04

Extrair Método 1

Transcrição

Começaremos a ver as **Técnicas de Refatoração**. Dentro do grupo de *técnicas de composição*, em primeiro temos `Extrair Método`. Para visualizar o seu funcionamento, é necessário abrir o Visual Studio.

No projeto `caelum-stella-csharp`, abriremos o arquivo `Moeda.cs` que está dentro da pasta `Inwords`. Veremos logo no início a classe `Moeda`, que entre outras coisas, fará a impressão do valor monetário por extenso.

Por exemplo, o valor de `R$ 500,32` terá como saída `quinhentos reais e trinta e dois centavos`. Para isso, usaremos o método `Extenso()` dentro da classe `Moeda`.

Para entendermos melhor o que esse método faz, é necessário visualizar o código completo. A desvantagem de códigos grandes é que sempre precisamos rolar o código para cima ou para baixo para que possamos visualizá-lo por inteiro. Esse código está dividido em partes:

```
public override string Extenso()
{
    StringBuilder builder = new StringBuilder();

    //Montar os inteiros

    //Montar os centavos
}
```

O comentário já nos ajuda a entender o objetivo de cada parte desse código. Entretanto, existe um problema, como por exemplo, quando fazemos uma melhoria no código, e o comentário não é atualizado. Isso pode enganar um futuro desenvolvedor que for lê-lo, pois o comentário estaria defasado.

E como podemos eliminar esses problemas? Podemos extrair o método! Então, selecionaremos o trecho do método a ser extraído, que se encontra abaixo do comentário `//Montar os inteiros`:

```
    //Montar os inteiros
    if (numeroOrigem < 0)
    {
        throw new ArgumentOutOfRangeException();
    }
    else if (numeroOrigem >= 1.0 || numeroOrigem == 0)
    {
        BuildNumeroMoeda(numeroOrigem, builder);
        BuildPreposicaoMilhoes(numeroOrigem, builder);
        BuildPalavraMoeda(numeroOrigem,builder );
    }
```

Após ter selecionado esse trecho do código, aparecerá uma lâmpada à esquerda que nos dará opções de refatoração. Entre as opções, encontraremos o "**Extract Method**". Ao clicar nessa opção, automaticamente esse trecho será substituído pela chamada do método. Nesse momento, teremos a oportunidade de renomear esse método como `MontarInteiros()`. Após pressionarmos o "Enter", teremos o trecho trocado dessa forma:

```
public override string Extenso()
{
    StringBuilder builder = new StringBuilder();

    //Montar os inteiros
    MontarInteiros(builder);
}
```

O método `MontarInteiros()` está localizado na própria classe `Moeda` um pouco mais abaixo. Ele recebe um `StringBuilder builder` como parâmetro. É interessante porque foi identificado que o trecho de código extraído, precisa da variável local `builder` . E com isso, foi criado automaticamente a chamada desse método já passando como parâmetro a variável que é uma dependência do trecho.

No comentário, temos somente a informação `//Montar inteiros` . E por causa da refatoração, esse comentário ficou obsoleto. Por isso, podemos deletá-lo. Adiante, temos o segundo trecho de código, aquele que monta os centavos!

Faremos o mesmo processo de refatoração feito com o método anterior, selecionando o método, clicando na lâmpada para extrair. Ou podemos também, fazer esse processo utilizando o atalho "Ctrl + ." e será exibida a opção de extração. Chamaremos esse método de `MontarCentavos()` .

Assim como o primeiro, o segundo método criado está situado logo mais abaixo. O método `Extenso()` ficou da seguinte forma:

```
public override string Extenso()
{
    StringBuilder builder = new StringBuilder();
    MontarInteiros(builder);
    MontarCentavos(builder);
    return builder.ToString();
}
```

Temos a declaração do `StringBuilder` , as chamadas dos métodos `MontarInteiros()` e `MontarCentavos()` , e por último é retornado o `builder` com o resultado do texto por extenso.