Trabalho Complementar – POO

Padrões de Projeto

- **O que é um Padrão de Projeto ou Design Pattern?**

O dia a dia de um programador é comum enfrentar vários desafios de resolver problemas computacionais, por exemplo desenvolver uma aplicação, atualizar projetos, resolver bugs e entre outras situações. Com isso foram desenvolvidas teorias para a resolução dos problemas comuns na programação, as teorias foram escritas por ***Erich Gamma, Richasrd Helm, Ralph Johnson e John Vlissides***, que deu a origem ao livro chamado "***Design Patterns: Elements of Reusable Object-Oriented Software***". Os autores ficaram conhecidos como a ***Guangue dos Quatro(Gang of Four),*** e tais padrões começaram a ser intitulados de ***Padrões GoF(GoF Patterns)***.

Resumindo um Padrão de Projeto, são soluções prontas para um problema computacional que já foram solucionados por outras pessoas.

- **Qual é a importância e quando trabalhar com os Padrões de Projetos ?**

Imagine que você está desenvolvendo um software em que você tem que solucionar vários problemas para chegar no seu objetivo final, agora vamos supor que você está empacado em um paradigma do seu projeto, assim que você começa a pesquisar como resolver esse e se depara que seu problema já foi solucionados por outras pessoas, e isso te ajudar a entender melhor o seu problema e como resolvê-lo. Assim como a matemática e a física existem fórmulas para resolver os problemas, no mundo do desenvolvimento não foi diferente, os Design Patterns trazem "fórmulas" para resoluções de problemas, e com isso nós programadores, conseguimos economizar tempo e dinheiro, a construção de um código mais limpo e refinado sem precisar ficar reinventado a roda, assim facilitando o nosso trabalho.

- **Classificações de Padrões de Projetos e suas Aplicações.**

A Gangue dos Quatro em sua obra identificaram 23 padrões, que foram subdivididos em 3 grupos: Padrões de Criação, Padrões Estruturais e Padrões Comportamentais.

  - Padrões de Criação

Esse padrão está diretamente ligado a criação de objetos é o seu objetivo, é evitar problemas e proporcionar controle na criação, e diminuir a dependência dos construtores das classes e a abstração do processo de criação do objeto. Os padrões para implementação são *Factory Method, Singleton, Abstract Factory, Builder e Prototype*, cada um com suas particularidades, os dois que são julgados como principais são:

Factory Method → Tem como objetivo deixar a classe que implementa uma interface decidir o objeto que será criado.

Singleton → Esse padrão visa definir apenas uma instância para a classe, ou seja, apenas um objeto da classe poderá existir tornando-se um ponto de acesso global à classe.

- Padrões Estruturais

Esse padrão está diretamente ligado a estruturação de uma classe, o objetivo desse padrão é mostrar como podemos construir uma classe tendo suas heranças, sem perder sua eficiência e flexibilidade, ou seja, definir formas e adicionar comportamentos a um objeto sem alterá-lo diretamente. Os padrões para implementação são *Adapter, Decorator, Facade, Proxy, Composite, Flyweight e Bridge*, cada um com suas particularidades, os três que são julgados como principais são:

Adapter → Utilizado quando há a necessidade de encaixar uma nova biblioteca de classes a uma aplicação.

Decorator → Permitir que as responsabilidades sejam adicionadas e removidas de um objeto.

Facade **(Fachada)** → É uma classe que fornece uma interface para um subsistema. Uma fachada pode fornecer funcionalidades limitadas, entregado ao cliente apenas as funcionalidades desejadas.

- Padrões Comportamentais

Esse padrão tem como objetivo lidar sobre a forma que um objeto se comunica com os demais, sem depender uns dos outros, esse padrão nos ajuda a trabalhar com os algoritmos e com delegação de responsabilidades entre os objetos. Os padrões para implementação são *Mediator, Observer, Commandm Strategy, Tamplate Method, Interpreter, Visitor, Memento, State e Chain Of Responsability*, cada um com suas particularidades, os dois que são julgados como principais são:

Mediator → Ele atua como um mediador entre relacionamentos de objetos **N** *para* **N** *(Muito para Muitos)*, fazendo com que você reduz as dependências entre objetos.

Observer → Objetivo principal é definir um relacionamento de objetos **1** *para* **N** *(Um para Muitos)*. Todos os dependentes que relacionam com este objeto, irão ser notificados quando algum objeto mudar seu estado.

Em resumo, os Padrões de Projetos, também conhecidos como Design Patterns, representam soluções para problemas comuns no desenvolvimento de software. Esses padrões são uma coleção de práticas testadas e comprovadas que foram documentadas por especialistas para facilitar o desenvolvimento de sistemas eficientes, reutilizáveis e flexíveis na programação orientada a objetos.

Portanto, ao dominar e aplicar os Padrões de Projetos apropriados a cada tipo de problema, os desenvolvedores ganharam uma mão na roda para que possam aprimorar e evoluir, para que seja possível a construções de melhores softwares com as melhores praticas e código mais limpo, sem ter que reinventar a roda a cada novo tipo de projeto. Os Design Patterns representam uma valiosa ferramenta na vida de qualquer programador que busca excelência no desenvolvimento de software.