**MEETINGS**

**Discussões:**

* SOLID
* COMMAND QUERY SEPARATION

**SOLID**

**O que é?**

Princípios para uma boa modelagem de classes.

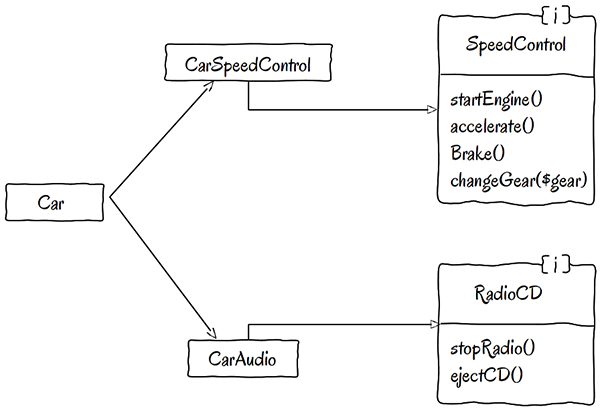
**Vantagens**

* Coesão no código;
* Legibilidade;
* Fácil Manutenção.

**Quando usar?**

Quando a classe está muito grande, ou seja, com métodos extensos significa que a sua classe não está adequada nos princípios de SOLID.

A Legibilidade não está muito explicita, causa confusão e entendimento ao ler o código.

* **SOLID** **Principle**
  + **SRP**: Classes com responsabilidade única
    - Exemple:
  + **OCP**: Classes abertas/fechadas, abertas para expansão e fechadas para mudanças. Você conseguirá implementar mais funcionalidades sem mudar a estrutura das classes.
    - Exemple
  + **LSP:** Classes filhas nunca deveriam infringir as definições de tipo da classe pai. uma subclasse deve sobrescrever os métodos da classe pai, de tal maneira que não quebre a funcionalidade do ponto de vista do cliente.
    - Exemple: Quando houver a necessidade de implementar uma nova funcionalidade não será afetado o sistema como um todo.
    - O Princípio de herança deve estar sempre presente.
  + **ISP:** Expõe as funcionalidades em formas de interfaces. Que separam as funcionalidades
    - Exemple:
    - 
  + **DIP**: As Classes não devem ser totalmente acopladas, e nem totalmente desacoplada. As classes devem ser separadas por responsabilidades e herdadas de abstrações como base. As Interfaces devem herdar das Abstrações, para não serem totalmente acopladas.
    - Exemple:

**Quando não usar?**

Quando

**Como fazer a equipe aderir ao todo?**

Com Softwares Legados, creio que seja mais difícil de implementar, pois demanda refatoração no código e reestruturação na arquitetura da solução.

Com projetos pequenos e que estão no início do desenvolvimento, seja mais fácil implementar os princípios SOLID.

Uma boa modelagem de código nunca será acertada na primeira tentativa, creio que é necessário implementar princípio a princípio.

Pull Request e Code Review: Orientará os desenvolvedores, para melhor desempenho.

**COMMAND QUERY SEPARATION**

**O que é?**

A ideia fundamental é que devemos dividir os métodos de um objeto em duas categorias nitidamente separadas:

Consultas: retornar um resultado e não alterar o estado observável do sistema (são livres de efeitos colaterais).

Comandos: Alterar o estado de um sistema, mas não retornam um valor.

**Quando usar?**

Separação da consulta comando é particularmente bem adequado para um projeto por **contrato metodologia (DBC)** -> *(desenvolvimento de software que prescreve que os desenvolvedores devem definir métodos formais, especificações de interface precisas e verificáveis dos componentes de desenvolvimento de software, que acarreta na definição de Tipo Abstrato de Dados com pre-condições, pos-condições e constantes. Estas especificações são definidas como um "contrato", de acordo com os próprios conceitos de condições e obrigações dos contratos de negócio).*

**Quando não usar?**

Torna mais difícil de implementar software re-entrantes e multi-threaded corretamente. Isso geralmente ocorre quando um padrão de não-thread-safe é usado para implementar a separação-consulta de comando.

**Como fazer a equipe aderir ao todo?**