Resumo arquitetura de software pt VII

Padrões de projeto

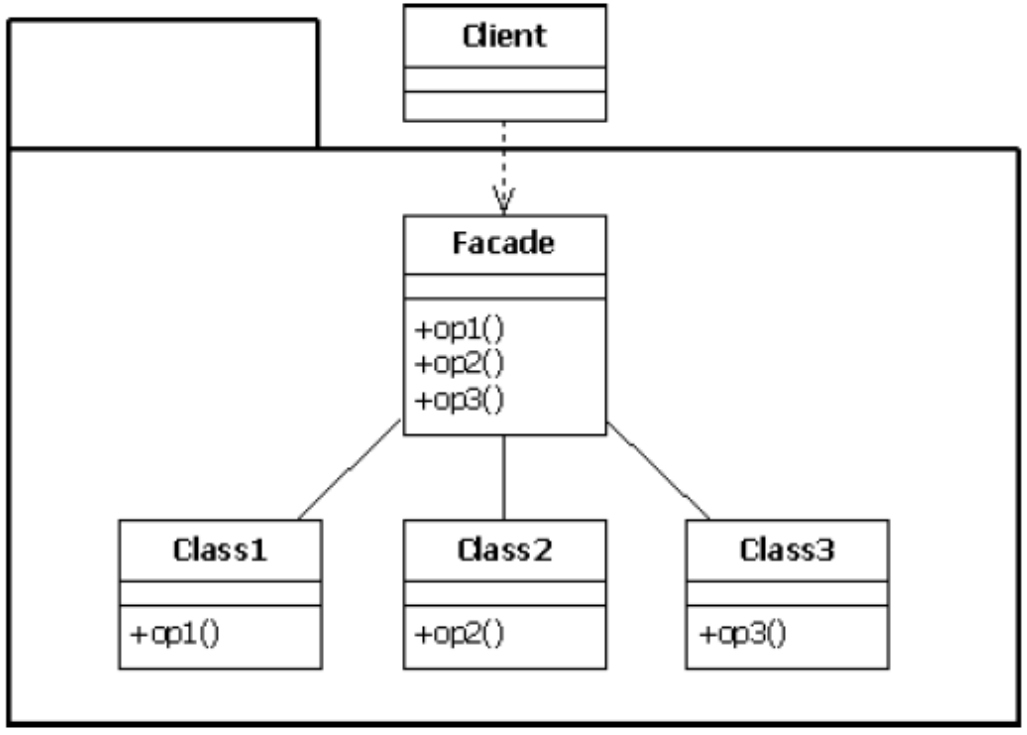
Padrões

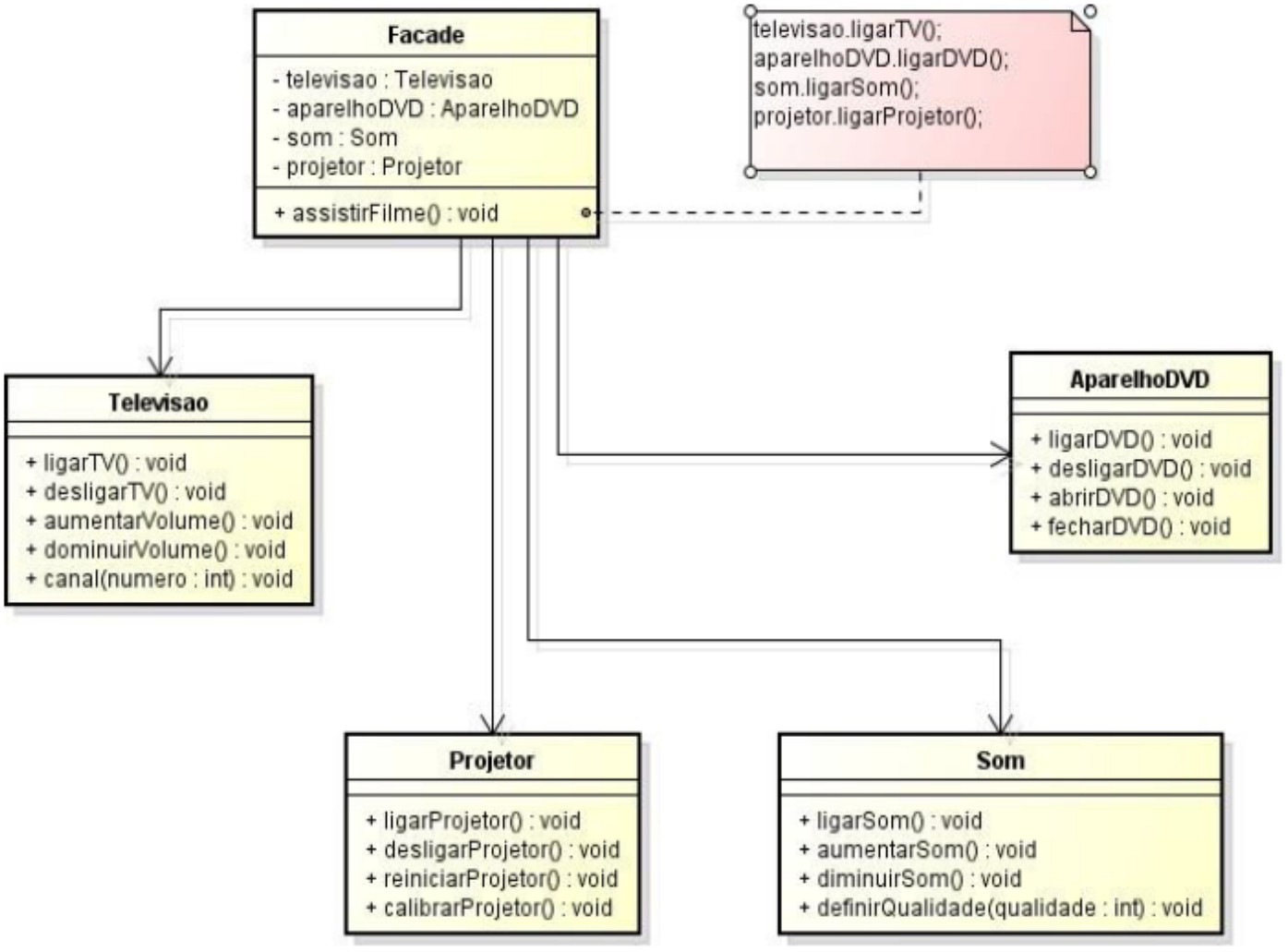
- sistemas estruturados utilizam padrões
- Reutilizam soluções bem sucedidas no passado
- Descreve uma solução comum para um problema recorrente
    - Um mecanismo é um padrão de projeto aplicado a um conjunto de classes
    - Um framework é um padrao de arquitetura que fornece um template

Padroes de projeto?

- Sistematizam soluçoes:
    - nome
    - problema
    - solução, consequencia, exemplo

- A representação pode ser:
  - Abstrata - para facilitar o entendimento
  - Diagrama UML - para funcionamento


Client
Facade
+op1()
+op2()
+op3()
Class1
+op1()
Class2
+op2()
Class3
+op3()


Facade
- televisao : Televisao
- aparelhoDVD : AparelhoDVD
- som : Som
- projetor : Projetor
+ assistirFilme() : void
televisao.ligarTV();
aparelhoDVD.ligarDVD();
som.ligarSom();
projetor.ligarProjetor();
Televisao
+ ligarTV() : void
+ desligarTV() : void
+ aumentarVolume() : void
+ dominuirVolume() : void
+ canal(numero : int) : void
AparelhoDVD
+ ligarDVD() : void
+ desligarDVD() : void
+ abrirDVD() : void
+ fecharDVD() : void
Projetor
+ ligarProjetor() : void
+ desligarProjetor() : void
+ reiniciarProjetor() : void
+ calibrarProjetor() : void
Som
+ ligarSom() : void
+ aumentarSom() : void
+ diminuirSom() : void
+ definirQualidade(qualidade : int) : void

Framework?

