SENAI Serviço Nacional de Aprendizagem Industrial **SENAI FATESG** Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. **Arquitetura e Projeto de Software** Docente: Alisson Rodrigues. Aluno: Querley Junio Rodrigues Ferreira.	**Data:** 18/09/2025 **Turma:** 1 **Tipo:** ATIVIDADE AVALIATIVA **Nota:** **Visto Professor:**

Atividade 2 - Princípios SOLID em prática.

Múltipla Escolha:

Questão 1:
b) Uma classe deve ter apenas um motivo para mudar.

Questão 2:
b) Princípio Aberto-Fechado (OCP)

Questão 3:
b) A subclasse deve substituir métodos existentes, mas sem alterar o comportamento esperado.

Questão 4:
b) Criar interfaces específicas para que as classes implementem apenas os métodos que realmente utilizam.

Questão 5:
c) Tanto classes de alto nível quanto de baixo nível devem depender de abstrações.

Questão 6:
b) SRP - Princípio da Responsabilidade Única

Questão 7:
b) SRP - Princípio da Responsabilidade Única

Questão 8:
b) Princípio Aberto-Fechado

Questão 9:
c) LSP - Princípio da Substituição de Liskov

Questão 10:
a) As classes clientes são forçadas a depender de métodos que não utilizam.

Questão 11:
d) A classe Urna Eleitoral apresenta uma quebra do SRP, uma vez que possui responsabilidades que deveriam ser de componentes distintos do software.

Dissertativas:

Questão 12:

São classes com responsabilidades demais, podem ser identificadas olhando se seus métodos e atributos tem pouca relação entre si, se a classe parece confusa, etc.

Questão 13:

Separando as responsabilidades entre classes menores, cada uma com sua função específica.

Questão 14:

Sim, diminuindo o acoplamento, também diminuímos dependências desnecessárias e facilitamos a manutenção, teste, etc.

Questão 15:

Porque quando uma classe muda, é necessário verificar outras, pois estas podem quebrar também.

Questão 16:

Use interfaces simples e abstrações, separe as responsabilidades entre classes menores, diminuindo dependências diretas e o acoplamento.

Questão 17:

O OCP ajuda porque permite adicionar funcionalidades sem alterar o código existente, ele mantém classes abertas para extensão mas fechadas para modificação.

Questão 18:

O OCP abre para extensão e o DIP desacopla classes usando abstrações, eles deixam o sistema mais flexível e modular.

Questão 19:

Composição e Delegação, criando uma instância de uma classe dentro de outra, evitamos heranças desnecessárias.

Questão 20:

Subclasses devem substituir suas classes base sem quebrar o comportamento esperado, se a classe pai cumpre uma função com um método, a classe filha também deve cumprir a mesma função.