Trabalho L.P.O.O. (SOLID)

Nome: Rafael Alves Faria

Resumo

O presente trabalho tem o intuito de apresentar e explicar os cinco princípios de boas práticas da programação orientada a objetos e de design de código, conhecidos pelo acrônimo SOLID.

Palavras-chave: Programação Orientada a Objetos, SOLID, Boas Práticas, Design de Código.

Introdução

SOLID é um acrônimo criado por Michael Feathers, com o intuito de reunir os cinco princípios de programação orientada a objetos e de design de código propostos por Robert C. Martin (Uncle Bob), apresentados no artigo *"The Principles of OOD"*.

Esses princípios ajudam o programador a desenvolver um código mais limpo, organizado e sustentável, destacando a separação de responsabilidades, diminuindo o acoplamento, facilitando a manutenção e promovendo o reaproveitamento do código.

Princípios do SOLID

1. Single Responsibility Principle (Princípio da Responsabilidade Única)

Esse princípio afirma que uma classe deve ter apenas uma responsabilidade. Quando atribuímos muitas funções a uma mesma classe, criamos as chamadas *God Classes* (classes que concentram responsabilidades demais).

Isso pode parecer conveniente no início, mas no futuro dificulta a manutenção, pois qualquer alteração em um ponto pode gerar problemas em outros. Ao manter apenas uma responsabilidade por classe, o código se torna mais limpo, modular e de lógica mais harmônica.

2. Open-Closed Principle (Princípio Aberto-Fechado)

Segundo esse princípio, entidades de software (classes, módulos, funções) devem estar abertas para extensão, mas fechadas para modificação.

Isso significa que, ao adicionar novas funcionalidades, o ideal é estender o código existente em vez de reescrevê-lo, evitando que alterações prejudiquem partes já estáveis do sistema.

Uncle Bob aconselha: *"Separe o comportamento extensível por trás de uma interface e inverta as dependências."* Ou seja, boas abstrações tornam o código preparado para evoluções futuras sem quebrar o que já funciona.

3. Liskov Substitution Principle (Princípio da Substituição de Liskov)

Definido por Barbara Liskov em 1987, esse princípio estabelece que uma classe derivada deve poder substituir sua classe base sem alterar o funcionamento do programa.

Em resumo: *“Se S é subtipo de T, então objetos do tipo T podem ser substituídos por objetos do tipo S sem afetar o programa.”*

Erros comuns que violam esse princípio incluem sobrescrever métodos de forma inadequada, lançar exceções inesperadas ou retornar valores que não fazem sentido em relação à classe base. Esses problemas podem ser evitados com uma boa estrutura de código e, muitas vezes, com o uso de injeção de dependência.

4. Interface Segregation Principle (Princípio da Segregação da Interface)

Esse princípio afirma que nenhuma classe deve ser forçada a implementar métodos que não utiliza.

Em vez de criar interfaces genéricas e extensas, é melhor dividir em interfaces menores e específicas.

5. Dependency Inversion Principle (Princípio da Inversão da Dependência)

Esse princípio pode ser resumido nas palavras de Uncle Bob:

1. Módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações.
2. Abstrações não devem depender de detalhes. Detalhes devem depender de abstrações.

Ou seja, em vez de programar diretamente para implementações concretas, devemos programar para interfaces e abstrações, garantindo maior flexibilidade.

É importante não confundir esse princípio com Injeção de Dependência:

- O Princípio da Inversão de Dependência é um conceito de design, que orienta a depender de abstrações.
- A Injeção de Dependência é um padrão de implementação que transfere a criação de objetos para uma entidade externa.

Conclusão

Seguindo os princípios do SOLID, o programador consegue produzir softwares mais robustos, escaláveis e flexíveis, facilitando futuras manutenções e implementações. Esses conceitos ajudam a evitar problemas comuns de acoplamento excessivo, tornando o código mais limpo e de fácil evolução.

Referências

PAIXÃO, João Roberto da. *O que é SOLID: o guia completo para você entender os 5 princípios da POO*. Medium, 6 jan. 2019. Disponível em: https://medium.com/desenvolvendo-com-paixao/o-que-%C3%A9-solid-o-guia-completo-para-voc%C3%AA-entender-os-5-princ%C3%ADpios-da-poo-2b937b3fc530. Acesso em: 3 set. 2025.

“Uncle Bob”. *The OLD Object Mentor blog site*. Disponível em: http://butunclebob.com/. Acesso em: 4 set. 2025.

TADEU, Armando. *Como funciona a Injeção de Dependências no Spring*. Algaworks, 3 mar. 2021. Disponível em: https://blog.algaworks.com/injecao-de-dependencias-spring/. Acesso em: 4 set. 2025.