- Baseado em padrões voltados para uma linguagem

Padrões de projeto?

- Cada padrão descreve um problema que ocorre repetidamente , descreve então a parte central da solução e forma o que pode usar esta solução

Gand of four?

- Vocabulário comum para conversar sobre projetos
- Soluções que nao tinham nome, passam a ter nome
- Fábricas, fachadas ....

O Formato dos padrões no GoF

- **Nome** (inclui número da página)
  - um bom nome é essencial para que o padrão caia na boca do povo
- **Objetivo / Intenção**
- **Também Conhecido Como**
- **Motivação**
  - um cenário mostrando o problema e a necessidade da solução
- **Aplicabilidade**
  - como reconhecer as situações nas quais o padrão é aplicável
- **Estrutura**
  - uma representação gráfica da estrutura de classes do padrão (usando OMT91) em, às vezes, diagramas de interação (Booch 94)
- **Participantes**
  - as classes e objetos que participam e quais são suas responsabilidades

- **Colaborações**
  - como os participantes colaboram para exercer as suas responsabilidades
- **Consequências**
  - vantagens e desvantagens, trade-offs (porque escolher uma ou outra)
- **Implementação**
  - com quais detalhes devemos nos preocupar quando implementamos o padrão
  - aspectos específicos de cada linguagem
- **Exemplo de Código**
  - no caso do GoF, em C++ (a maioria) ou Smalltalk
- **Usos Conhecidos**
  - exemplos de sistemas reais de domínios diferentes onde o padrão é utilizado
- **Padrões Relacionados**
  - quais outros padrões devem ser usados em conjunto com esse
  - quais padrões são similares a este, quais são as diferenças

Tipos de padroes de projeto?

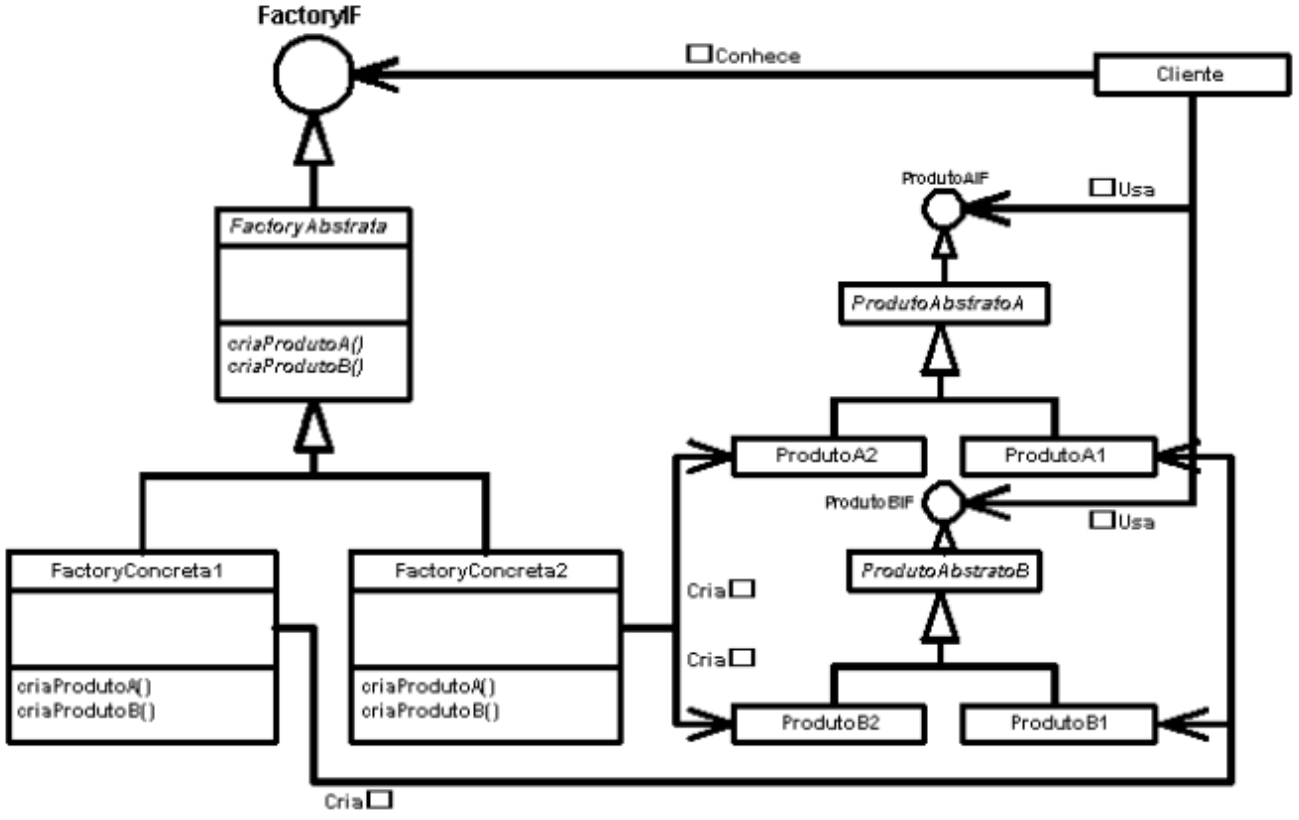
- **Criacionais**
- **Estruturais**
- **Comportamentais**

