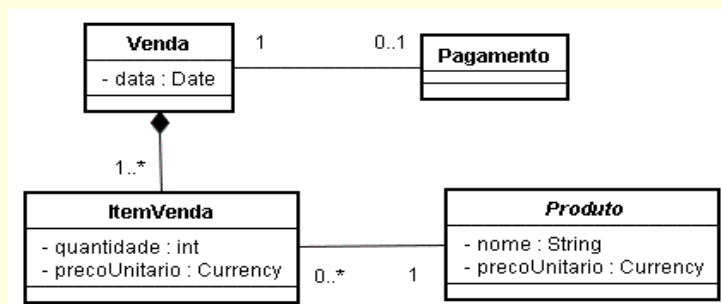


Expert

- É o padrão mais usado para atribuir responsabilidades
- **Problema:** dado um comportamento (responsabilidade) a qual classe essa responsabilidade deve ser alocada?
- **Solução:** atribuir essa responsabilidade ao **especialista da informação** – a classe que tem a informação necessária para satisfazer a responsabilidade.

Expert

- **Exemplo:**
 - Caso de uso registrar venda, foi identificada a responsabilidade do sistema gerar o total da venda.
 - Que classe deve assumir essa responsabilidade?



Expert

- A informação necessária para uma tarefa computacional frequentemente está “espalhada” por vários objetos.
- Portanto, há muitos **experts parciais**
 - Exemplo: determinar o total de uma venda requer a colaboração de 3 objetos, em 3 classes diferentes.
- Neste caso **mensagens** são usadas para estabelecer as colaborações
- Note que, com o uso do padrão Expert o **encapsulamento** das classes é mantido, já que:
 - objetos usam sua própria informação para cumprir suas responsabilidades ou
 - enviam mensagens a seus colaboradores para obter informações que não possuem

Julho 06

Prof(s). Eduardo Bezerra & Ismael H. F. Santos

17

Expert

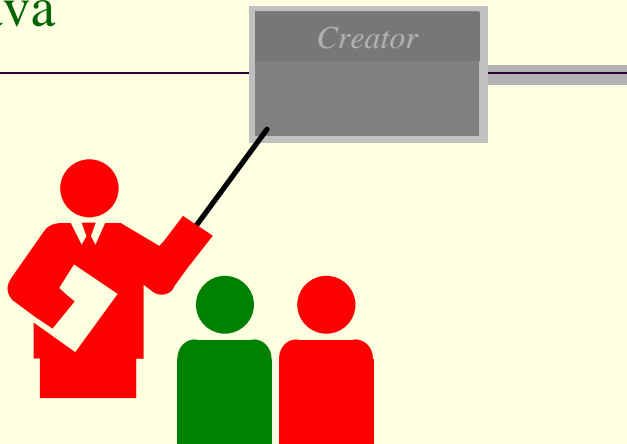
- Um aspecto importante a notar é que a atribuição de responsabilidades muitas vezes não tem correspondente no mundo real.
- Por exemplo, no mundo real, uma venda não calcula seu próprio total
 - Isso seria feito por uma pessoa (se não houvesse software)
- Mas no mundo OO:
 - Entidades inertes (como produtos) ou até conceitos (como uma venda) podem ter responsabilidades
 - Personificação dos objetos: objetos estão vivos!

Julho 06

Prof(s). Eduardo Bezerra & Ismael H. F. Santos

18

POO-Java



Julho 06

Prof(s). Eduardo Bezerra & Ismael H. F. Santos

19

Creator

- **Problema:** quem deve ser o responsável por criar instâncias de uma determinada classe? ”
- **Solução:** um objeto deve ser criado por outro que o possua como parte (agregação) ou esteja fortemente associado a ele.
- Para identificar o criador de um objeto A, verifique:
 - se o objeto A é *parte* em um relacionamento todo/parte; normalmente o todo é o responsável pela criação de A.
 - se algum outro objeto tem uma associação de um para muitos, onde A é o lado muitos.
 - se o objeto A está associado ao objeto de controle.
 - se alguma classe tem dados necessários à inicialização de A.

