Modelação e Design
21: Princípios de Design Orientado a Objetos

Leonor Melo
leonor@isec.pt

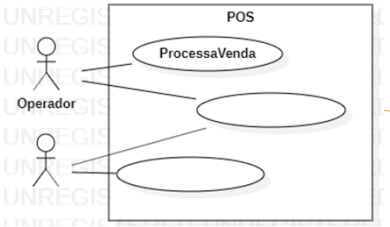
1

Design Orientado a Objetos

- Responsabilidades
- Princípios GRASP
  - Criador
  - Perito em informação
  - Acoplamento baixo
  - Controlador
  - Coesão elevada
  - Polimorfismo
  - Invenção
  - Variações protegidas

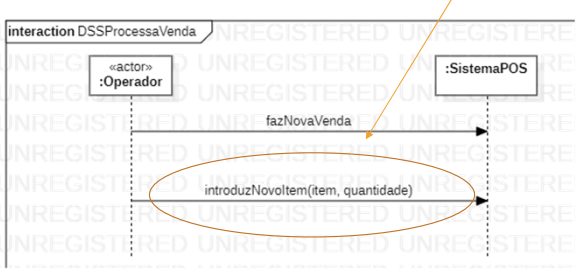
2

Onde estamos



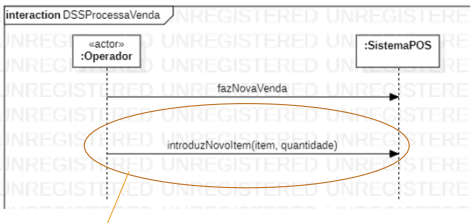
ProcessaVenda

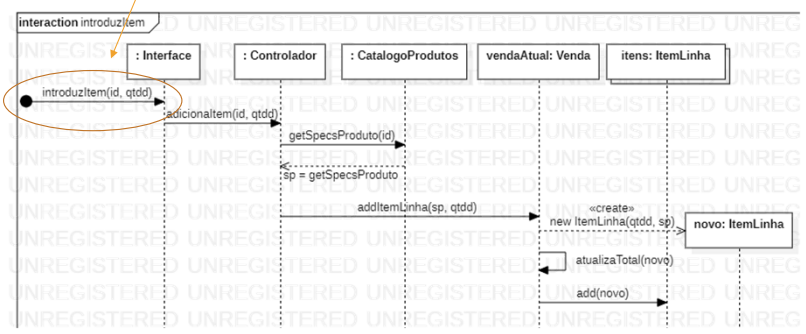
1. O cliente chega....
2.
3. O operador introduz o identificador do item...



3

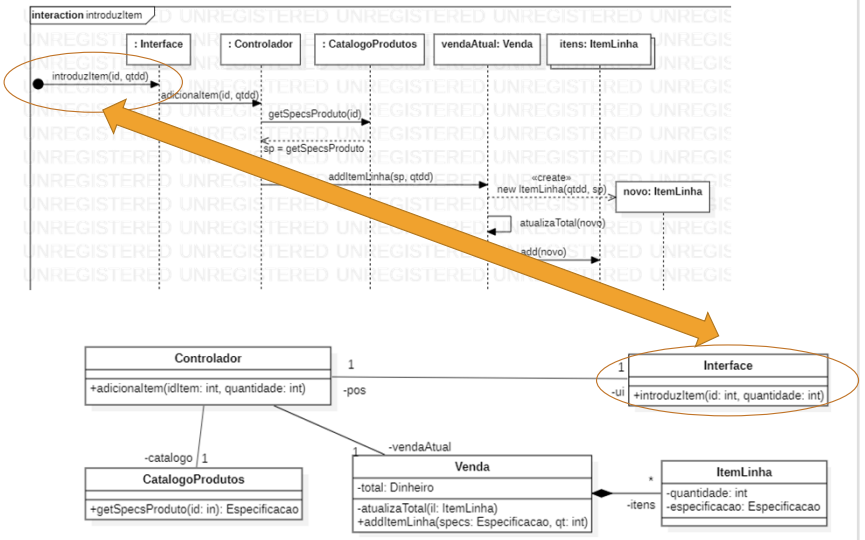
Onde estamos





4

Onde estamos



5

Princípios de design

- Estratégias que ajudam a criar um design
  - Limpo
  - Modular
  - Fácil de testar
  - Fácil de corrigir
  - Fácil de manter