Abstract Factory

Abstract factory?

- Prover uma interface para criação de famílias de objetos relacionados sem especificar sua classe concreta

- Use uma fábrica abstrata quando:
  - um sistema deve ser independente da forma como seus produtos são criados e representados;
  - um sistema deve poder lidar com uma família de vários produtos diferentes;
  - você quer prover uma biblioteca de classes de produtos mas não quer revelar as suas implementações, quer revelar apenas suas interfaces.


FactoryIF
Conhece
Cliente
ProdutoAIF
Usa
FactoryAbstrata
criaProdutoA()
criaProdutoB()
ProdutoAbstratoA
ProdutoA2
ProdutoA1
ProdutoBIF
Usa
FactoryConcreta1
criaProdutoA()
criaProdutoB()
FactoryConcreta2
criaProdutoA()
criaProdutoB()
Cria
ProdutoAbstratoB
Cria
ProdutoB2
ProdutoB1
Cria

- Para look-and-feel diferentes (Motif (Linux), Windows, Mac, Presentation Manager, etc.) temos formas diferentes de manipular janelas, scroll bars, menus, etc.
- Para criar uma aplicação com GUI que suporte qualquer look-and-feel, precisamos ter uma forma simples de criar objetos (relacionados) de uma mesma família
- Os objetos são dependentes porque não podemos criar uma janela estilo Windows e um menu estilo Motif (Linux)
- Java já resolveu este problema internamente no package awt usando Abstract Factory e você não precisa se preocupar com isso

Fábrica Abstrata – Conseqüências

- O padrão
  1. Isola as classes concretas dos clientes;
  2. Facilita a troca de famílias de produtos (basta trocar uma linha do código pois a criação da fábrica concreta aparece em um único ponto do programa);
  3. Promove a consistência de produtos (não há o perigo de misturar objetos de famílias diferentes);
  4. Dificulta a criação de novos produtos ligeiramente diferentes (pois temos que modificar a fábrica abstrata e todas as fábricas concretas).

Fábrica Abstrata – Implementação

- Na fábrica abstrata, cria-se um método fábrica para cada tipo de produto. Cada fábrica concreta implementa o código que cria os objetos de fato.

- Se tivermos muitas famílias de produtos, teríamos um excesso de classes "fábrica concretas".

- Para resolver este problema, podemos usar o **Prototype**: criamos um dicionário mapeando tipos de produtos em instâncias prototípicas destes produtos

- Então, sempre que precisarmos criar um novo produto pedimos à sua instância prototípica que crie um clone (usando um método como **clone()** ou **copy()**).