9

TEXTO BASE

ANÁLISE E MODELAGEM DE SISTEMAS

Texto base

9

O.O. Técnicas

Interfaces e Padrões de Projetos

Prof. Renato de Tarso Silva

Resumo

Esta aula apresenta técnicas da orientação a objetos que permitem maior viabilidade na aplicação de conceitos reais em ambientes de software. Elas facilitam a manutenção da coesão sistêmica, e melhoram a organização e a colaboração de abstrações dos cenários a serem implementados.

9.1. Interfaces

Uma interface é uma representação de um elemento (conceito/entidade) com declarações públicas das propriedades deste determinado elemento. Tem a responsabilidade de formar um conjunto coerente de características e comportamentos de vários tipos de objetos que contenham aspectos comuns entre si. Fazendo uma analogia com a construção civil, imagine projetar uma casa que exigisse quebrar a fundação sempre que fosse necessário erguer uma parede, ou um apartamento no qual seria necessário mudar toda a fiação para se substituir uma tomada. Em termos de software, deve-se cuidar da estruturação com clara separação de conceitos.

De acordo com BOOCH (2005),

> As interfaces definem uma linha entre a especificação do que uma abstração realiza e a implementação do modo como isso é realizado pela abstração. Uma interface é uma coleção de operações utilizadas para especificar um serviço de uma classe ou de um componente.

O uso de interfaces é uma forma de alcançar um bom grau de separação, pois consiste em especificar “conexões” claras no sistema, estabelecendo partes que poderão ser modificadas de maneira independente. Escolhendo interfaces corretas, pode-se utilizar componentes-padrão em bibliotecas e frameworks na implementação,

dispensando sua construção. À medida se que descobre implementações melhores, pode-se substituir antigas aplicações sem afetar componentes ou interdependências.

Imagine uma loja que revenda 15 marcas de TVs, e cada marca contém 5 a 10 modelos. Poderiam ser necessários 50 controles remotos diferentes, um para cada modelo, ou utilizar apenas um controle remoto que saiba emitir a instrução que todos os modelos entendam. Note na figura 9.1.1., um exemplo de interface bastante conhecido e útil no dia-dia das pessoas, o controle remoto universal.

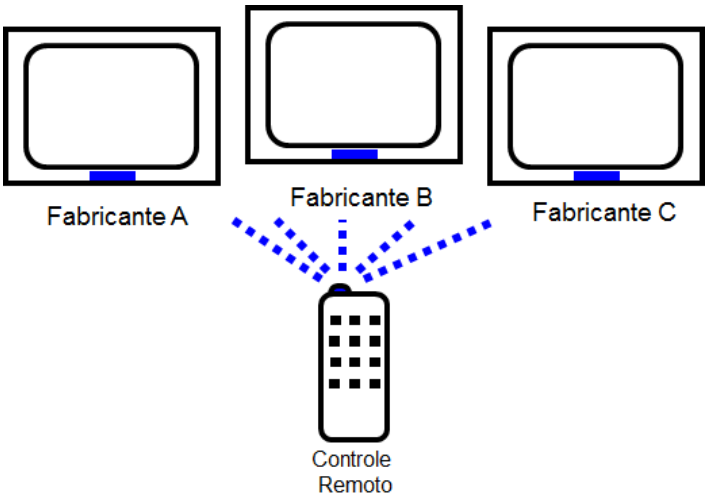

Figura 9.1.1. Interface - Controle remoto universal

Uma interface define uma espécie de contrato entre fornecedor e consumidor de objetos, que estabelece o que pode oferecer mas não sabe como proceder. É sempre uma classe abstrata, de modo que suas funcionalidades são realizadas por outras classes. Pode também ser utilizada com intuito de concentrar responsabilidades, tendo o poder de responder solicitações de várias classes, e que são processadas por outras classes.

A intimidade com o conceito de interfaces instiga o estudo de padrões conhecidos de associações entre classes que funcionam como uma poderosa caixa de ferramentas para implementação de conceitos complexos, ou repetitivos, no software.

9.2. Padrões de Projetos

Um padrão é uma solução comum para um problema básico em um determinado contexto, como um mecanismo aplicado a uma sociedade de classes, muito comumente utilizado em frameworks. Um framework é um padrão de arquitetura que fornece templates extensíveis a aplicações dentro de um domínio de sistemas.

> Todos os sistemas bem estruturados são repletos de padrões. [...] Se estiver projetando um novo sistema ou desenvolvendo um já existente, você nunca realmente estará começando do nada. Em vez disso, a experiência e as convenções o levarão a aplicar formas comuns para resolver problemas básicos.

Os padrões de projeto são comuns na UML, partes importantes do vocabulário dos desenvolvedores. Criar modelos com bons padrões explicitados em seu sistema

torna seu sistema mais compreensível, fácil de desenvolver e de manter. Veja na figura 9.2.1. uma representação dos padrões de projetos definidos pelo GoF (Gang of Four).

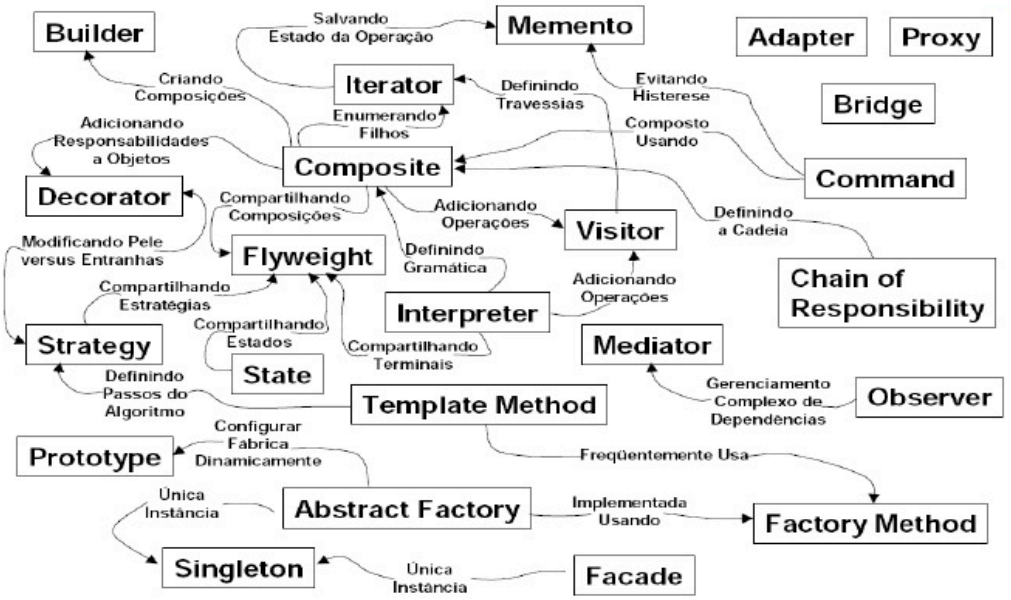

Figura 9.2.1. Padrões de Projeto GoF - *Design Patterns*

Existem três tipos básicos de padrões de projetos estabelecidos pelo GoF:

- Criacional: Criação de Objetos;
- Estrutural: Composição de Objetos;
- Comportamental: Interação entre Objetos.

Veja na tabela 9.2.2. a alocação de cada padrão de projeto entre estes três tipos:

		Propósito		
		Criação	Estrutura	Comportamento
Escopo	Classe	Factory Method	Class Adapter	Interpreter Template Method
	Objeto	Abstract Factory	Object Adapter	Chain of Responsibility
		Builder	Bridge	Command
		Prototype	Composite	Iterator
		Singleton	Decorator	Mediator
			Facade	Memento
			Flyweight	Observer
			Proxy	State
				Strategy
				Visitor

Tabela 9.2.2. Tipos Padrões de Projeto

Apesar de existirem 24 padrões de projeto conhecidos, para a didática serão abordados três dos mais usados. Neste link é possível conhecer melhor todos os padrões de projetos desenvolvidos pelo GoF.

