**Instituto Federal Catarinense – Campus Rio do Sul**

**Disciplina: CCC0627 Programação Orientada a Objetos II - 2022.2**

**Docente: Rodrigo Curvello**

**Acadêmico: João Vitor Lehmen Sanmartin**

**Data: 11/12/2022**

**Resumo**

Os padrões de projeto são arquiteturas comprovadas para construir software orientado a objetos (dos Santos, Jefersson Alex. "Padrões de Projeto.")

Design patterns são como plantas pré-projetadas de uma construção, que você pode alterar para se adequar melhor na resolução de um problema recorrente em seu código. O que diferencia os padrões de projeto das funções e bibliotecas é que você não pode simplesmente copiá-los direto para seu programa, já que eles não são um pedaço de código, mas sim um conceito que serve como uma solução. (Chagas, 2022)

Esses padrões capturam soluções que foram desenvolvidas e aperfeiçoadas ao longo do tempo, e facilitam a reutilização e flexibilidade nos sistemas de software. Eles não exigem nenhum recurso incomum da linguagem, nem truques de programação. Todos podem ser implementados em linguagens orientadas a objetos comuns. (Gamma, 2009.)

Assim, para se implementar um padrão de projeto você deve seguir o conceito dos padrões escolhidos e ajustá-lo ao problema que deseja resolver. Isso dependerá das características do projeto. Dessa forma, fazemos uma implementação que se encaixe exatamente à demanda da nossa aplicação. (Chagas, 2022)

**Introdução**

Os padrões de projetos servem para construir softwares confiáveis com arquiteturas já comprovadas, promovendo reutilização de projeto e ajudando a identificar erros e armadilhas comuns no dia a dia de um desenvolvedor. Também servem para estabelecer um vocabulário comum entre os desenvolvedores e encurtam as fases do projeto. (dos Santos, Jefersson Alex. "Padrões de Projeto.")

Christopher Alexander afirma: “cada padrão descreve um problema no nosso ambiente e o cerne da sua solução, de tal forma que você possa usar essa solução mais de um milhão de vezes, sem nunca o fazer da mesma maneira”. Muito embora Alexander estivesse falando acerca de padrões em construções e cidades, o que ele diz é verdadeiro em relação aos padrões de projeto orientados a objetos. Nossas soluções são expressas em termos de objetos e intefaces em vez de paredes e portas, mas ao cerne de ambos os tipos de padrões está a solução para um problema num determinado contexto. (Gamma, 2009.)

Em geral, um padrão de projeto tem quatro elementos essenciais. O nome do padrão, é uma referência que podemos usar para descrever um problema do projeto, suas soluções e consequências em uma ou duas palavras. Ter um nome torna mais fácil pensar sobre projetos e comunicá-los. (Gamma, 2009.)

O problema que descreve em que situação aplicar o padrão, isso explica o contexto do problema. Algumas vezes o problema incluirá uma lista de condições que devem ser satisfeitas para que faça sentido aplicar certo padrão. (Gamma, 2009.)

A solução que descreve os elementos que compõe o padrão de projeto, seus relacionamentos, suas responsabilidades e colaborações. A mesma descreve um projeto concreto ou uma interpretação em particular. (Gamma, 2009.)

As consequências que descrevem os resultados e análises das vantagens e desvantagens aplicadas do padrão. Elas frequentemente envolvem balanceamento entre espaço e tempo. E podem abordar sobre aspectos sobre linguagens e implementação. Relacionar essas consequências explicitamente ajudam a compreendê-las e avalia-las. (Gamma, 2009.)