6

Design guiado por responsabilidades

- Objetos / classe / componentes são vistos como entidades com
  - Responsabilidades
  - Papel a desempenhar
  - Colaborações
- Responsabilidade
  - "contrato ou obrigação de um classificador"
- GRASP – princípios que auxiliam na atribuição das responsabilidades

7

Responsabilidades

- Um objeto tem 2 tipos de responsabilidades
  - Responsabilidades de fazer
    - Fazer alguma coisa ele próprio como criar um objeto ou fazer um cálculo
    - Iniciar uma ação em outro objeto
    - Controlar e coordenar atividades com outros objetos
  - Responsabilidades de conhecer
    - Conhecer dados privados e encapsulados
    - Conhecer objetos com os quais tem uma relação
    - Conhecer dados que pode derivar ou calcular

8

Responsabilidades

- Objeto Venda
  - Responsabilidades de fazer
    - Uma Venda é responsável por criar um ItemDeLinha
  - Responsabilidades de conhecer
    - Uma Venda é responsável por conhecer o total
- Modelo do domínio muitas vezes inspira as "responsabilidades de conhecer" dos objetos que também pertençam ao modelo de design

9

Colaborações

- Responsabilidades são implementas através de métodos que
  - agem sozinhos, ou
  - **colaboram** com outros métodos e objetos
- Responsability Driven Design
  - Metáfora
  - Sistema é visto como uma comunidade de objetos com responsabilidades que colaboram entre si

10

Princípios do GRASP

- Criador (Creator)
- Perito (Information Expert)
- Acoplamento Baixo (Low Coupling)
- Controlador (Controller)
- Coesão Elevada (High Cohesion)
- Polimorfismo (Polymorphism)
- Invenção (Pure Fabrication)
- Indireção (Indirection)
- Variações protegidas (Protected Variations)

11

Criador

- Perguntas a que quer responder:
  - Quem cria um objeto?
  - Quem deveria criar instancias de determinada classe?
- Resposta:
  - Atribuir à classe B a responsabilidade de criar instancias da classe A se
    - B "contém" ou "é composta por" A
    - B regista A
    - B usa A com frequência / proximidade
    - B tem a informação necessária para inicializar A

12

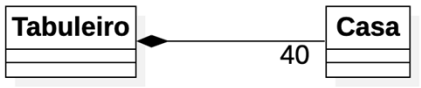
Criador

- Exemplo:
  - Tabuleiro e Casa
  - Tabuleiro tem uma relação de composição com a Casa
  - Tabuleiro é o "todo" e Casa é a "parte"
    - As instancias de Casa devem ser criadas pela classe Tabuleiro

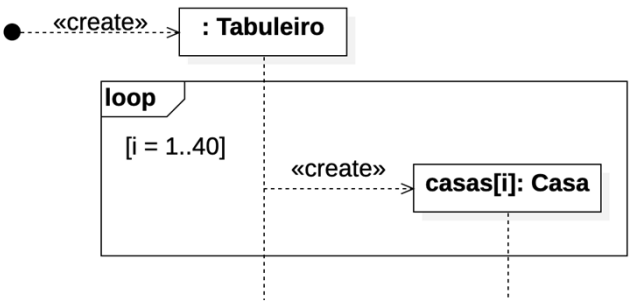
13

Criador

- Diagrama de Classes:



- Diagrama de sequencia:



14

Perito

- Perguntas a que quer responder:
  - Qual o princípio básico para atribuir uma responsabilidade a um objeto?
- Resposta:
  - Atribuir a responsabilidade à classe que tem a informação necessária para executar a operação (sozinha ou em colaboração com outras classes)

