Modelação e Design
23: Design patterns

Leonor Melo
leonor@isec.pt

1

Design patterns

- Princípios de Orientação a Objetos
- Definição e uso de design patterns
- Design patterns
  - creacionais
  - estruturais
  - comportamentais

2

Princípios de Orientação a Objetos

- Princípios OO:
  - Encapsular o que varia
  - Preferir composição a herança
  - Depender de interfaces e não de implementações
  - Design que favoreça o baixo acoplamento entre objetos que interajam
  - "only talk to your friends" (o objeto *a* pode pedir um serviço a um objeto *b*, mas não deve "atravessar" o objeto *b* para chegar ao objeto *c* e pedir-lhe um serviço)

3

Princípios de Orientação a Objetos

- Princípios OO (continuação):
  - Classes devem ser abertas para extensão mas fechadas para modificação
  - Depender de abstrações. Não depender de classes concretas
  - "dont call us, we'll call you": uma classe cliente recebe um serviço de que depende sem ter de o saber construir. O serviço injetado faz parte do estado do cliente.
  - A classe deve ter uma razão para mudar (a classse deve apenas ter um propósito)

4

Definição e uso

- Um design pattern (padrão de desenho) é uma solução
  - Genérica e
  - Reproduzível
  - Para um problema de design comum
- Não é uma solução de design completa que possa ser transformada em código
  - É uma descrição ou template do modo de resolver o problema
    - Que pode ser aplicado a diferentes situações

5

Utilização

- Design Patterns:
  - Fornecem soluções genéricas, documentadas num formato que não requer que fique associado apenas a um caso ou problema específico
  - Podem acelerar o processo de desenvolvimento ao fornecer paradigmas de desenvolvimento testados com sucesso
  - Melhoram a comunicação e legibilidade do código (para quem esteja familiarizado com os patterns)

6

Elementos essenciais

- Elementos essenciais:
  - Nome
    - O nome do design pattern. Fornece um vocabulário comum para os designers de software
  - Problema
    - Descrição do da situação onde o pattern deve ser aplicado
  - Solução
    - Elementos que compõe o design: estrutura, participantes, colaborações
  - Consequências
    - Resultados, vantagens e desvantagens de aplicar o pattern

7

Design Patterns do GoF

- Gang of Four (GoF)
  - Autores do livro "Design Patterns Elements of reusable Object-Oriented Sortware"
  - documentaram 23 problemas comuns e as respetivas soluções mais bem aceites
- Classificados por
  - Propósito
  - Âmbito
    - Classes: relações entre classes e subclasses, estático
    - Objeto: relações entre objetos, dinâmico

8

Tipos de design patterns

- Creational
  - Como podem os objetos ser criados
    - Facilidade de manutenção
    - Controlo
    - Extensibilidade
- Structural
  - Como organizar em estruturas
    - Gestão da complexidade
    - Eficiência
- Behavioural
  - Como atribuir responsabilidades aos objetos
    - Desacoplar os objetos
    - flexibilidade
    - Melhor comunicação

9

Creational patterns

- Tornam o processo de instanciação mais abstrato
- Exemplos:
  - Factory
  - Singleton
  - Builder
  - Prototype

10

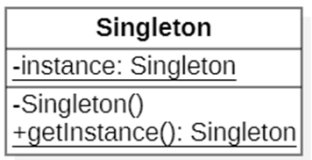
Padrão Singleton

- Garante que apenas uma instancia é criada e fornecem acesso global a esse objeto:
- Construtor é privado
- Um atributo privado para um objeto da própria classe
- Método getInstance()
  - Cria a instancia se esta ainda não existir
  - Devolve a referencia para a instancia

11

Padrão Singleton

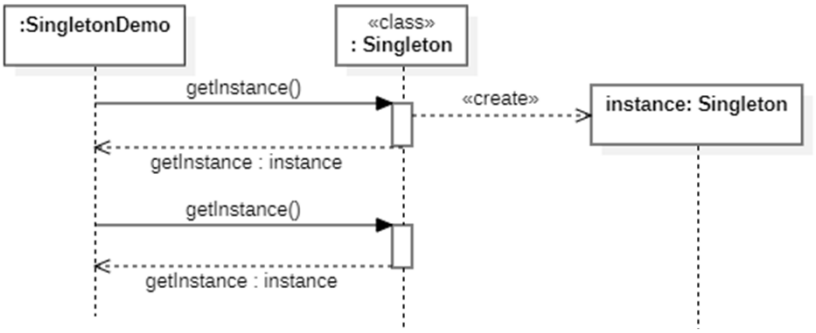
- Diagrama de classes:



