VANTAGENS E DESVANTAGENS GRASP

1. Creator: Quem cria instâncias de A?

B – Promove low coupling, fazendo com que as instâncias de uma classe sejam responsáveis pela criação de objetos de que precisam fazer referência

M - Creation pode exigir uma complexidade significativa.

2. Information Expert: Como atribuir responsabilidades aos objects?

B - Facilitates information encapsulation

B - Promove low coupling

M - Pode fazer com que uma classe se torne excessivamente complexa

3. Low Coupling: Como reduzir o impacto de mudanças e encorajar ao reuse?

B - **Compreensibilidade**: As classes são mais fáceis de entender de forma isolada

B - **Manutenção**: As classes não são afetadas por alterações noutros componentes

B - **Reutilização:** mais fácil de manter classes

M - Um maior acoplamento para classes estáveis não é um grande problema

4. High Coesion: Como manter as classes focadas e manageable?

B - Compreensibilidade, Manutenção

B - Complementa Low Coupling

M - Sometimes desirable to create less cohesive server objects

5. Controller: Quem deve ser responsável pelos UI events?

B - Increased potential for reuse

B - Controller just forwards

M - Reason about the states of the use case

6. **Polymorphism: Como lidar com o comportamento com base no tipo sem if/esle ou switch?**

B - Mais fácil e confiável do que usar selection logic

B - Mais fácil adicionar comportamentos adicionais mais tarde

M - Aumenta número de classes no design

M - Pode tornar o código menos fácil de seguir

7. **Pure Fabrication: What object should have a responsibility when no class of the problem domain may take it without violating High Cohesion and Low Coupling?**

B - A alta coesão é suportada porque as responsabilidades são "factored" numa classe que se concentra apenas num conjunto muito específico de tarefas relacionadas.

B - O potencial de reutilização pode ser aumentado devido à presença de fine grained Pure Fabrication classes.

8. **Indirection: Como evitar Direct Coupling?**

B - Low coupling

B - Promove reusability

9. **Protected Variations: How to design objects, subsystems, and systems so that variations or instabilities in the elements do not have na undesirable impact on other elements?**

B - Mantém o coupling entre classes baixo e torna o design mais robusto

B - Adiciona uma pequena quantidade de sobrecarga na forma de chamadas de método indireto