15

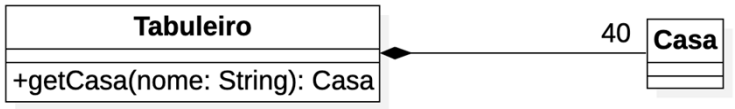
Perito

- Exemplo:
  - Qual a classe que deve ser responsável por, dado o nome de uma casa, devolver a referencia para essa casa?
    - A classe Tabuleiro, que conhece todas as casas

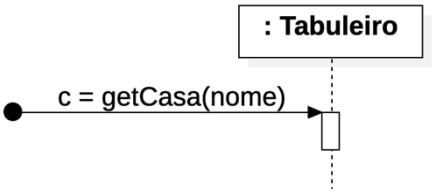
16

Perito

- Diagrama de Classe:



- Diagrama de Sequencia:



17

Acoplamento Baixo

- Perguntas a que quer responder:
  - Como reduzir o impacto de alterações?
- Resposta:
  - Atribuir a responsabilidade de forma a que acoplamentos desnecessários permaneçam baixos.
  - Usar esse princípio para avaliar alternativas

18

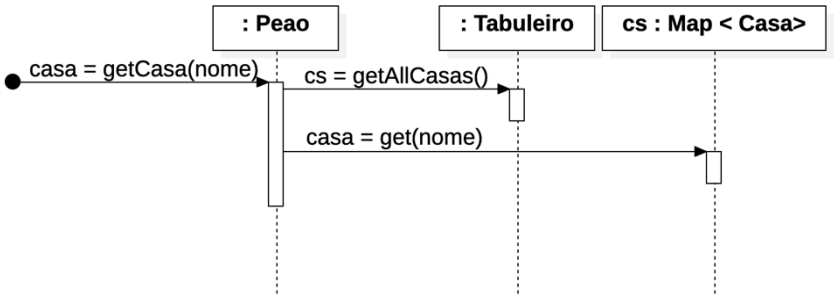
Acoplamento Baixo

- Exemplo de acoplamento desnecessariamente elevado:
  - Tornar a classe Peão (ou outra qualquer) responsável por, dado o nome de uma casa, devolver a referencia para essa casa

19

Acoplamento Baixo

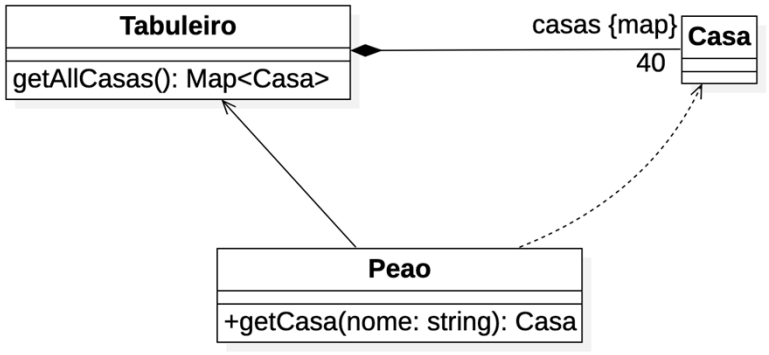
- Diagrama de Sequencia do exemplo de acoplamento desnecessariamente elevado:



20

Acoplamento Baixo

- Diagrama de Classe do exemplo de acoplamento desnecessariamente elevado:



21

Acoplamento Baixo

- Quando é necessário alterar o software, o baixo acoplamento tende a reduzir
  - Tempo necessário
  - Esforço despendido
  - Defeitos introduzidos

22

Controlador

- Uma arquitetura simples tem pelo menos uma camada de interface com o utilizador (UI layer) e uma camada de domínio/aplicação/negócio
- Os objetos da camada de UI (botões, caixas de texto,…) recebem eventos do utilizador, mas não devem ser responsáveis pela lógica do domínio
  - Devem delegar o pedido para os objetos do domínio