12

Padrão Singleton

- Exemplo de utilização:



13

Padrão Singleton

- Quando usar
  - Quando queremos garantir que apenas uma instancia da classe é criada
  - Quando essa instancia tem de ser acessível por todo o código
  - Em ambientes multi-thread quando vários threads têm de aceder ao mesmo recurso (objeto singleton)
- Usos comuns
  - Classes de configuração

14

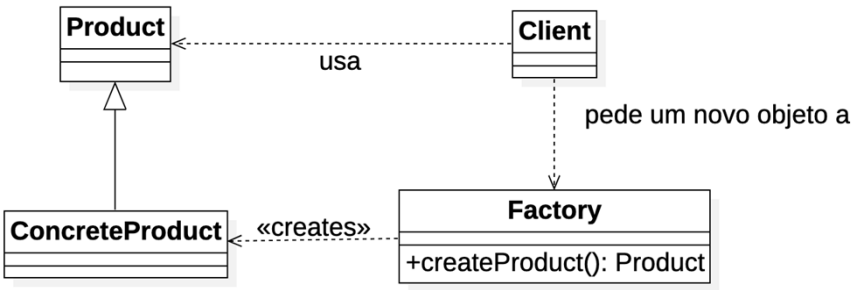
Padrão Factory

- Cria um objeto omitindo os detalhes da instanciação:
  - Permite que a identidade do objeto seja escolhida apenas em run-time
- Refere-se ao objeto acabado de criar através de um interface comum
- O método que cria o objeto em particular recebe dados genéricos
  - O objeto criado tem de fazer parte de uma hierarquia de classes

15

Padrão Factory

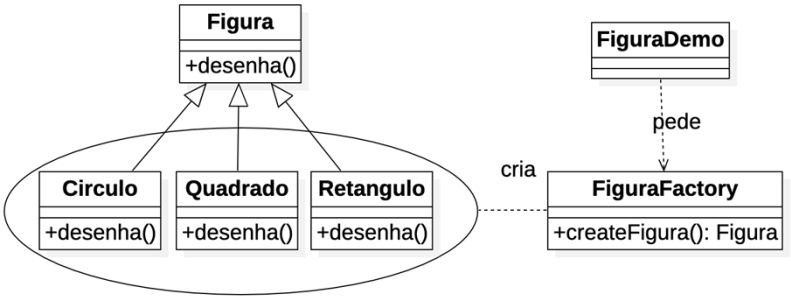
- Diagrama de classes:



16

Padrão Factory

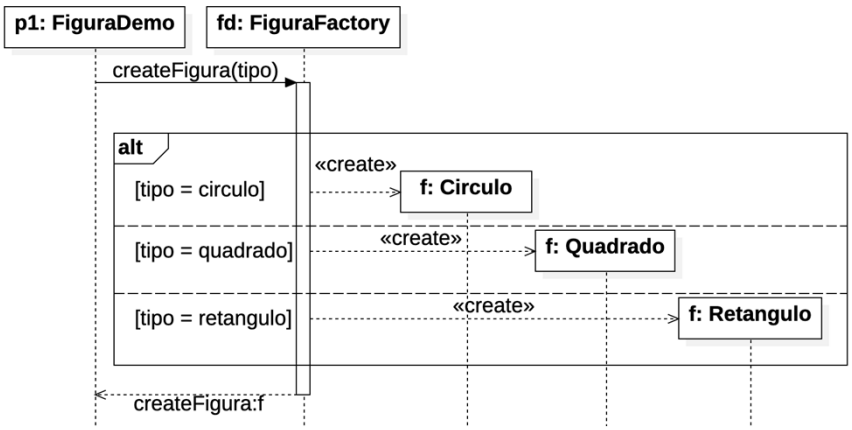
- Exemplo:



17

Padrão Factory

- Exemplo utilização:



