Design de Software

Padrões GRASP (General Responsibility Assignment Software Patterns)

- Responsabilidades
    - Relacionada às obrigações dos objetos em termos de comportamento
    - Tipos básicos
        - Saber: saber sobre os seus dados, sobre os objetos relacionados, sobre coisas que ela pode calcular
        - Fazer: iniciar ações entre outros objetos, controlar e coordenar atividades em outros objetos
    - Responsabilidades são atribuídas aos objetos do sistema durante o Projeto OO
    - Tradução de responsabilidades para classes e métodos é influenciada pela granularidade da responsabilidade
    - Métodos são implementados para cumprir responsabilidades
    - Exemplo
        - A classe Venda pode definir um ou mais métodos específicos para cumprir a responsabilidade de imprimir.
- Padrões de Software
    - Os padrões de projeto de software ou padrões de desenho de software, são também muito conhecido pelo termo original em inglês: Design Patterns
    - Descrevem soluções para problemas recorrentes no desenvolvimento de sistemas de software orientado a objetos
    - Um padrão de projeto estabelece um nome e define o problema, a solução, quando se aplica esta solução e suas consequências.
    - Padrões capturam experiência existente e comprovada em desenvolvimento de software, ajudando a promover boa prática de projeto.
- Padrões GRASP
    - Um padrão de projeto é uma estrutura recorrente no projeto de software orientado a objetos. Pelo fato de ser recorrente, vale a pena que seja documentada e estudada.
    - Um padrão de projeto nomeia, abstrai e identifica os aspectos chave de uma estrutura de projeto comum para torná-la útil para a criação de um projeto orientado a objetos reutilizáveis.
    - Os principais atributos de uma boa descrição de um padrão de projeto são:
        - Nome
        - Problema
        - Solução
        - Consequências
    - Definição de padrão: um padrão expressa uma solução reutilizável descrita através de três partes: um contexto, um problema e uma solução
        - Contexto: estende o problema a ser solucionado, apresentando situações de ocorrência desses problemas.
        - Problema: determinado por um sistema de forças, onde estas forças estabelecem os aspectos do problema que devem ser considerados.
        - Solução: mostrar como resolver o problema recorrente e como balancear as forças associadas a ele.

- Em geral os padrões de projeto podem ser classificados em três diferentes tipos:
  - Padrões de criação: abstraem o processo de criação de objetos a partir da instanciação de classes.
  - Padrões estruturais: tratam da forma como classes e objetos estão organizados para a formação de estruturas maiores.
  - Padrões comportamentais: preocupam-se com algoritmos e a atribuição de responsabilidade entre objetos.
- Os padrões GRASP (General Responsibility Assignment Software Patterns) codificam ideias e heurísticas existentes para atribuir responsabilidades a objetos, auxiliando a elaborar os diagramas de interação
- Padrão Especialista (Expert)
  - Problema: Qual é o princípio mais básico pelo qual as responsabilidades são atribuídas em um projeto Orientado a Objetos?
  - Solução: Atribuir a responsabilidade ao especialista da Informação – a classe que tem a informação necessária para resolver a responsabilidade.
  - Exemplo: Quem deveria ser responsável por calcular o total-geral de uma venda? A classe Venda possui a informação para isso.
  - Encapsulamento é mantido, leva a fraco acoplamento entre objetos e sistemas mais robustos e fáceis de manter, leva a alta coesão, já que os objetos fazem tudo que é relacionado à sua própria informação
- Padrão Criador (Creator)
  - Problema: Quem deve ser responsável por criar uma nova instância de alguma classe?
  - Solução: Atribua à classe B a responsabilidade de criar uma instância da classe A se:
    1. B agrega objetos de A
    2. B contém A
    3. B armazena instâncias de A
    4. B usa objetos de A
    5. B possui informação necessária a criação de A (B é um especialista para criar A)
  - Exemplo: Quem deveria ser responsável por criar a instância da classe Item de Venda? A Classe Venda agrega muitos objetos da classe Itens de Venda.
- Padrão Baixo Acoplamento (Low Coupling)
  - Problema: Como minimizar dependências e maximizar o reuso?
  - Solução: Atribuir a responsabilidade de modo que o acoplamento (dependência entre classes) seja baixo
  - O acoplamento é uma medida de quão fortemente uma classe está conectada, possui conhecimento ou depende de outra classe
- Padrão Alta Coesão (High Cohesion)
  - Problema: Como manter a complexidade (das classes) sob controle?
  - Solução: Atribuir uma responsabilidade de modo que a coesão seja alta
  - A coesão é uma medida do quanto as responsabilidades de uma classe estão relacionadas.

        - Exemplo: Quem deve ser responsável por criar um Pagamento e associá-lo à Venda? Pelo padrão Criador, seria POST. Mas se POST for o responsável pela maioria das operações do sistema, ele vai ficar cada vez mais sobrecarregado e sem coesão. Uma solução melhor seria a classe Venda criar Pagamento. Neste caso a responsabilidade de criar pagamento fica melhor distribuída pois Venda tem uma associação forte com pagamento.
    - Padrão Controlador (Controller)
        - Problema: Quem deveria ser responsável por lidar com um evento do sistema?
        - Solução: Atribua responsabilidades para lidar com mensagens de eventos do sistema a uma classe que:
            1. represente o sistema como um todo
            2. represente a organização
            3. represente algo ativo no mundo real envolvido na tarefa (por exemplo o papel de uma pessoa)
            4. represente um controlador artificial dos eventos de sistema de um caso de uso
        - Exemplo: Exemplo: Quem deveria ser o controlador para eventos do sistema tais como entrarItem e Vendafim? A melhor solução é usar uma classe exclusivamente para fazer este papel. Isto causa menor acoplamento e a classe PontodeVenda tem apenas o papel de controladora.