06

Code Smells

Transcrição

Agora, veremos a segunda refatoração: *incorporar método*. Para ver essa técnica, também precisaremos do Visual Studio. Dentro do projeto `refatoracao` e dentro da pasta `Aula01`, teremos uma outra que receberá o nome `R02.InlineMethod` usada para incorporar métodos. Dentro dela, temos a pasta `depois`, que contém o arquivo `Motoboy.cs`.

Temos uma classe `Motoboy` que obtém a avaliação de algum motoboy. Essa classe chamará o método `TemMaisDeCincoEntregasNoturnas()` e retornará um *boolean* indicando se a variável `qtdeEntregasNoturnas` é ou não maior que 5.

```
class Motoboy
{
    private int qtdeEntregasNoturnas;

    int GetAvaliacao()
    {
        return (TemMaisDeCincoEntregasNoturnas()) ? 2 : 1;
    }

    bool TemMaisDeCincoEntregasNoturnas()
    {
        return qtdeEntregasNoturnas > 5;
    }
}
```

O nome do método faz referência ao que expressão faz, ou seja, **o corpo do método é tão evidente quanto o seu nome**. Neste caso, aplicaremos a técnica "Extrair método" para fazer a operação inversa. A ação é chamada de **Incorporar método**.

Para colocar em prática essa técnica, pegaremos a expressão de retorno, copiaremos e substituiremos onde o método é chamado. Depois eliminaremos o método `TemMaisDeCincoEntregasNoturnas()`.

```
class Motoboy
{
    private int qtdeEntregasNoturnas;

    int GetAvaliacao()
    {
        return (qtdeEntregasNoturnas > 5) ? 2 : 1;
    }
}
```

Agora temos um método tão claro quanto o anterior.

A primeira técnica que vimos foi o *Extract Method,* que consiste em pegar um trecho de código que pode ser agrupado. Podemos refatorar em situações como:

- Código duplicado;
- Método muito grande;
- Comentários e cadeias de mensagens.

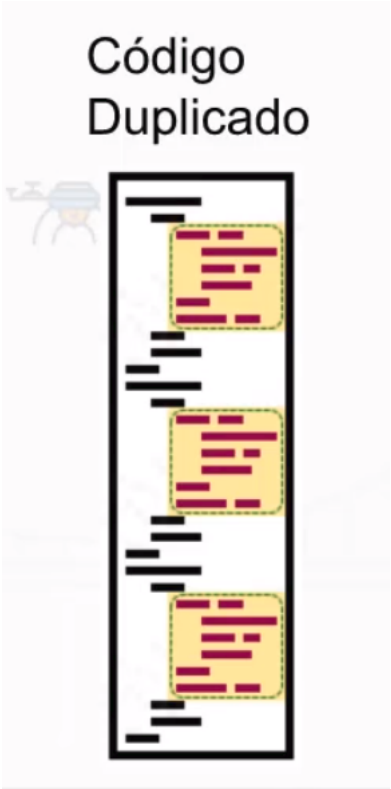
Não extrairemos o método quando a própria expressão retornada pelo ele, for auto-instrutiva.

Vimos também o *Inline Method*, trata-se da operação inversa do "Extract Method". Usaremos essa técnica quando o método for tão óbvio quanto o seu nome, a ponto de se tornar desnecessário. Nós não refatoramos quando o corpo do método não é auto-explicativo, quando é óbvio, e quando o nome do método não explica o que a expressão faz.

Agora, veremos uma técnica de composição chamada de **Code Smells**. Essa técnica consiste em revisar o código para descobrir se algo está errado. Adiante, veremos algumas situações em que algo está "cheirando mal".

