# Atividade Polimorphysm_e_pure_fabrication

1.Pure Fabrication: Qual foi a classe criada na atividade que é um exemplo do

padrão Pure Fabrication? Por que ela é considerada uma "invenção pura" e qual

tipo de responsabilidade ela assumiu?

-ServicoNotificacao, ela é considerada uma invenção pura pois não estar no programa real, ela assumiu a responsabilidade de enviar mensagens.

2. Alta Coesão: Como o padrão Pure Fabrication (a classe que você identificou na

Questão 1) ajudou a manter a Alta Coesão da classe de domínio Pedido?

-Sim, pois se o envio de mensagens fosse responsabilidade da classe pedido, ela se tornaria complexa, tornando-a de baixa coesão.

3. Polymorphism: Na atividade, o Polimorfismo foi implementado usando qual

recurso do Java? E qual é o método comum que todas as classes de notificação

(Email e SMS) devem implementar?

-É a interface(notificação) implements para implementar, classes de Email e SMS possui método comum de enviar().

4. Flexibilidade do Polimorfismo: Se o cliente decidisse adicionar uma nova

forma de notificação (por exemplo, NotificacaoPush), quantas classes

existentes precisaríamos modificar? E qual seria o processo para integrar essa

nova funcionalidade?

-Nenhuma classe será alterada, criar a nova classe, implementar com a interface e método comum, no código Main criar uma instancia de notificação push e inserir no serviço notificação.

5. Acoplamento (Conceito): O que é o "acoplamento" no contexto de

programação orientada a objetos? Como o uso da interface Notificacao no

ServicoNotificacao (Pure Fabrication) contribui para o Baixo

Acoplamento?

-Classes independentes entre si facilitando alterações, o serviconotificação não estar implementado nas classes completas, ele estar para interface, serviconotificação chama apenas o método enviar(), facilitando o baixo acoplamento.

6. Variedade de Comportamentos: Explique a diferença no comportamento do método enviar() entre a classe NotificacaoEmail e a classe NotificacaoSms. Como o polimorfismo permite essa variação sem mudar a classe que as utiliza (ServicoNotificacao)?

-O modo de envio do Email é diferente de um sms, por exemplo Email aceita mais caracteres, pois o seriviçonotificação ele apenas chama o método da interface não se importando se Email ou SMS.

7. Information Expert vs. Pure Fabrication: Se tivéssemos seguido o padrão Information Expert estritamente, qual classe teria sido a candidata a enviar a notificação? Por que essa escolha seria ruim, justificando o uso do Pure Fabrication?

-Classe pedido possuindo toda a informação de Email e envio, isso causaria um problema de alto acoplamento e baixo coesão contendo muitas funções.

8. Vantagem do ServicoNotificacao: Em um cenário de testes (unidade), qual das classes é mais fácil de testar isoladamente: o Pedido com a lógica de envio de email dentro dele, ou o ServicoNotificacao separado, como fizemos? Por quê?

-Seria o ServiçoNotificação, ele teria alta coesão e baixo acoplamento, pois estaria inserido no conceito de polimorfismo e Pure frabrication. O ServiçoNotificação é uma purê frabrication que orquesta a notificação e ServiçoNotificação não estar acoplado aos tipos de mensagens, mas a interface de notificação, possível graças ao polimorfismo.