Nome: Alison Samuel Alves do Monte

Matrícula: 241012279

Curso: Ciência da Computação

Os princípios sólidos do modelo de Orientação a Objetos

- POO não é necessariamente a respeito de programar se baseando em conceitos do mundo real, na verdade é a respeito de lidar bem com dependências para a prevenção de códigos rígidos, frágeis ou desconfiáveis.
- Três termos para códigos desconfiáveis:
  - Rigidez: quando uma mudança em uma parte do código pode quebrar outra parte do código.
  - Fragilidade: quando as coisas deixam de funcionar sem estarem relacionadas.
  - Imobilidade: quando um código não pode ser reutilizado fora de seu contexto original.

- Primeiro princípio [S]OLID: SRP: Princípio da Responsabilidade Única:
  - Uma classe deve realizar uma tarefa específica.
  - Uma mudança em um motivo não deve afetar outro.
- Segundo princípio S[O]LID: OCP: Aberto / fechado (princípio)

L Você deve ser capaz de estender o comportamento de uma classe sem modificá-la.

L Você deve estender o seu código por subclasses para que a classe original não precise ser alterada.

- Lembrando que os princípios SOLID não são isolados e, muitas vezes, devem ser utilizados em conjunto.

• Terceiro Princípio SO**L**ID : LSP : Princípio da Substituição de Liskov

L Se uma classe pai pode fazer algo, uma classe filho também deve ser capaz de fazê-lo.

L Não é necessário escolher corretamente quem é a classe pai e quem é a classe filha, para que não haja conflitos (como em: Um quadrado é uma subclasse de um retângulo? Não, devem ser tratadas como classes separadas.)

• Quarto Princípio SOL**I**D : ISP : Princípio da Segregação de Interface.

L É melhor ter muitas pequenas interfaces do que poucas interfaces gran-des demais.

L Ex.: Ao invés de um mamífero que não deve ser obrigado implementar em uma grande classe chamada "mamífero" que contém o comportamento "comer", é melhor que ele seja integrado em uma pequena interface chamada: "mamíferos que não comem".

Parte 2

- Quinta Princípio SOLID: DIP: Princípio da inversão de dependência:
  - O objetivo da inversão de dependência é acoplar-se a algo abstrato e não concreto.
    - Isso permite que o comportamento seja definido em tempo de execução e facilita a reutilização de código.
    - Inversão de dependência: Princípio que inverte as dependências, garantindo que módulos de alto nível não dependam de módulos de baixo nível, mas de abstrações.
    - Injeção de dependência: Processo de aplicar a inversão de dependências, passando as dependências externamente em vez de instanciá-las (criá-las) diretamente.
    - Injeção de construtor: Injeta dependências através do construtor da classe, garantindo que a classe só funcione com as dependências fornecidas.
    - Injeção de Parâmetro: injeta dependências como parâmetros de métodos, geralmente nos setters, permitindo maior flexibilidade.