Padrões de projeto

Prof. Hugo de Paula

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Curso de Engenharia de Software

Sumário

Padrões de projeto

Padrão

Um padrão (*pattern*) é uma descrição das características de uma solução comprovada para um problema recorrente, onde os elementos essenciais são considerados e os detalhes irrelevantes são omitidos. (Gamma et al, 1995)

- Solução bem conhecida para um problema específico.
- Originária do conceito de arquitetura de software, e posteriormente transportada para o domínio da programação.

Reuso de software

Arquitetura de um software

Como os componentes do sistema são organizados, ou distribuídos, e como eles se relacionam externamente com outros softwares.

- Reuso de software:
  - Aumenta produtividade.
  - Melhora qualidade do software.
  - Utiliza artefatos já testados e validados.

Design Pattern

- Beck e Cunningham apresentaram os primeiros padrões computacionais na conferência OOPSLA, em 1987, para a construção de janelas na linguagem Smalltalk.
- Popularizou-se com o livro de Gamma, Helm, Johnson, Vlissides (Gang of 4), *Design Patterns: Elements of Reusable Object-Oriented Software* (1995).

Padrões são uma técnica que potencializa a reutilização de código. Padrões não garantem reutilização de código, mas servem como modelos que ajudam desenvolvedores a estruturar problemas.

Princípio de POO aplicados a padrões

- **Encapsulamento**: um padrão encapsula uma solução bem definida.
- **Generalização**: deve ser possível a construção de várias realizações a partir deste padrão.
- **Abstração**: os padrões representam abstrações do conhecimento estrutural.
- **Abertura**: um padrão deve ser aberto para extensão.
- **Combinatoriedade**: é possível se relacionar padrões ou compor padrões hierarquicamente para tratar de problemas de maior nível de detalhes.

Estrutura de um padrão de projeto

- **Nome**: ilustra como é arquitetada a proposta de solução. Exemplos de nomes: Conjunto unitário (Singleton), Fábrica abstrata (Abstract Factory), etc.
- **Problema**: A definição do problema que se pretende resolver. Em uma frase apresenta as restrições a serem atendidas.
- **Contexto**: São as situações em que o padrão se aplica.
- **Solução**: Relacionamentos entre as entidades.
- **Exemplo**: Diagramas que ilustram a aplicação da solução.

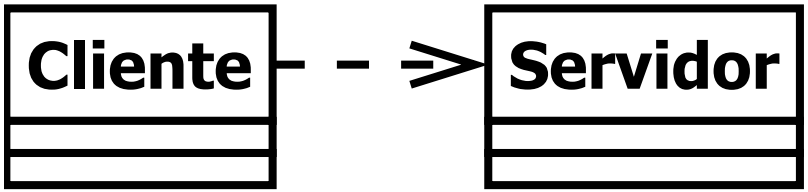
Reuso por herança (mecanismos de herança)

- **Extensão** (*extends*): subclasse estende a superclasse, acrescentando novos atributos e métodos.
  Classe provê implementação de métodos e atributos.
- **Implementação** (*implements*): interface define um padrão de comportamento que deve ser implementado pela subclasse.
  Interface provê assinaturas de métodos e constante.
- **Polimórfica de inclusão** (*override*): funciona como a extensão, mas a superclasse pode prover padrões de comportamento através de métodos abstratos (*abstract*), que devem ser especificados nas subclasses.
  Classe provê atributos, implementação de métodos e assinaturas de métodos abstratos.

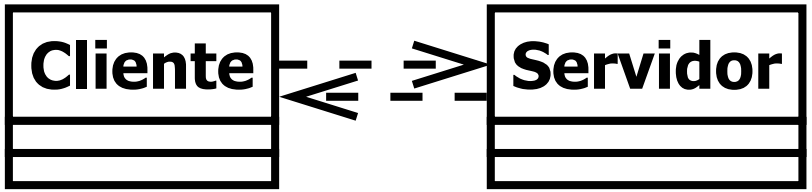
Reuso por delegação

- Uma classe/objeto utiliza serviços de outro através de relações entre esses objetos. Por exemplo: Associação, Agregação, Composição.
- Relações de uso produzem acoplamento:
  - Acoplamento pode ser fraco (unidirecional) ou forte (bidirecional).
  - Acoplamento concreto: remetente deve ter conhecimento da classe do destinatário.
  - Acoplamento abstrato: remetente não tem conhecimento da classe do destinatário. Implementado com classes abstratas e interfaces.