9.2.1. Padrão de Projeto Singleton

Ao aplicar este padrão de projeto, uma classe proverá apenas 1 instância, um ponto global para o tipo de objeto. Isso sempre acontecerá independentemente de quantos e quais objetos solicitam uma instância a esta classe.

Imagine que você necessita que uma, e somente uma, instância de determinada classe seja possível de ser criada, por exemplo, uma conexão com banco de dados (BD). Suponha que há diversas referências à conexão de BD na execução de um código, instanciar todas as vezes a classe de BD acarretará em grande perda de desempenho e possibilitará inconsistência de dados. Usando o padrão Singleton, é garantido que nesta execução será instanciada a classe somente uma vez, por apenas um único objeto.

A classe representada na figura 9.2.1.1. exemplifica a modelagem deste padrão de projeto, no qual o atributo singleton é um objeto do tipo Singleton, uma auto instância.

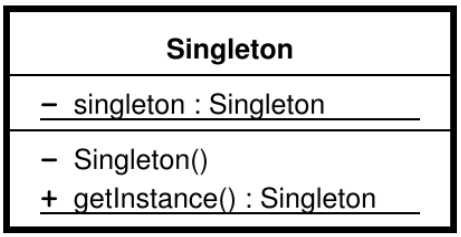

Figura 9.2.1.1. Classe Singleton

Na figura 9.2.1.2., pode-se notar como o mecanismo acontece. Uma solicitação é aceita e o retorno vai ser a própria instância existente, ou uma nova instância, somente caso já não exista uma instância deste tipo de objeto.

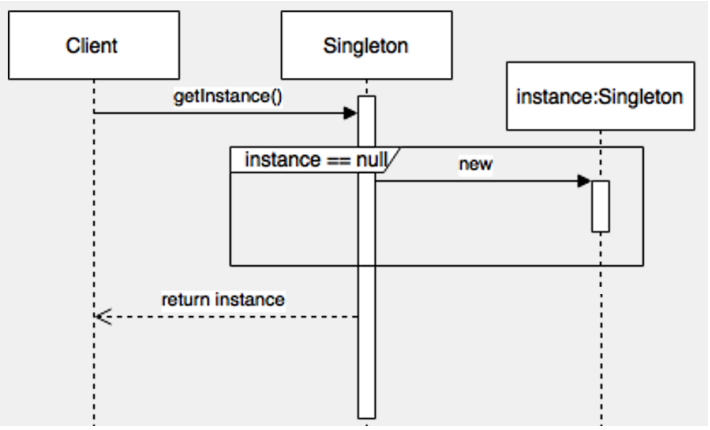

Figura 9.2.1.2. Padrão de Projeto Singleton - Sequência

9.2.2. Padrão de Projeto Façade

Por definição, o padrão Facade fornece uma interface unificada para um conjunto de interfaces em um subsistema. Ele define uma interface de nível mais alto, disponível a outros objetos, o que facilita a utilização dos recursos de determinado subsistema e seus componentes. O Padrão de Projeto Facade serve para ocultar a complexidade de uma ou mais classes através de uma espécie de Fachada (Facade). A intenção desse Padrão de Projeto é expor acesso a funcionalidades de várias entidades por meio de uma interface.

Note a figura 9.2.2,1. um exemplo de modelagem deste padrão de Projeto. A classe *ShapeMaker* serve de interface, pois ela conhece as operações necessárias para a criação de cada forma (*Circle, Rectangle, Square*). Assim, ela pode utilizar a implementação de cada forma (*Shape*), independente de qual tipo de forma é solicitada por outros objetos.