18

Structural patterns

- Como devem se devem compor as classes e objetos de maneira a formar estruturas maiores
- Exemplos:
  - Adapter
  - Decorator
  - Proxy
  - Bridge
  - Composite

19

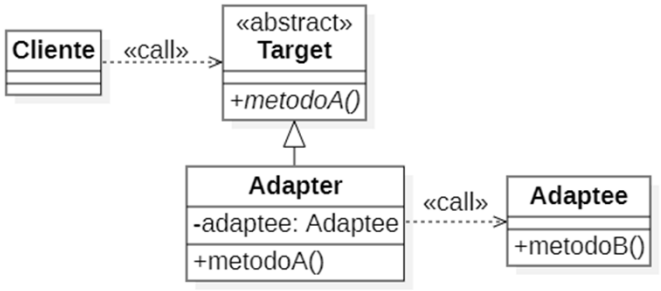
Adapter pattern

- Converte o interface de uma classe em outro interface que o cliente está à espera de usar
- Permite que classe que normalmente não trabalhariam juntas devido a interfaces incompatíveis o façam

20

Adapter pattern

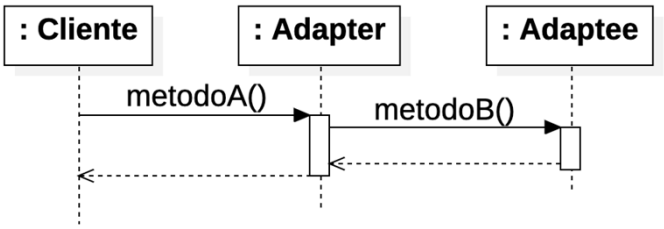
- Diagrama de classes:



21

Adapter pattern

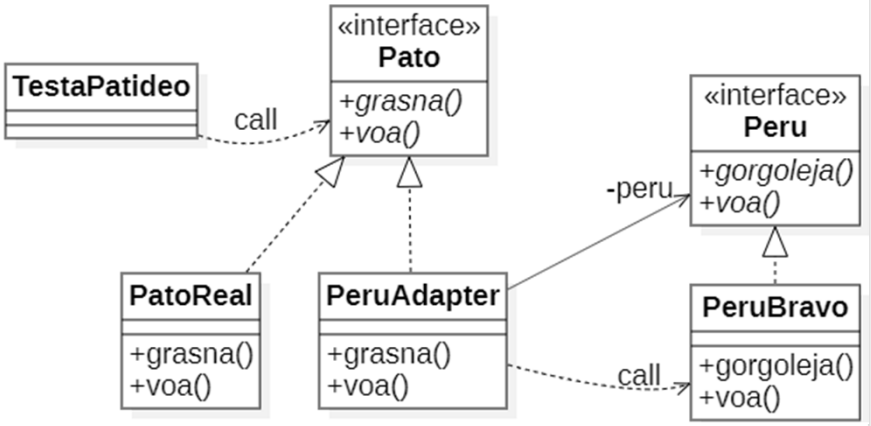
- Diagrama de sequencia:



22

Adapter pattern

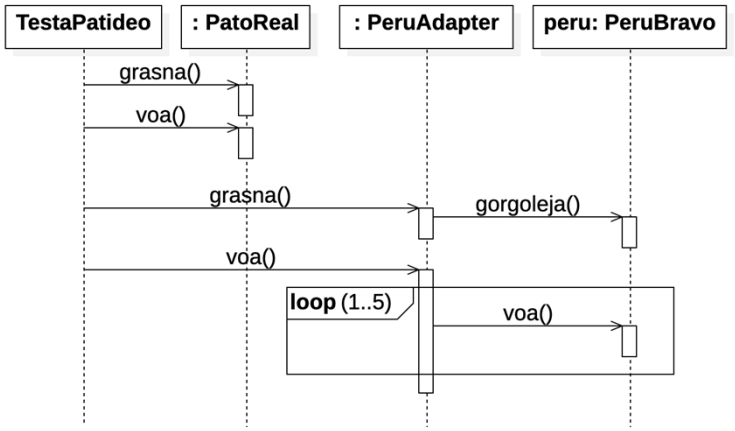
- Exemplo: PeruAdapter, interface de um pato, comportamento executado por um peru



