PADRÕES DE PROJETOS

PROF. DEMÉTRIUS DE CASTRO

PROF2303@IESP.EDU.BR

83 9 8773-0383

WWW.DEMETRIUSDECASTRO.COM.BR

O QUE É PADRÃO DE PROJETO?

São soluções típicas para problemas comuns em projeto de software. Eles são como plantas de obra pré-fabricadas que você pode customizar para resolver um problema de projeto recorrente em seu código.

Você não pode apenas encontrar um padrão e copiá-lo para dentro do seu programa, como você faz com funções e bibliotecas que encontra por aí. O padrão não é um pedaço de código específico, mas um conceito geral para resolver um problema em particular. Você pode seguir os detalhes do padrão e implementar uma solução que se adeque às realidades do seu próprio programa.

O QUE É PADRÃO DE PROJETO?

Vamos tentar entender sob outra perspectiva: imagine você no futuro, daqui 15 anos, certamente terá passado por N problemas, repetidamente e em diferentes projetos. A tendência natural é que você crie soluções comuns para resolvê-los de tal forma que você consiga aplicá-las no futuro em outros projetos. Essas soluções, criadas por você, mesmo que não compartilhadas à literatura (livros ou artigos acadêmicos), podem ser consideradas como sendo Padrões de projetos, sim, padrões de projeto criados por você para resolver os seus problemas

O QUE É PADRÃO DE PROJETO?

Os padrões são frequentemente confundidos com algoritmos, porque ambos os conceitos descrevem soluções típicas para alguns problemas conhecidos. Enquanto um algoritmo sempre define um conjunto claro de ações para atingir uma meta, um padrão é mais uma descrição de alto nível de uma solução. O código do mesmo padrão aplicado para dois programas distintos pode ser bem diferente.

O QUE É PADRÃO DE PROJETO?

Uma analogia a um algoritmo é que ele seria uma receita de comida: ambos têm etapas claras para chegar a um objetivo. Por outro lado, um padrão é mais como uma planta de obras: você pode ver o resultado e suas funcionalidades, mas a ordem exata de implementação depende de você.

DO QUE CONSISTE UM PADRÃO DE PROJETO?

A maioria dos padrões são descritos formalmente para que as pessoas possam reproduzi-los em diferentes contextos. Aqui estão algumas seções que são geralmente presentes em uma descrição de um padrão:

- O **Propósito** do padrão descreve brevemente o problema e a solução.
- A **Motivação** explica a fundo o problema e a solução que o padrão torna possível.
- As **Estruturas** de classes mostram cada parte do padrão e como se relacionam.
- **Exemplos de código** em uma das linguagens de programação populares tornam mais fácil compreender a ideia por trás do padrão.

Alguns catálogos de padrão listam outros detalhes úteis, tais como a aplicabilidade do padrão, etapas de implementação, e relações com outros padrões.

HISTÓRIA DOS PADRÕES

Quem inventou os padrões de projeto?

Essa é uma boa pergunta, mas não muito precisa. Os padrões são soluções típicas para problemas comuns em projetos orientados a objetos. Quando uma solução é repetida de novo e de novo em vários projetos, alguém vai eventualmente colocar um nome para ela e descrever a solução em detalhe. É basicamente assim que um padrão é descoberto.

HISTÓRIA DOS PADRÕES

O conceito de padrões foi primeiramente descrito por Christopher Alexander em Uma Linguagem de Padrões. O livro descreve uma “linguagem” para o projeto de um ambiente urbano. As unidades dessa linguagem são os padrões. Eles podem descrever quão alto as janelas devem estar, quantos andares um prédio deve ter, quão largas as áreas verdes de um bairro devem ser, e assim em diante.

HISTÓRIA DOS PADRÕES

A ideia foi seguida por quatro autores: Erich Gamma, John Vlissides, Ralph Johnson, e Richard Helm. Em 1994, eles publicaram Padrões de Projeto — Soluções Reutilizáveis de Software Orientado a Objetos, no qual eles aplicaram o conceito de padrões de projeto para programação. O livro mostrava 23 padrões que resolviam vários problemas de projeto orientado a objetos e se tornou um best-seller rapidamente. Devido a seu longo título, as pessoas começaram a chamá-lo simplesmente de "o livro da Gangue dos Quatro (Gang of Four)" que logo foi simplificado para o "livro GoF".

HISTÓRIA DOS PADRÕES

Desde então, dúzias de outros padrões orientados a objetos foram descobertos. A “abordagem por padrões” se tornou muito popular em outros campos de programação, então muitos desses padrões agora existem fora do projeto orientado a objetos também.

POR QUE DEVO APRENDER PADRÕES DE PROJETO?

A verdade é que você pode conseguir trabalhar como um programador por muitos anos sem saber sobre um único padrão. Muitas pessoas fazem exatamente isso. Ainda assim, contudo, você estará implementando alguns padrões mesmo sem saber. Então, por que gastar tempo para aprender sobre eles?

POR QUE DEVO APRENDER PADRÕES DE PROJETO?

- Os padrões de projeto são um kit de ferramentas para **soluções tentadas e testadas** para problemas comuns em projeto de software. Mesmo que você nunca tenha encontrado esses problemas, saber sobre os padrões é ainda muito útil porque eles ensinam como resolver vários problemas usando princípios de projeto orientado a objetos.

POR QUE DEVO APRENDER PADRÕES DE PROJETO?

- Os padrões de projeto definem uma linguagem comum que você e seus colegas podem usar para se comunicar mais eficientemente. Você pode dizer, “Oh, é só usar um Singleton para isso,” e todo mundo vai entender a ideia por trás da sua sugestão. Não é preciso explicar o que um singleton é se você conhece o padrão e seu nome.

ALGUMAS CRÍTICAS AO USO DE PADRÕES DE PROJETOS

Gambiarras para uma linguagem de programação fraca

Geralmente a necessidade de padrões surge quando as pessoas escolhem uma linguagem de programação ou uma tecnologia que tem uma deficiência no nível de abstração. Neste caso, os padrões se transformam em gambiarras que dão à linguagem superpoderes muito necessários.

Por exemplo, o padrão **Strategy** pode ser implementado com uma simples função anônima (lambda) na maioria das linguagens de programação moderna.

ALGUMAS CRÍTICAS AO USO DE PADRÕES DE PROJETOS

Uso injustificável

Se tudo que você tem é um martelo, tudo parece com um prego.

Esse é o problema que assombra muitos novatos que acabaram de se familiarizar com os padrões. Tendo aprendido sobre eles, tentam aplicá-los em tudo, até mesmo em situações onde um código mais simples seria suficiente.

CLASSIFICAÇÃO DOS PADRÕES

Padrões de projeto diferem por sua complexidade, nível de detalhes, e escala de aplicabilidade ao sistema inteiro sendo desenvolvido. Eu gosto da analogia com a construção de uma rodovia: você sempre pode fazer uma intersecção mais segura instalando algumas sinaleiras ou construindo intercomunicações de vários níveis com passagens subterrâneas para pedestres.

Os padrões mais básicos e de baixo nível são comumente chamados *idiomáticos*. Eles geralmente se aplicam apenas à uma única linguagem de programação.

CLASSIFICAÇÃO DOS PADRÕES

Os padrões mais universais e de alto nível são os *padrões arquitetônicos*; desenvolvedores podem implementar esses padrões em praticamente qualquer linguagem. Ao contrário de outros padrões, eles podem ser usados para fazer o projeto da arquitetura de toda uma aplicação.

Além disso, todos os padrões podem ser categorizados por seu propósito, ou intenção e normalmente são classificados em três grupos principais de padrões:

CLASSIFICAÇÃO DOS PADRÕES

- **Padrões criacionais** fornecem mecanismos de criação de objetos que aumentam a flexibilidade e a reutilização de código.
- P**adrões estruturais** explicam como montar objetos e classes em estruturas maiores, enquanto ainda mantém as estruturas flexíveis e eficientes.
- P**adrões comportamentais** cuidam da comunicação eficiente e da assinalação de responsabilidades entre objetos.

S.O.L.I.D

O SOLID é uma sigla em inglês para cinco princípios de projeto que possuem o objetivo de fazer programas mais compreensíveis, flexíveis e sustentáveis.

Não faz parte dos padrões de projeto, mas por serem boas práticas que garantem diversos benefícios é bem comum de aprender ambos os temas ao mesmo tempo.

S.O.L.I.D

- “S” Single Responsibility Principle (Princípio de responsabilidade única): uma classe deve ter uma e apenas uma razão.
- “O” Open-Closed Principle (Princípio aberto/fechado): objetos devem estar disponíveis para extensão, mas fechados para modificação.
- “L” Liskov Substitution Principle (Princípio de substituição de Liskov): uma subclasse deve ser substituível por sua superclasse.
- “I” Interface Segregation Principle (Princípio de segregação de interface): uma classe não deve ser obrigada a implementar métodos e interfaces que não serão utilizadas.
- “D” Dependency Inversion Principle (Princípio de inversão de dependência): dependa de abstrações e não de implementações.

PADRÕES QUE ESTUDAREMOS

GRASP

GOF

GRASP

Os padrões GRASP (General Responsibility Assignment Software Patterns) consistem em uma série de princípios baseados em conceitos para atribuição de responsabilidades a classes e objetos na construção de bons softwares usando programação orientada a objetos. Mesmo estando caindo em desuso, ainda é comum encontrar empresas fazendo uso desse padrão. Ele é dividido em nove padrões, sendo eles:

GRASP

- **Information Expert** (Especialista na Informação)
- **Creator** (Criador)
- **Controller** (Controlador)
- **Low Coupling** (Baixo Acoplamento)
- **High Cohesion** (Alta Coesão)
- **Polymorphism** (Polimorfismo)
- **Pure Fabrication** (Pura Fabricação)
- **Indirection** (indireção)
- **Protected Variations** (Proteção contra Variações)

GOF

Tem como objetivo abstrair a instanciação de objetos. Com eles, o sistema vai solicitar um objeto de um determinado tipo e o terá prontinho, sob demanda, sem nem se preocupar com as nuances da criação. Fazendo um paralelo com o mundo real, uma empresa automobilística quando precisa de amortecedores, ela terceiriza (solicita-os) e então os instala em seus carros, sem se preocupar com o todo envolvido na criação desse componente. Ele é dividido em 23 padrões, sendo eles:

GOF

PADRÕES DE CRIAÇÃO

- Singleton (Único)
- Factory Method (Método de fábrica)
- Abstract Factory (Fábrica abstrata)
- Builder (Construtor)
- Prototype (Protótipo)

GOF

PADRÕES ESTRUTURAIS

- Adapter (Adaptador)
- Bridge (Ponte)
- Composite (Composto)
- Decorator (Decorador)
- Facade (Fachada)
- Flyweight (Peso mosca)
- Proxy

GOF

PADRÕES COMPORTAMENTAIS

- Observer (Observador)
- Strategy (Estratégia)
- Command (Comando)
- Chain of Responsibility (Corrente de Responsabilidade)
- Interpreter (Intérprete)
- Mediator (Mediador)

GOF

PADRÕES COMPORTAMENTAIS

- Memento (Memorando)
- State (Estado)
- Template Method (Método de Modelo)
- Visitor (Visitante)
- Iterator (Iterador)

GOF

C	Abstract Factory
S	Adapter
S	Bridge
C	Builder
B	Chain of Responsibility
B	Command
S	Composite
S	Decorator
S	Facade
C	Factory Method
S	Flyweight
B	Interpreter
B	Iterator
B	Mediator
B	Memento
C	Prototype
S	Proxy
B	Observer
C	Singleton
B	State
B	Strategy
B	Template Method
B	Visitor

REVISÃO

Antes de entrarmos no estudo do Padrões de Projetos, precisamos revisar alguns conceitos importantes:

- HERANÇA
- COMPOSIÇÃO
- INTERFACES
- POLIMORFISMO

REVISÃO: EXERCÍCIO I

- Escreva um código que receba um valor e retorne esse valor em + (linhas e colunas), começando pelo menor número para o maior.
- Ex.:
  - Valor Digitado: 5
- Resultado:
  - +
  - ++
  - +++
  - ++++
  - +++++

REVISÃO: EXERCÍCIO II

- Escreva um código que receba 3 valores inteiro < 11 e > -1, faça a sua soma, retire a média e informe: se o valor for < 4, “REPROVADO”, se for > 3 e < 7, “PROVA FINAL”, se for >= 7, “APROVADO”.

REVISÃO: EXERCÍCIO III

- Escreva um código que receba 2 valores inteiro e uma string (+, -, *, /). De acordo com a string, realize a operação.