**Desenvolvimento**

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 designs patterns, mas por serem boas práticas que garantem diversos benefícios é bem comum de aprender ambos os temas ao mesmo tempo. (Chagas, 2022)

“S” Single Responsibility Principle (Princípio de responsabilidade única): uma classe deve ter uma e apenas uma razão para mudar. (Chagas, 2022)

“O” Open-Closed Principle (Princípio aberto/fechado): objetos devem estar disponíveis para extensão, mas fechados para modificação. (Chagas, 2022)

“L” Liskov Substitution Principle (Princípio de substituição de Liskov): uma subclasse deve ser substituível por sua superclasse. (Chagas, 2022)

“I” Interface Segregation Principle (Princípio de segregação de interface): uma classe não deve ser obrigada a implementar métodos e interfaces que não serão utilizadas. (Chagas, 2022)

“D” Dependency Inversion Principle (Princípio de inversão de dependência): dependa de abstrações e não de implementações. (Chagas, 2022)

Os três principais padrões de projeto definidos pelo livro “Design Patterns: Elements of Reusable Object-Oriented Software” de 1994, escrito por GOF (Gang of Four: Rich Gamma, Richard Helm, Ralph Johnson e John Vlissides) são os padrões criacionais, estruturais e comportamentais. Esses padrões foram divididos e agrupados de acordo com a natureza do problema que eles solucionam. (Chagas, 2022)

Padrões criacionais: estes padrões oferecem diversas alternativas de criação de objetos, o que aumenta a flexibilidade e a reutilização de código. Alguns dos principais padrões desse tipo são: (Chagas, 2022)

- Factory Method: Define uma interface para criar um objeto, mas deixa as subclasses decidirem qual classe a ser instanciada. Permite a uma classe postergar a instanciação. (Gamma, 2009.)

- Abstract Factory: Fornece uma interface para a criação de famílias de objetos relacionados ou dependentes sem especificar suas classes concretas. (Gamma, 2009.)

- Factory Builder

Padrões estruturais: Nos mostram como montar objetos e classes em estruturas maiores, sem perder a eficiência e flexibilidade. Alguns dos principais padrões desse tipo são: (Chagas, 2022)

- Adapter: Converte a interface de uma classe em outra interface esperada pelos clientes. O adapter permite que certas classes trabalhem em conjunto pois de outra forma seria impossível por causa de suas interfaces incompatíveis. (Gamma, 2009.)

- Bridge: Separa uma abstração de sua implementação, de modo que as duas possam variar independentemente. (Gamma, 2009.)

- Composite: Compõe objetos em estrutura de árvores para representar hierarquias do tipo partes-todo. O Composite permite que os clientes tratem os objetos individuais e composições de objetos de maneira uniforme.

Padrões comportamentais: Nos ajudam a trabalhar melhor com os algoritmos e com a delegação de responsabilidades entre os objetos. Os padrões que se destacam nesse tipo são: (Chagas, 2022)

- Chain of Responsibility: Evita o acoplamento do remetente de uma solicitação ao seu destinatário, dando a mais de um objeto a chance de tratar a solicitação. Encadeia os objetos receptores e passa a solicitação ao longo da cadeia até que um objeto a trate. (Chagas, 2022)

- Command: Encapsula uma solicitação como um objeto, desta forma permitindo que você parametrize clientes com diferentes solicitações, enfileire ou registre solicitações e suporte operações que podem ser desfeitas. (Chagas, 2022)

- Interpreter: Dada uma linguagem, define uma representação para sua gramática juntamente com um interpretador que usa a representação para interpretar sentenças nessa linguagem. (Chagas, 2022)

**Referências:**

* Dos Santos, Jefferson. **Padrões de Projeto**. Disponível em: <https://homepages.dcc.ufmg.br/~jefersson/cursos/dcc052/Aula16.pdf> . Acesso em: 11/12/2022 às 20:40hrs.
* GAMMA, Erich. **Padrões de projetos: soluções reutilizáveis.** 2009. Disponível em:<https://books.google.com.br/books?hl=pt-BR&lr=&id=U91CYCqTCgkC&oi=fnd&pg=PA9&dq=padr%C3%B5es+de+projeto&ots=PZl_6wjLJN&sig=2gA_vd0HAtuahHj8b2FaulfOIUc#v=onepage&q=padr%C3%B5es%20de%20projeto&f=false> Acesso em: 11/12/2022 às 20:48hrs.
* CHAGAS, Igor. **Design patterns: Breve introdução aos padrões de projeto.** 2022. Disponível em: <https://www.alura.com.br/artigos/design-patterns-introducao-padroes-projeto>. Acesso em: 08/12/2022 às 15:14hrs.