23

Adapter pattern

- Exemplo: PeruAdapter, interface de um pato, comportamento executado por um peru



24

Behavioral patterns

- Dizem respeito a algoritmos e a atribuição de responsabilidades aos objetos
  - Usam herança (classes) ou composição (objetos) para distribuir as responsabilidades
- Exemplo:
  - Strategy
  - Command
  - Iterator
  - Observer
  - Chain of responsibility

25

Observer

- Permite que vários objetos sejam notificados da alteração de estado de outros objetos existentes no sistema
- Quando um objeto muda de estado todos os seus dependentes são notificados e atualizados de forma automática
  - A informação pertence apenas ao objeto concreto que está a ser observado. Os observadores concretos têm apenas uma relação de dependência com o observado

26

Observer

- Os observadores podem subscrever (e deixar de subscrever) as atualizações do observado
- Quando existe uma mudança de estado do observado este notifica todos os observadores
  - A informação pertence apenas ao objeto concreto que está a ser observado.
  - Os observadores concretos têm apenas uma cópia da informação disponibilizada pelo observado

27

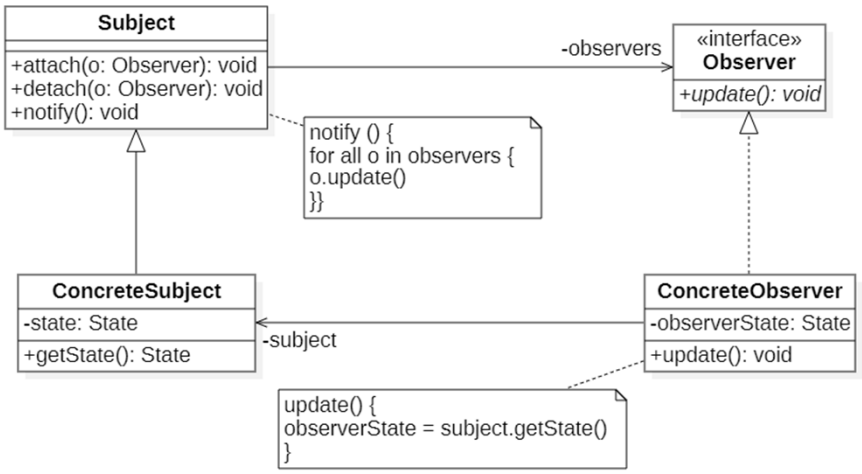
Observer

- Responsabilidade do observado:
  - notificar os observadores de que ocorreu uma mudança de estado
- Responsabilidade dos observadores:
  - subscreverem (e deixarem de subscrever) o observado
  - atualizar a sua representação do estado do observado quando são notificados

28

Observer

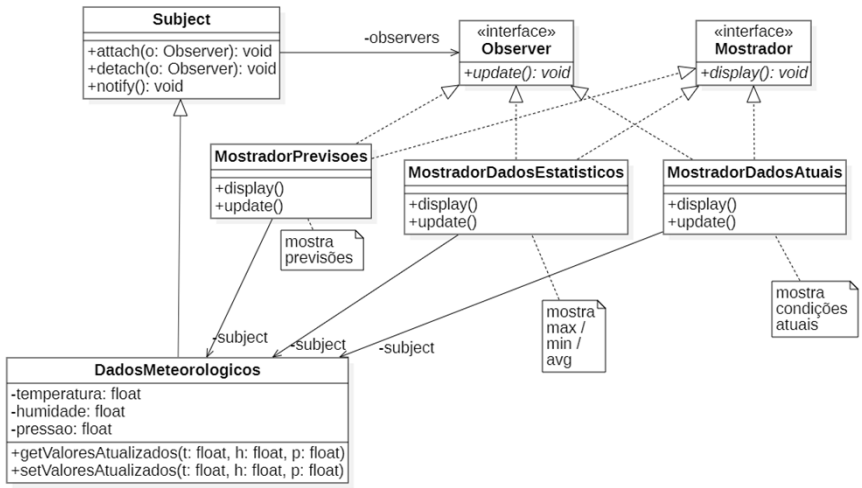
- Diagrama de classes:



29

Observer

- Exemplo:



30

Observer

- Exemplo: