Pure Fabrication

1️⃣ O que é?

- Padrão GRASP que define uma classe artificial, criada apenas para melhorar a coesão, reduzir o acoplamento ou aumentar a reutilização do código.
- Não representam conceitos do domínio.
- No mundo real, não existe um "Controller", "UI" ou “Repository” como entidade do negócio (são pure fabrication).

💡 **Ou seja, é uma classe que não representa um conceito real do domínio, mas que facilita a implementação.**

2️⃣ Por que é importante?

✅ **Evita baixa coesão** – Se um objeto do domínio for sobrecarregado com demasiadas responsabilidades.

✅ **Reduz o acoplamento** – Permite separar responsabilidades que não deveriam estar no mesmo objeto.

✅ **Aumenta a reutilização** – Classes "fabricadas" podem ser reutilizadas em diferentes partes do sistema.

✅ **Separa preocupações técnicas** – Exemplo: guardar dados no banco de dados **não deve ser responsabilidade de uma entidade do domínio**.

3️⃣ Quando se aplica?

📌 Quando aplicar **Information Expert** resultaria numa classe do domínio com muitas responsabilidades.

📌 Quando se quer **evitar forte dependência** entre classes do domínio e código técnico (ex.: banco de dados, APIs externas).

📌 Quando se precisa de **uma classe auxiliar** para lidar com lógica complexa que não se encaixa numa única entidade.

4️⃣ Como funciona?

Exemplo:

💡 **Problema:**

Se aplicarmos **Information Expert**, a classe **"Department"** ficaria responsável por guardar os seus próprios dados.

Isso causaria **baixa coesão e alto acoplamento**, pois **Department** estaria a misturar regras de negócio com persistência de dados.

🔷 **Solução com Pure Fabrication:** Criamos uma classe **"DepartmentRepository"** para **gerenciar a persistência dos objetos Department**, separando as responsabilidades.