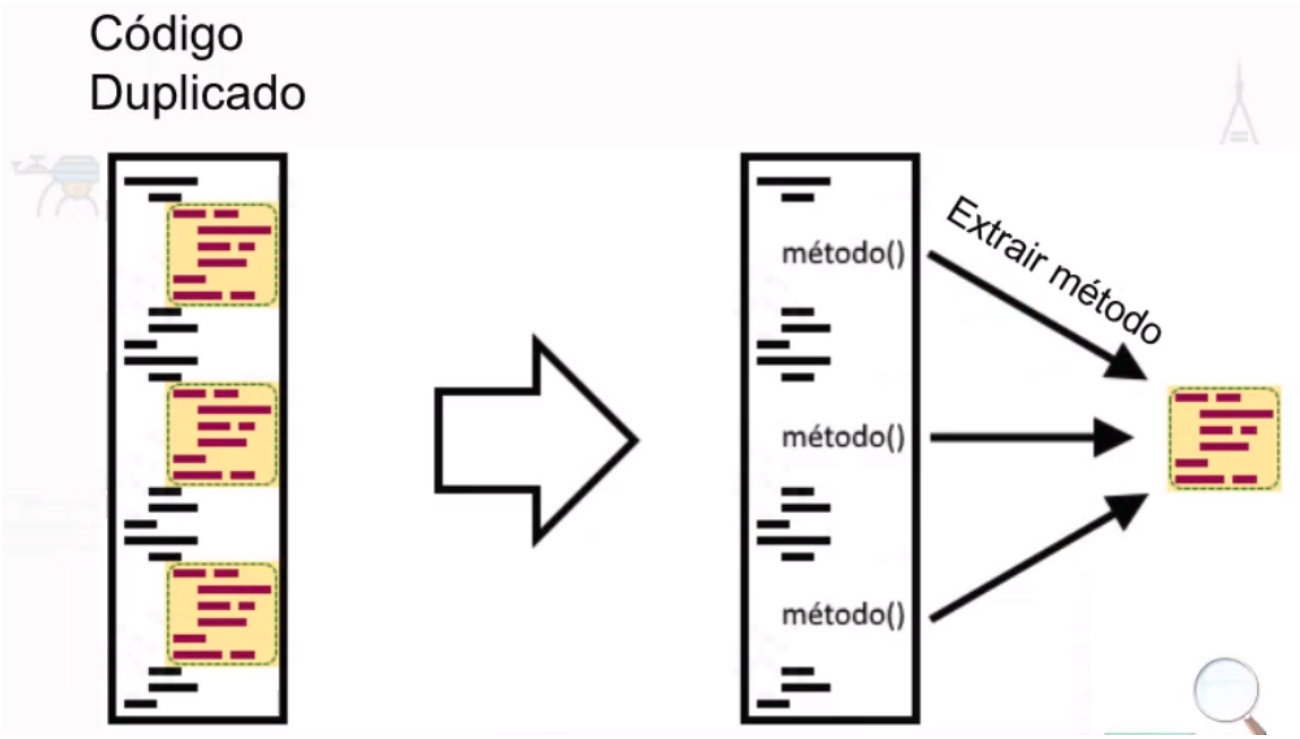
Código Duplicado

Com o código duplicado, violamos uma das boas práticas de programação, conhecida como **DRY**: *Don't Repeat Yourself* ("Não se Repita", traduzido para português).



O problema da duplicação é que se temos um código replicado que está com bug, provavelmente os outros trechos também o terão. E para corrigir esse bug, é preciso fazer a manutenção do código em vários lugares onde esse trecho foi copiado.

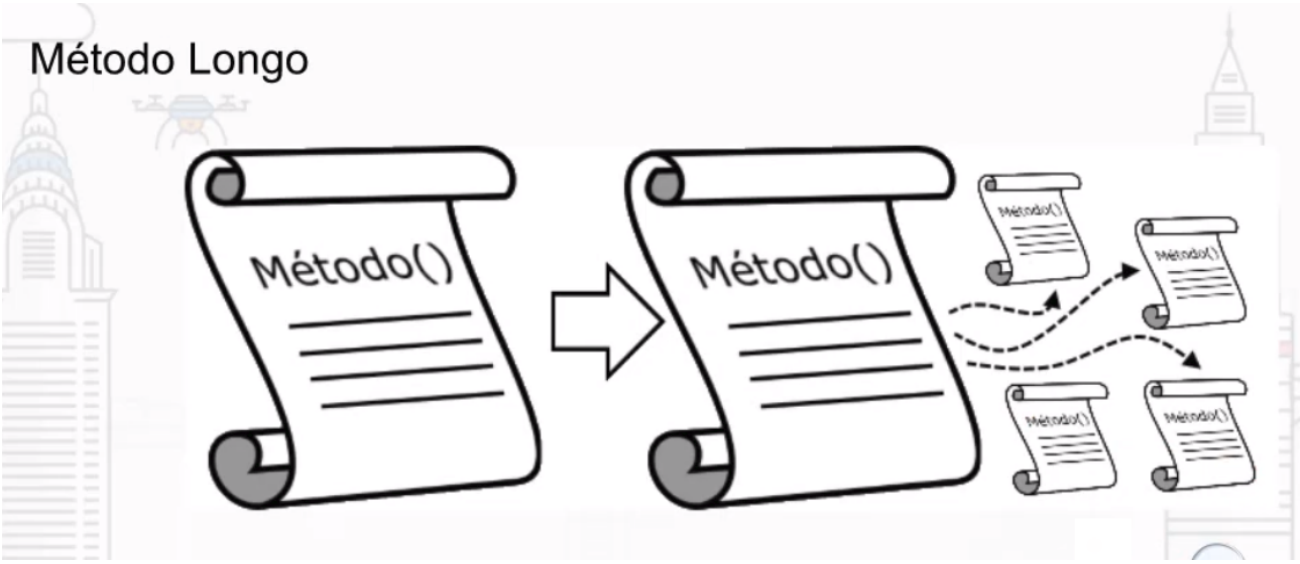
A solução para o código duplicado é **extrair método*.



