SOLID

O Que é SOLID ?

- “Bons sistemas de software começam com um código limpo. Por um lado, se os tijolos não são bem-feitos, a arquitetura da construção perde a importância. Por outro lado, você pode fazer uma bagunça considerável com tijolos bem-feitos” . Martin, Robert C.. Arquitetura Limpa (Robert C. Martin).
- SOLID não é uma tecnologia.
- SOLID não é um design pattern.
- SOLID não é algo opcional em sistemas grandes.
- SOLID é a orientação a escrita de código resiliente, sólido.
- SOLID é um acrônimo criado por Michael Feathers, Após observar os princípios descritos por Robert C. Martin (a.k.a. Uncle Bob) e abordados no artigo The Principles of OOD.

Formação do Acrônimo

- S - SRP Princípio da Responsabilidade Única (Single Responsibility Principle).
- O - OCP Princípio Aberto/Fechado (Open-Closed principle).
- L - LSP Princípio de Substituição de Liskov (Liskov Substitution Principle).
- I - ISP Princípio da Segregação de Interface (Interface Segregation Principle).
- D - DIP Princípio da inversão de Dependências (Dependency Inversion Principle).

SRP - Princípio da Responsabilidade Única

- “Qualquer organização que projeta um sistema (definido de forma mais ampla aqui do que apenas sistemas de informação) inevitavelmente produzirá um projeto cuja estrutura é uma cópia da estrutura de comunicação da organização” (lei de Conway).
- “Um módulo (Classe, Função, Método) deve ter uma, e apenas uma, razão para mudar”. (Definição amplamente difundida).
- “Um módulo deve ser responsável por um, e apenas um ator.” (Responsável pela alteração.)

OCP - Princípio de Aberto e Fechado

- “Um artefato de software deve ser aberto para extensão, mas fechadas para modificação.” Betrand Meyer, 1988
- Este princípio preza pela evolução do software sem alterações dos artefatos quanto a seus comportamentos, mas a extensão do código já existente.
- Quando um software tem um comportamento de constante mudança em seus artefatos, fatalmente esses artefatos estão destinados ao fracasso.
    - Salvo Correções de Erros.

LSP - O Princípio de Substituição de Liskov

- “Se q(x) é uma propriedade demonstrável dos objetos x de tipo T. Então q(y) deve ser verdadeiro para objetos y de tipo S onde S é um subtipo de T.”
- Uma classe base deve poder ser substituída por sua subclasse sem impacto ou mudança no comportamento.
- Exemplo: Se Andar como cachorro, latir como cachorro mas usar pilhas não é um cachorro.

ISP - Princípio da Segregação de Interface

- Uma classe não deve depender de mais itens do que ela realmente precisa.
- O princípio da segregação de interfaces auxilia na diminuição do nível de acoplamento do sistema.

DIP - Princípio da Inversão de Dependência

- Sistemas flexíveis são aqueles em que as dependências de código fonte se referem apenas a abstrações e não a itens concretos.
- Não dependa de implementações concretas e sim de suas abstrações.
- Regras deste princípio:
  - Não se refira a classes concretas voláteis.
  - Não derive de classes concretas voláteis;
  - Não sobrescreva funções concretas;