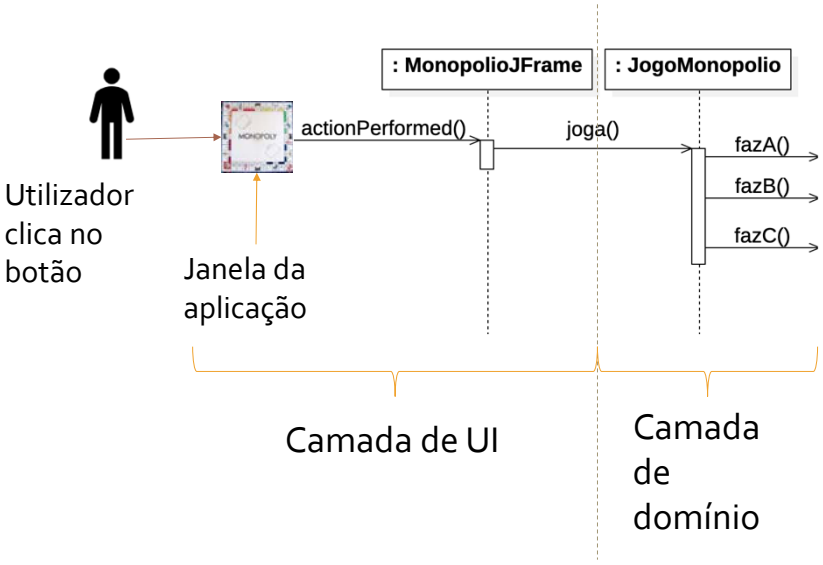
23

Controlador

- Pergunta:
  - Qual é o primeiro objeto, depois da camada de interface com o utilizador (UI), que deve receber a mensagem da camada de UI
- Resposta:
  - Atribuir responsabilidade a um objeto que represente uma das seguintes escolhas:
    - Representa o "sistema" completo,
    - Representa o dispositivo onde o sistema se está a executar ou um subsistema
    - Representa um caso de uso dentro do qual a operação ocorre

24

Controlador



25

Controlador

- Bloated Controller
  - Controlador com demasiadas responsabilidades
    - Solução: adicionar outros controladores
  - Controlador que realiza demasiadas tarefas
    - Solução: controlador deve delegar parte das tarefas a outras classes

26

Coesão Elevada

- Pergunta:
  - Como manter os objetos focados, compreensíveis, manejáveis e como consequência com acoplamento baixo?
- Resposta:
  - Atribuir responsabilidades de forma que a coesão permaneça elevada.
  - Usar a coesão como critério entre soluções alternativas

27

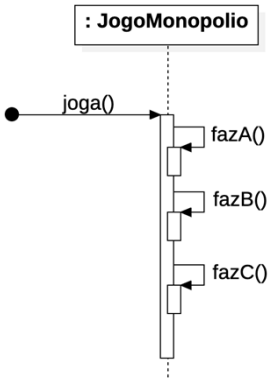
Coesão Elevada

- Coesão:
  - Mede quão funcionalmente relacionadas estão as operações de um elemento de software; quanto trabalho está a ser realizado por um dado elemento
- Baixa coesão (mau):
  - Um elemento faz muitas atividades não relacionadas entre si
  - Um elementos é responsável por uma parte grande do trabalho
- Baixa coesão tende a originar um acoplamento elevado

28

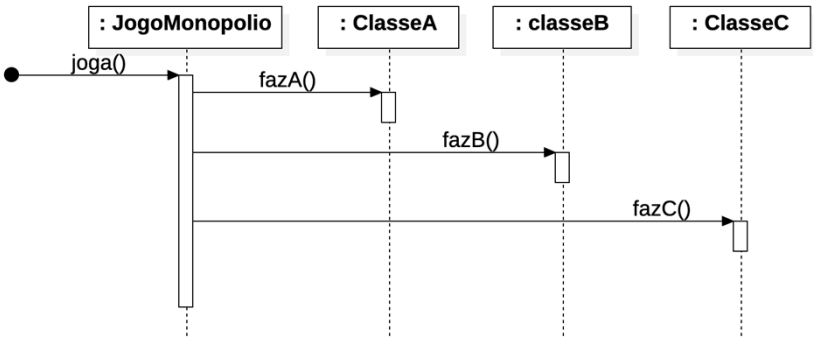
Coesão Elevada

- Diagrama de Sequencia de um exemplo de coesão baixa (pior):



