Separação de Responsabilidade

Lógica de Programação I

Definindo o tamanho dos métodos

Quando falamos em dividir o código em funções ou métodos (nome mais apropriado dentro da linguagem Java), é comum vir a dúvida: *quando eu sei que devo separar o código num método próprio?* É sobre isso que vamos discutir nesse tópico.

Definindo o tamanho dos métodos

Quando um código está difícil de ser compreendido ou alterado, é um bom sinal que ele está longo demais, assumindo muitas responsabilidades.

Definindo o tamanho dos métodos

Funções Pequenas

> "*A primeira regra para funções é que elas devem ser pequenas. A segunda é que devem ser ainda menores.*"
>
> (Robert C. Martin)

Não existe um estudo que comprove a eficiência de métodos pequenos, que eles sejam mais rápidos ou melhores, ou que gerem menos erros. O fato é que ao restringir ao máximo sua responsabilidade, o código fica mais coeso e, por consequência, mais legível.

Definindo o tamanho dos métodos

Faça apenas uma coisa

Sempre que você escrever um código, imagine-se explicando-o para um colega. No livro "**O programador pragmático**" (*David Thomas, Andrew Hunt*), os autores descrevem a cena de um desenvolvedor explicando o comportamento do código para um pato de borracha em sua mesa. Esse momento de "olhar com outros olhos", ou de um ponto de vista diferente, é fundamental em todo processo criativo. E na programação não é diferente.

Definindo o tamanho dos métodos

Faça apenas uma coisa

1. Pegue seu telefone
2. Pegue sua carteira
3. Pegue sua chave
4. Abra a porta de casa e saia
5. Se dirija até a rua
6. Ande pela calçada
7. Verifique se é seguro atravessar a rua
8. Atravesse a rua
9. Continue na calçada até a venda
10. Localize o freezer
11. Localize o sorvete de creme
12. Vá ao caixa
13. Pague pelo sorvete ...

Definindo o tamanho dos métodos

Faça apenas uma coisa

1. Pegue telefone, carteira e chave
2. Vá à venda mais próxima
3. Compre o sorvete de creme

Exercícios

Exercício 1: Qual é o principal benefício da separação de responsabilidades em um código?

a) Torna o código mais curto e conciso.
b) Facilita a reutilização e manutenção do código.
c) Aumenta a velocidade de execução do código.
d) Reduz a quantidade de comentários necessários.

Exercício 2: Qual das seguintes opções é um exemplo de separação de responsabilidades em um sistema de gerenciamento de vendas?

a) A classe **Venda** gerencia tanto a venda quanto a geração de relatórios.

b) A classe **Relatorio** gerencia tanto a exibição de dados quanto a validação de entrada.

c) A classe **Venda** gerencia apenas a venda, enquanto a classe **Relatorio** gerencia apenas a geração de relatórios.

d) A classe **Venda** gerencia a venda e a geração de relatórios ao mesmo tempo.

Exercício 3: Quando dizemos que uma classe em um sistema deve ter uma única responsabilidade, o que estamos buscando principalmente?

a) Reduzir a quantidade de métodos na classe.
b) Aumentar o acoplamento entre as classes.
c) Permitir que a classe tenha várias funcionalidades diferentes.
d) Melhorar a coesão e facilitar a manutenção.

Exercício 4: Qual das seguintes opções é uma consequência positiva da separação de responsabilidades em um projeto de software?

a) Aumento da complexidade do código.
b) Dificuldade na identificação de bugs.
c) Facilitação da colaboração entre desenvolvedores.
d) Redução da necessidade de documentação.

Exercício 5: Converta graus Celsius para Fahrenheit

Elabore um programa em Java que peça ao usuário para inserir uma temperatura em graus Celsius. Desenvolva um método que converta essa temperatura para Fahrenheit e mostre o resultado.

Fórmula:

```
(Celsius * 9 / 5) + 32
```

Exercício 6: Conversor de Moeda

Desenvolva um programa em Java que converta um valor em dólar para real. Utilize um método que receba o valor em dólar como entrada e retorne o valor equivalente em real.

Considere uma taxa de câmbio fixa.

Taxa de Câmbio: 4.86

Obrigada