UNIVERSIDADE DE BRASÍLIA

Faculdade do Gama

Técnicas de Programação em Plataformas Emergentes

Trabalho Prático

Etapa 3

Caio Brandão Santos - 170007413

Brasília, DF

2025

Questão 1

1. Simplicidade

a. **Definição**: Um código simples é fácil de entender e seguir. Ele evita complexidade desnecessária.
b. **Maus-Cheiros**: Códigos complexo e difíceis de entender gerar maus-cheiros. Exemplos incluem métodos longos (Long Method) e classes grandes (Large Class), onde uma classe ou um método se torna muito extenso, tornando-se difícil de trabalhar.

2. Elegância

a. **Definição**: Um código elegante é limpo, claro e utiliza as melhores práticas de programação.
b. **Maus-Cheiros**: "Duplicate Code" e "Long Parameter List" são exemplos de maus-cheiros que violam a elegância de um código. Esses maus-cheiros tornam o código mais repetitivo e poluído, tornando o código menos claro e legível.

3. Modularidade

a. **Definição**: Um código modular é dividido em partes distintas e independentes que podem ser desenvolvidas e modificadas separadamente.
b. **Maus-Cheiros**: "Divergent Change" e "Shotgun Surgery" são maus-cheiros que afetam a modularidade. "Divergent Change" ocorrem quando precisamos alterar muitos métodos independentes para conseguir realizar uma mudança em uma classe. Já o “Shotgun Surgery” é o contrário e ocorre quando precisamos alterar muitas classes para poder realizar uma pequena modificação.

4. Boas Interfaces

a. **Definição**: Interfaces bem definidas facilitam a comunicação entre diferentes partes de um sistema.
b. **Maus-Cheiros**: "Inappropriate Intimacy" e "Message Chains" são maus-cheiros que indicam problemas com as interfaces. Intimidade inapropriada ocorre quando uma classe utiliza muito de atributos e/ou métodos de outra classe. Já cadeias de

mensagens ocorrem quando um objeto acessa um método de um segundo objeto, que acessa um método de um terceiro objeto, e assim por diante. Imagine uma série de chamadas como $a->b()->c()->d(). Este tipo de cadeia de mensagens é perigoso pois uma alteração no relacionamento entre dois objetos pode afetar a cadeia inteira.

5. Extensibilidade

a. **Definição**: Um código extensível pode ser facilmente ampliado com novas funcionalidades sem grande refatoração.
b. **Mau-Cheiro**: "Parallel Inheritance Hierarchies" pode ser considerado um mau-cheiros que impacta a extensibilidade. Ocorre quando cada vez que você adiciona uma subclasse a uma hierarquia, é necessário adicionar uma subclasse correspondente à outra hierarquia, tornando o código difícil de manter e estender.

6. Evitar Duplicação

a. **Definição**: Código duplicado é um desperdício e propenso a erros; evita-se a repetição de código.
b. **Mau-Cheiro**: "Duplicate Code" é um claro mau-cheiro que contradiz este princípio. Quando o mesmo código aparece em vários lugares, fica difícil mantê-lo.

7. Portabilidade

a. **Definição**: Um código portátil pode ser facilmente adaptado para diferentes ambientes e plataformas.
b. **Mau-Cheiro**: "Incomplete Library Class" é um mau-cheiro que pode dificultar portabilidade. Quando uma biblioteca ou classe de biblioteca está incompleta, torna-se difícil usá-la em diferentes ambientes e plataformas, pois funcionalidades necessárias podem estar ausentes.

8. Código Idiomático e Bem Documentado

a. **Definição**: Um código idiomático segue as convenções e melhores práticas do idioma em que foi escrito e é bem documentado para facilitar a compreensão.
b. **Maus-Cheiros**: Dentre os vários maus-cheiros que podem ferir o princípio de código bem documentado, um deles é chamado de “Magic Number”. Ele se refere ao uso de números “hard-coded” no código sem nenhum contexto. Como por exemplo” if (price > 35)” qual o significado do número 35? Alguém diferente do autor dessa linha, provavelmente ficará confuso quanto ao significado.

Questão 2

a) **Classe Grande (Large Class):** A classe IRPF possui muitos atributos e métodos, o que a torna difícil de entender e manter.
    i. **Princípios violados:** Fácil de entender, fácil de manter, menos provável de ter bugs.
    ii. **Operações de refatoração:** Extrair classes para dividir a responsabilidade em classes menores e mais coesas.

b) **Métodos Longos (Long Method):** Alguns métodos, como cadastrarDeducaoIntegral, são longos e realizam múltiplas tarefas.
    i. **Princípios violados:** Fácil de entender, fácil de manter, menos provável de ter bugs.
    ii. **Operações de refatoração:** Extrair método para dividir métodos longos em métodos menores e mais específicos.

c) **Nomes de Métodos e Variáveis Pouco Descritivos:** Alguns nomes de métodos e variáveis não são suficientemente descritivos, o que dificulta a compreensão do código.
    i. **Princípio violados:** Fácil de entender.
    ii. **Operações de refatoração:** Renomear métodos e variáveis para nomes mais descritivos.

Referências Bibliográficas

1. Martin Fowler. Refactoring: Improving the design of Existing Code. Addison-Wesley Professional, 1999.
2. Code Smells. Disponível em: https://refactoring.guru/pt-br/refactoring/smells. Acesso em 12 fev. 2025