04

Incorporar Variável Temporária

Transcrição

Assim como a técnica "extrair método" possui a sua forma inversa, conhecida como "incorporar método", a técnica "extrair variável" tem a inversa: **incorporar variável temporária**.

No Visual Studio, abriremos o projeto `refatoracao`, em que encontraremos a classe `CalculadoraDePrecos.cs` dentro de "Aula02 > R04.InlineTemp > depois".

A `CalculadoraDePrecos.cs` tem um método que verifica se o pedido tem desconto ou não. O método `TemDesconto()` retorna um **valor verdadeiro** somente se o valor dos produtos do pedido, for superior a `R$ 1000,00`. Entretanto, dentro desse método, temos a declaração de uma variável temporária chamada `valorProdutos`, que recebe `pedido.ValorProdutos()`:

```
class CalculadoraDePrecos
{
    bool TemDesconto(Pedido pedido)
    {
        decimal valorProdutos = pedido.ValorProdutos();
        return (valorProdutos > 1000);
    }
}
```

Assim a variável temporária é usada para verificar se ela ultrapassou os `R$ 1000,00`. Mas qual é o problema com a declaração da variável? Será que ela está armazenando uma expressão complexa? Não, a expressão armazenada é tão óbvia quanto o próprio nome da variável.

Como vimos anteriormente, este é o caso em que o corpo do método será explicado pelo próprio nome. Podemos simplificar o nosso código incorporando a expressão de uma variável no próprio código que utiliza essa variável.

Quando posicionamos o cursor em cima da variável `valorProdutos`, a lâmpada à esquerda aparecerá e nos dará acesso às opções de refatoração. Temos a opção *Inline temporary variable*, ou seja, "incorporar a variável temporária".

Automaticamente, após selecionarmos essa opção, o Visual Studio extrairá a expressão armazenada na variável e a substituirá pela expressão seguinte:

```
class CalculadoraDePrecos
{
    bool TemDesconto(Pedido pedido)
    {
        return (valorProdutos > 1000);
    }
}
```

Se o valor dos produtos do pedido for maior que `1000`, será retornado **verdadeiro**. Essa expressão é tão clara quanto àquela que utilizada com a variável, e com isso, temos um ganho na facilidade da leitura.

No final desse código, temos a classe `Pedido`, onde existe o método `TemDesconto()`. Este possui uma expressão bem mais complexa:

```
class Pedido
{
    public bool TemDesconto()
    {
        //aqui NÃO É um bom exemplo para inline method!
        bool clienteHaMaisDe5Anos = (DateTime.Today.Subtract(clienteDesde).TotalDays / 365) >= !
        bool compraEspecial = valorProdutos > 1000;
        return clienteHaMaisDe5Anos && compraEspecial;
    }
}
```

Ocorre uma diferença entre datas na variável `clienteHaMaisDe5Anos` para verificar a quantidade de anos, e também, por meio da variável `compraEspecial` é verificado se o valor dos produtos ultrapassou `1000` reais. No fim, o método retorna uma condição para verificar se ele é um cliente há mais de cinco anos e também se o valor da compra ultrapassou mil reais.

Será que podemos incorporar a variável temporária aqui? Vamos tentar utilizar a mesma técnica de refatoração posicionando o cursor na variável `clienteHaMaisDe5Anos`, e depois clicando na opção *Inline temporary variable* dentro da lâmpada.

Agora, faremos o mesmo processo com a variável `compraEspecial`, incorporando a variável temporária. Este é o resultado:

```
class Pedido
{
    public bool TemDesconto()
    {
        //aqui NÃO É um bom exemplo para inline method!
        return (DateTime.Today.Subtract(clienteDesde).TotalDays / 365) >= 5 && valorProdutos > 1
    }
}
```

Agora temos um código muito mais difícil de entender do que o código anterior. Nesse caso, não devemos refatorar pois estamos prejudicando a leitura do código.

É muito importante saber quando usar e quando não usar cada uma das refatorações.

Vamos desfazer o processo utilizando o atalho "Ctrl + Z".