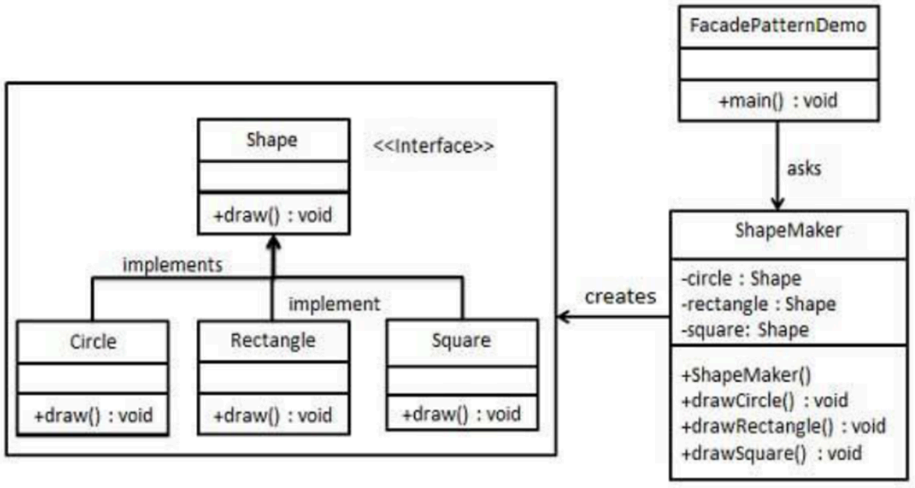

Figura 9.2.2.1. Padrão de Projeto Façade - Formas

9.2.3. Padrão de Projeto Abstract Factory

Uma interface que permite criar uma família de objetos relacionados/dependentes sem especificar classe concreta. O padrão isola classes concretas, encapsulando a responsabilidade e o processo de criação de objetos de produtos abstratamente relacionados. A figura 9.2.3.1 traz um exemplo deste padrão de projeto, em que as Classes *FábricaDeCarro* contêm as operações abstratas necessárias para *criarCarroSedan* e para *criarCarroPopular*, independentemente da Marca ou Modelo, pois as implementações são feitas pelas classes instanciadas abstratamente.

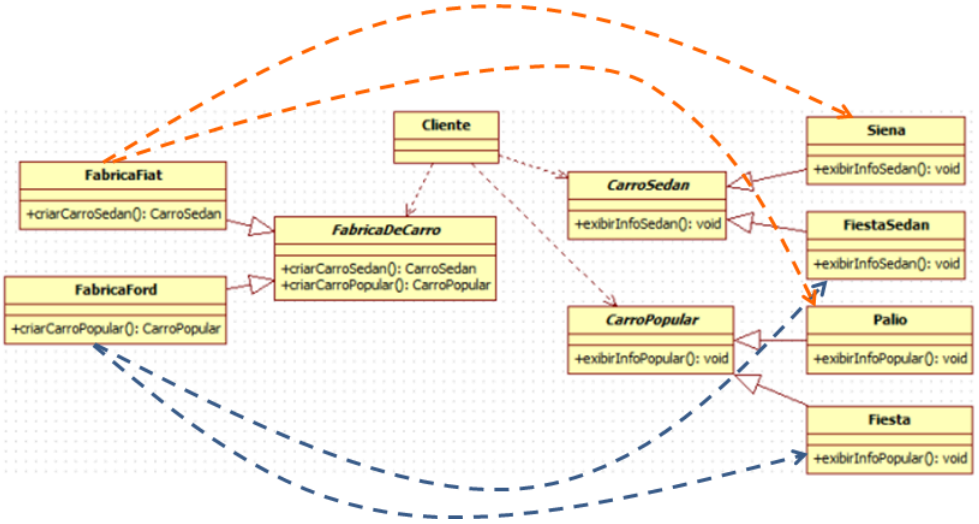

Figura 9.2.3.1. **Padrão de Abstract Factory - Fábrica de Carros**

Referências

Booch G., Rumbaugh J., Jacobson I. “The Unified Modeling Language user Guide” 2ª Edição. 2005.