Então, os lugares que tinham esse trecho copiado, são substituídos pela a chamada desse novo método.

Método Longo

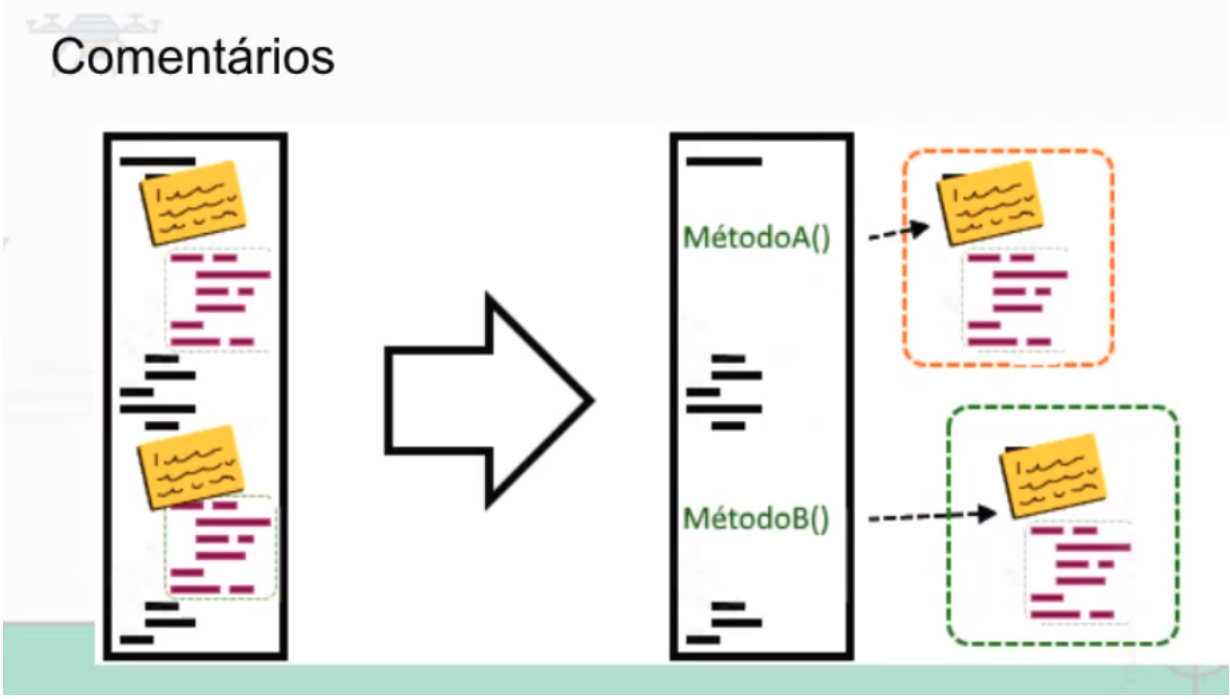
Quando temos um método muito longo, geralmente violamos o "Princípio de Responsabilidade Única" ou *Simple Responsibility Principle*, no qual é dito que o método deve realizar **somente uma única tarefa**. Outro problema é que os **métodos longos** são difíceis de ler, enquanto os **métodos curtos** são mais fáceis de comparar e de entender.



Comentários

A necessidade de comentários são considerados um *Code Smell* porque **podem não explicar o código da forma correta**, estar mais desatualizados do que o código, enganar o desenvolvedor e também podem inibir a refatoração.

Resolvemos os comentários, extraindo métodos.



De acordo com a imagem, podemos ver a eliminação de dois comentários por meio da extração dos métodos `A()` e `B()`.