Coesão Elevada

- Diagrama de Sequencia de um exemplo de coesão alta (melhor):

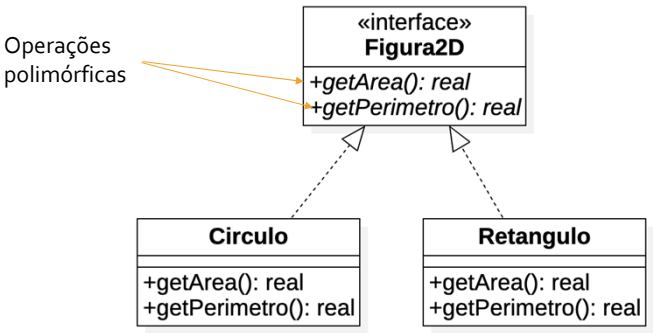


Polimorfismo

- Problema:
  - Como lidar com alternativas baseadas no tipo dos objetos? Como criar componentes que possam ser facilmente substituídos sem danos para o sistema?
- Solução:
  - Usar operações polimórficas (i.e. com a mesma assinatura, mas com implementações distintas consoante o tipo de objeto) – os diferente objetos devem estra relacionados entre si pela implementação do mesmo interface ou por relações de generalização

31

Polimorfismo



- Se enviarmos uma mensagem getArea() a um objeto2D, a implementação correta vai ser automaticamente usada de acordo com o tipo do objeto.

32

Invenção

- Problema:
  - Que objeto deve ter determinada responsabilidade quando não queremos violar os princípios de coesão elevada e acoplamento fraco, mas a solução sugerida pelo "perito em informação" não é boa
- Solução:
  - atribuir um conjunto muito coeso de responsabilidades a uma classe de conveniência artificial, mesmo que não represente nenhuma entidade do domínio, de forma a suportar coesão elevada, acoplamento fraco e reutilização

33

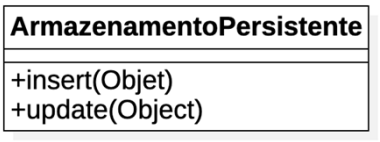
Invenção

- Exemplo:
  - Que objeto deve ser responsável por guarda a informação de uma Venda na base de dados?
  - A informação pertence à Venda pelo que o perito em informação é a Venda
  - Mas guardar informação na base de dados implica uma série de operações relacionadas apenas com a base de dados
  - Implica também relacionamento com determinado interface de base de dados
    - Acrescentar estas operações diminuir a coesão e aumentar o acoplamento da Venda

34

Invenção

- Exemplo:
  - É melhor criar uma classe unicamente responsável por gravar a informação na base de dados, que eventualmente poderá ser reutilizada por várias outras classes



35

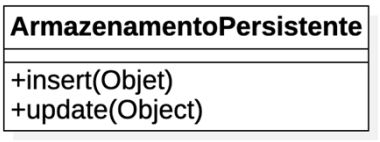
Indireção

- Problema
  - Como atribuir responsabilidades de forma a evitar acoplamento direto entre duas (ou mais) classes? Como desacoplar objetos de forma que o acoplamento permaneça baixo e o potencial de reutilização continue elevado?
- Solução:
  - Atribuir a responsabilidade a um objeto intermédio que irá mediar as relações entre os outros componentes

36

Invenção

- Exemplo:
  - O ArmazenamentoPersistente é também um exemplo de indireção, ao servir de intermediário entre a Venda e a Base de Dados



37

Variações Protegidas

- Problema
  - Como desenhar objetos, subsistemas e sistemas de forma que as variações ou instabilidade nestes elementos não tenham um impacto negativo nos outros elementos?
- Solução:
  - Identificar os pontos de previsível instabilidade e atribuir responsabilidades de forma a criar um interface estável

38

Variações Protegidas

- Exemplo:
  - A criação do interface para aplicação do polimorfismo no exemplo das figuras geométricas:
    - Mais tarde podem ser acrescentadas outras figuras (Triangulo, Pentágono, …): se estas novas classes também implementarem o interface Figura2D, o restante software, que já depende do Figura2D, não tem de ser alterado.