**Atividade 2 – Questionário sobre Refatoração**

### 1. Como a refatoração influencia o processo de manutenção e evolução de um código legado?

A refatoração melhora o código, tornando-o mais legível e fácil de trabalhar. Isso facilita a manutenção e evolução do sistema permitindo que novos devs consigam compreender o código rapidamente. Vale ressaltar que no processo da refatoração, há a etapa de remoção das duplicações e simplificação da estrutura, isso reduz a chance de bugs quando o código é alterado.

### 2. Em que momento do ciclo de desenvolvimento você acha mais apropriado realizar a refatoração?

A refatoração deve ser feita de forma contínua, especialmente quando você está mexendo no código para adicionar algo novo ou corrigir bugs. Isso inclui quando após incrementar uma feature.

### 3. Quais sinais de "código com cheiro" você já encontrou em sua experiência, e como lidou com eles?

Alguns exemplos de "código com cheiro" são métodos ou classes muito grandes, código repetido e nomes de variáveis/métodos confusos. Para melhorar, normalmente divide-se o código em partes menores, dá-se nomes melhores e removem-se duplicações.

### 4. Qual o impacto da refatoração na performance do sistema?

A refatoração pode melhorar a performance quando o código fica mais simples e organizado. Ao remover partes desnecessárias, como código repetido ou funções que não são tão eficientes, o sistema pode rodar mais rápido e usar menos recursos. Mesmo que o foco da refatoração seja deixar o código mais fácil de entender e manter, às vezes isso também acaba tornando o programa mais ágil e performático.

### 5. Como a refatoração pode contribuir para melhorar a testabilidade do código?

Ao refatorar, o código fica mais modular e simples, o que facilita a criação de testes. Métodos e classes menores são mais fáceis de testar (PRU de código limpo), e usando injeção de dependência, por exemplo, você consegue testar as partes sem depender de outras.

### 6. De que forma a prática de refatoração afeta a produtividade da equipe?

No começo pode parecer que refatorar toma tempo (e realmente toma), mas a longo prazo, deixa o trabalho da equipe mais rápido e eficiente. Um código mais limpo é mais fácil de entender e mexer, o que reduz o tempo gasto com bugs e problemas.

### 7. Quais estratégias do livro você utiliza para simplificar métodos e classes complexas?

Algumas estratégias incluem:

* **Extração de métodos**: dividir métodos longos em métodos menores com uma única responsabilidade.
* **Renomeação**: usar nomes descritivos para métodos e variáveis, aumentando a clareza do código.
* **Remoção de código duplicado**: consolidar código repetido em métodos reutilizáveis.

### 8. Como você lida com o desafio de manter funcionalidades enquanto refatora um código?

Testes automatizados ajudam a garantir que o código continua funcionando como antes. Deve-se refatorar aos poucos, validando cada mudança, o que reduz a chance de quebrar algo.

### 9. O que você considera ser o maior benefício da refatoração para a legibilidade do código?

O maior benefício da refatoração é que ela pega um código bagunçado e transforma em algo mais fácil de entender. Isso ajuda a equipe a trabalhar melhor junto e permite que novos desenvolvedores consigam se adaptar mais rápido ao projeto, deixando o trabalho em geral mais eficiente.

### 10. De que maneira a refatoração influencia o design orientado a objetos e a modularidade?

Refatorar melhora o design do código, deixando as classes e métodos mais organizados e com funções específicas. Isso facilita o desenvolvimento e manutenção do sistema, já que as partes ficam mais independentes e fáceis de testar e modificar.