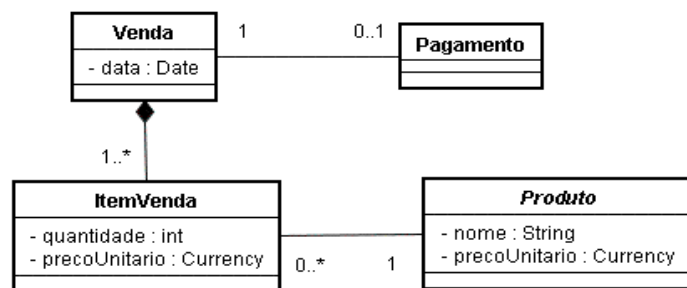
Julho 06

Prof(s). Eduardo Bezerra & Ismael H. F. Santos

20

Creator

- Exemplo:
 - Quem deve criar objetos ItemVenda?
 - Quem deve criar objetos Pagamento?
 - Quem deve criar objetos Venda?



Julho 06

Prof(s). Eduardo Bezerra & Ismael H. F. Santos

21

Creator

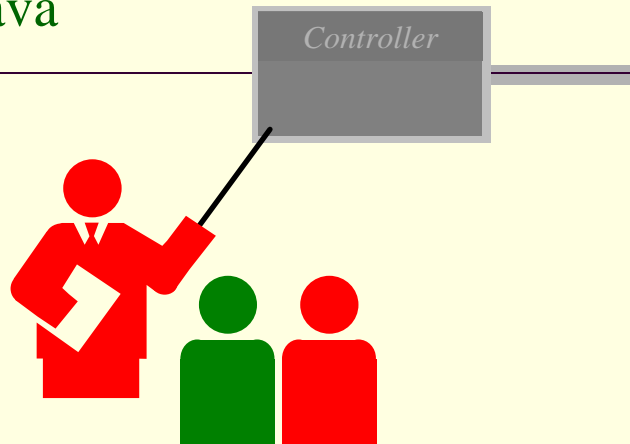
- É mais adequado escolher criador que estará conectado ao objeto criado, de qualquer forma, depois da criação.
- Isso leva ao **acoplamento baixo**, já que o objeto criado deve normalmente ser visível ao criador.
- Exemplo de criador que possui os valores de inicialização
 - Uma instância de Pagamento deve ser criada
 - A instância deve receber o total da venda
 - Quem tem essa informação? Venda
 - Venda é um bom candidato para criar objetos da classe Pagamento
- **Creator é um caso particular de Expert**
 - Por que?

Julho 06

Prof(s). Eduardo Bezerra & Ismael H. F. Santos

22

POO-Java



Controller

- **Problema:** quem deveria ser responsável por tratar um **evento do sistema**?
- **Solução:** atribuir a responsabilidade do tratamento de um evento do sistema a uma classe que representa uma das seguintes escolhas:
 - Representa o “sistema” todo (**controlador fachada**)
 - Representa um tratador oficial de todos os eventos de sistema de um caso de uso (**controlador de caso de uso**)

POO-Java

Outros
Padrões



Julho 06

Prof(s). Eduardo Bezerra & Ismael H. F. Santos

25

Outros padrões de análise

- Além dos padrões GRASP catalogados por Craig Larman, Martin Fowler escreveu um excelente livro sobre padrões de análise:
 - Martin Fowler, *Analysis Patterns - Reusable Object Models*, Addison-Wesley, 1997.
- Alguns padrões descritos no livro:
 - Padrões de organizações e responsabilidades.
 - Padrões de Observações e Medições
 - Padrões de Observações para a Finanças Corporativas
 - Padrões de Inventário e Contabilidade
 - Padrões de Planejamento
 - Padrões para o Comércio
 - Padrões de Contratos de Derivativos

Julho 06

Prof(s). Eduardo Bezerra & Ismael H. F. Santos

26