**Atividade 1 – Questionário sobre Clean Code**

### 1. O que você entende pelo conceito de "código limpo"? Como ele se aplica à sua experiência de desenvolvimento?

**Um código limpo** é um código que é simples de entender, fácil de modificar/manutenir e livre de complexidade desnecessária. É um conceito criado e defendido pelo autor Robert Cecil Martin. No livro, ele cria e defende a idéia de boas práticas de programação e mostrando que é possível organizar de forma que outros desenvolvedores possam compreendê-lo facilmente a partir dessa convenção.

Na experiência de desenvolvimento, ele se aplica por uma escrita cuidadosa, evitando redundâncias e mantendo o código sempre claro para futuras manutenções e colaboração em equipe.

### 2. Como a simplicidade de design influencia a qualidade do código?

A **simplicidade de design** reduz a probabilidade de erros e torna o código mais fácil de manter e evoluir com adição de novas features. Um design simples evita complexidades desnecessárias e ajuda a focar no que é essencial, garantindo que o código seja mais claro e compreensível, além de facilitar o entendimento e a integração de novos desenvolvedores no projeto.

### 3. Quais práticas você adota para manter nomes de variáveis e funções significativos e claros?

Adotar **nomes descritivos** que demonstrem claramente o propósito de variáveis e funções é essencial. Com isso, nos referimos a evitar abreviações desnecessárias (lprTxt()), usar verbos para funções que executam ações (limparTexto()) e substantivos para variáveis que armazenam dados (**var** texto **=** “”). A escolha de bons nomes elimina a necessidade de comentários explicativos e melhora a legibilidade do código.

### 4. De que forma a refatoração contínua ajuda a manter um código limpo e sustentável?

A **refatoração consiste em revisar o código, identificar erros e pontos de melhoria e reescrevê-los de uma maneira melhor, fazê-la de maneira contínua** mantém o código organizado, limpo e atualizado, sem alterar sua funcionalidade. Ela corrige inconsistências, melhora a estrutura das linhas e reduz a complexidade, tornando o código mais sustentável a longo prazo. Assim, o código não se deteriora/obsolesce, mesmo com novas adições e mudanças.

### 5. Como o "Princípio da Responsabilidade Única" impacta a modularidade e legibilidade do código?

O **Princípio da Responsabilidade Única (SRP)** sugere que cada classe ou função deve ter uma única responsabilidade ou motivo para mudar. Isso melhora a **modularidade,** porque o código é dividido em pequenos componentes com funções bem definidas, uma vez que cada parte do código tem um propósito claro e isolado. Ou seja, evita funções extensas e que serviriam a mais de um propósito desnecessariamente.

### 6. Em que situações a regra de "escreva o mínimo de comentários possível" não se aplica?

Em situações onde o código precisa lidar com lógica complexa, exceções ou contexto específico do negócio que não é óbvio apenas pela leitura do código. Comentários também são úteis para explicar decisões de design ou escolhas técnicas não triviais.

### 7. Como você equilibra código limpo e pressão por entregas rápidas?

O equilíbrio do código limpo com prazos apertados pauta-se em **priorizar boas práticas básicas** como nomes significativos, modularidade e simplicidade. Refatorações mais profundas podem ser sacrificadas em decorrência desses prazos curtos.

### 8. Que técnicas do livro ajudaram você a escrever testes mais eficazes?

O livro fortalece muito a importância de testes automatizados claros e específicos. A aplicação do PSU nas funções teste ajuda a escrever testes de mais fácil compreensão e mantêm alta a cobertura de código. Além disso, técnicas como TDD (onde os testes são escritos antes do código) garantem a “testabilidade” de um código.

### 9. Qual a importância de manter funções pequenas e focadas em uma única tarefa?

Funções **pequenas e focadas** são mais fáceis de entender, testar e manter. Esse tipo de organização melhora a legibilidade do código e facilita futuras modificações.

### 10. Como "Clean Code" mudou sua percepção sobre boas práticas de programação?

No início, eu via a programação como uma área mais próxima das exatas, onde bastava fazer o código funcionar. Com o tempo, percebi que há muito mais envolvido. O "Clean Code" me ensinou que a verdadeira qualidade do código não está apenas em sua funcionalidade, mas em sua manutenção, clareza e simplicidade. Essa abordagem me mostrou a importância de escrever código que seja compreensível e fácil de modificar, promovendo uma melhoria contínua e evitando problemas futuros.