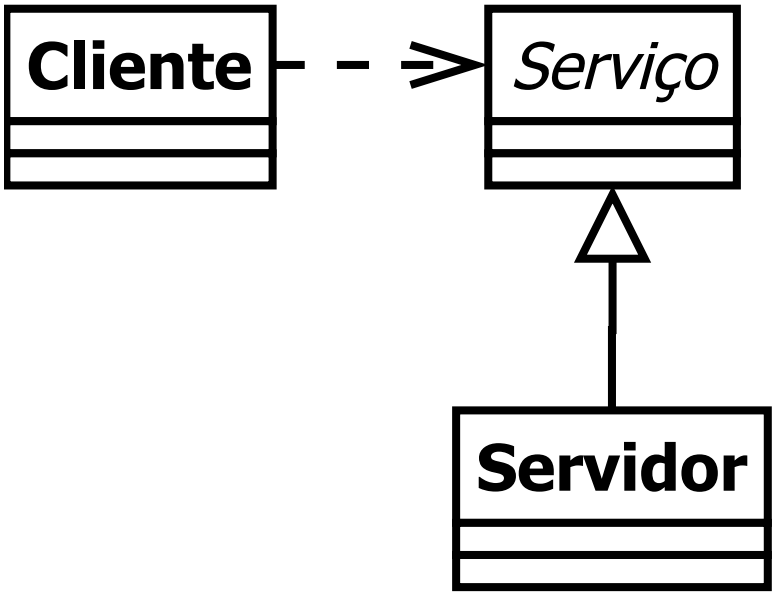
Exemplos de acoplamento



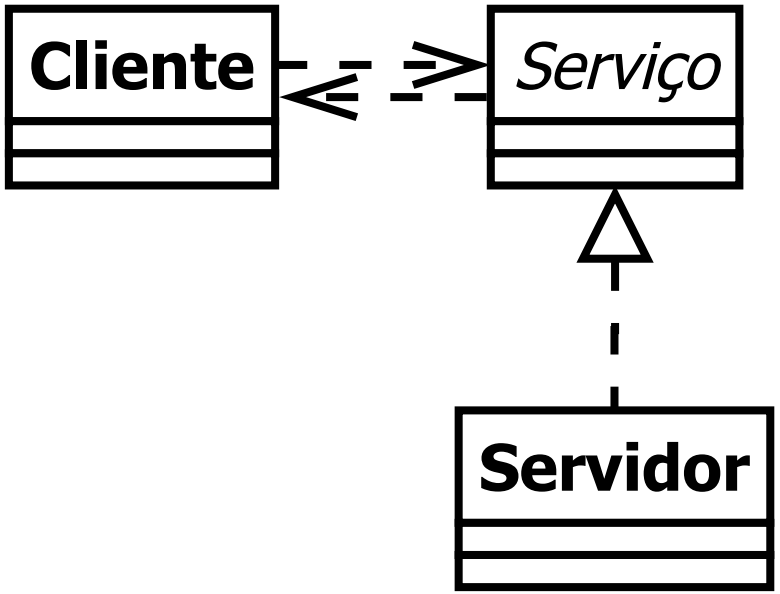
Acoplamento concreto fraco



Acoplamento concreto forte



Acoplamento abstrato por classe abstrata

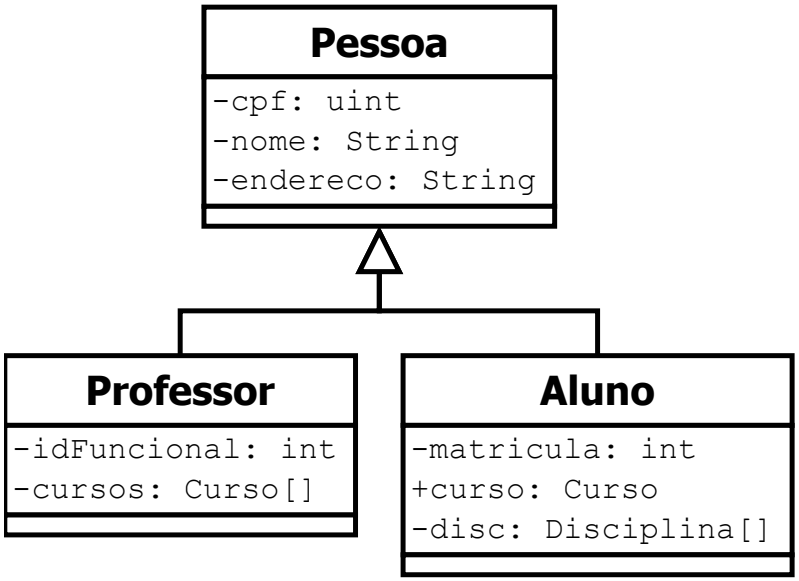


Acoplamento abstrato por interface

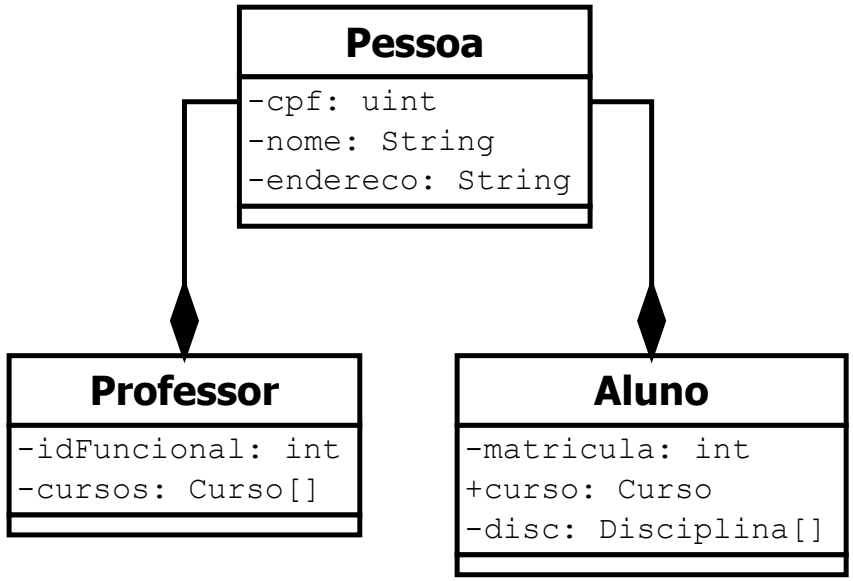
Formas de reuso: resumo

- **Reuso por herança**: subclasses herdam comportamento de superclasses.
  - é mais fácil de implementar.
  - porém subclasse pode precisar acessar aspectos específicos da superclasse (viola o princípio da ocultação da informação).
  - Pode violar o LSP (Princípio da Substituição de Liskov).
- **Reuso por delegação**: se um objeto não pode realizar uma operação, ele delega para outro objeto.
  - Objeto reusa operações de outro sem precisar ser subclasse do segundo.

Formas de reuso: exemplo



Reuso por herança



Reuso por delegação

Categorias de padrões em relação ao reuso

- Dois tipos de escopo:
  - **Escopo de classe**: utilizam herança para compor ou variar objetos.
  - **Escopo de objeto**: utilizam delegação para delegar suas responsabilidades para outro objeto.
- Classificação segundo Gamma et. al. (1995)
  - **Criacionais**: abstraem o processo de criação de objetos. relacionados com inicialização e configuração de objetos.
  - **Estruturais**: tratam do desacoplamento entre a interface e a implementação de objetos. Relacionados à composição de objetos.
  - **Comportamentais**: definem mecanismos de colaboração entre objetos. Distribuem as responsabilidades e padronizam as comunicações.

Classificação dos padrões (Gamma et. al., 1995)

		Propósito		
		Criacional	**Estrutural**	**Comportamental**
Escopo	**Classe**	Factory Method	Class Adapter	Interpreter Template Method
	Objeto	Abstract Factory Builder Prototype Singleton	Object Adapter Bridge Composite Decorator Facade Flyweight Proxy	Chain of responsibility Command Iterator Mediator Memento Observer State Strategy Visitor