# Princípios em Projeto de Software

## Atividades de Aprendizado e Avaliação

**Aluno:_André Luis Quiosi_____RA: 2369958**

**Data: 31/03/2023**

## Use esta cor no seu texto

1. Considerando o texto no link “Inversão de Controle & Injeção de Dependência”,

    a) A Inversão de controle pode ser entendida como a mudança do conhecimento que uma classe tem em relação à outra.

    b) Na primeira versão da classe VendaDeProduto, o problema é o .acoplamento que essa classe tem em relação à classe Log.

    c) Abrir o código fonte da classe VendaDeProduto para mudar o nome do arquivo de log? Comente (3 a 5 linhas)

    R: .Isso é caudado devido ao acoplamento forte que existe entre as duas classes, uma classe conhece e constrói a outra, com isso ela fere o principio de responsabilidade única.

    d) O que a classe VendaDeProduto sabe sobre a classe Log? Comente (2 a 3 linhas)

    R: .A classe VendaDeProduto sabe criar e sabe também que a classe Log precisa de um nome de um arquivo para funcionar.

    e) A .Inversão de controle se dá pela mudança na estrutura do código, de modo que as .classes passam a ser .injetadas . Assim, a classe VendaDeProduto não mais necessita de conhecimento sobre a instanciação da classe Log.

    f) No padrão “.....” as dependências são injetadas via construtor.

    g) A .injeção de dependência torna possível e simples a escrita e execução de .TDD.

    h) A inserção de uma .interface definindo os serviços da classe Log reduziria ainda mais o .acoplamento.

2. Considerando o conteúdo do vídeo “SOLID fica FÁCIL com Essas Ilustrações”

    a) Porque o ROBO MULTIFUNCIONAL quebra o princípio “S” do SOLID? Comente (2 a 3 linhas)

    R: .porque ele está realizando várias tarefas, sendo que deveriam existir um robô para cada responsabilidade.

b) Com unidades independentes e isoladas você consegue

   i) .reaproveitar código.

   ii) .refatoração.

   iii) .testes automatizados.

   iv) Menos .bugs., e mesmo que gere bugs você consegue .isolar. e .concertar o problema.;

c) Em um software com alto .acoplamento., basta um componente no lugar errado para manchar todo o sistema com .algum mau comportamento

d) O nome da função ou componente deve expressar .tudo o que ela está fazendo.

e) O princípio Open/Closed prescreve que deve ser possível adicionar novas funcionalidades sem .precisar alterar a classe base.

f) No princípio Open/Closed, a classe deve estar aberta para .estender novas funcionalidades. mas fechada para .alterações..

g) Uma forma de garantir a extensão sem quebrar o princípio Open/Closed se dá pelo conceito de .design patern..

h) Respeitar o Princípio de Liskov força fazer .abstrações no nível certo. e ser .mais consistente.

i) O exemplo do “pinguim” demonstra a ..... do Princípio da Substituição de Liskov. A abstração “Ave” não está .correta., pois nem toda ave .pode voar..

j) O Princípio da .Substituição de Liskov. promove a especificação de interfaces ..... e mais .....

k) O Princípio da Injeção de Dependências defende que uma classe/módulo não deve .. de outra classe/módulo, mas sim dos .. que este último oferece.

l) No contexto do *Dependency Injection Principle* a classe depende dos serviços .declarados. em uma interface, ou seja, ela não possui ….. com a classe que faz a implementação dos serviços, …... sua existência.

m) Os princípios SOLID foram especificados em 1996 por .Robert C. Martin.