01

Ocultar Delegado

Transcrição

> Começando deste ponto? Você pode fazer o [DOWNLOAD (https://github.com/alura-cursos/csharp-refatorando-codigo/archive/3fe8cedfcb0ca475eac6ad5c47f65f9be091227c.zip)](https://github.com/alura-cursos/csharp-refatorando-codigo/archive/3fe8cedfcb0ca475eac6ad5c47f65f9be091227c.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

As próximas técnicas que veremos são **Ocultar Delegado** e **Remover Intermediário**.

Delegação é o ato de transferir uma responsabilidade para outro objeto. Temos um exemplo da técnica **Ocultar Delegado** a seguir:

```
var resultado = this.GetObjetoA().GetObjetoB().GetObjetoC().GetObjetoD();
```

Neste caso queremos obter o objeto `d` usando o método `GetObjetoD()`. Mas para isso é necessário acessar o objeto `a`, para acessarmos o objeto `b` e assim poderemos acessar o objeto `c`. Desta forma, teremos acesso ao objeto `d`. Este exemplo se enquadra em um *Code Smells* chamado **Cadeia de Mensagens**, que viola a "Lei de Demeter". Esta lei estabelece que cada objeto só pode acessar o objeto que está próximo a ele.

Resolveremos isso, acessando no projeto `refatoracao`, o arquivo `Escola.cs`, que está localizado em "Aula07 > R14.HideDelegate > depois". Dentro de `Escola.cs`, encontraremos uma classe `Teste`, que acessará uma sequência de objetos do arquivo.

```
class Escola
{
    public string Nome { get; private set; }
    public Funcionario Diretor { get; private set; }
}

class Departamento
{
    public Escola Escola { get; private set; }
}

class Funcionario
{
    private Departamento Departamento { get; private set; }
}

class Teste
{
    public Teste()
    {
        var maria = new Funcionario();
        var diretorDaMaria = maria.Departamento.Escola.Diretor;
    }
}
```

Se queremos obter o diretor de um funcionário, que no caso é a Maria, passaremos a instância `maria`, depois, a instância `Departamento`. Agora acessaremos `Escola` e, então, chegaremos ao `Diretor`. Nossa tarefa é ocultar a delegação, transferindo-a para dentro da classe `Funcionario`.

A primeira tarefa é trocar a visibilidade de `Departamento` para `private`, e então retirar o `private set`:

```
class Funcionario
{
    private Departamento Departamento { get; }
}
```

E então, colocaremos um método para acessar o `Diretor` nessa classe:

```
class Funcionario
{
    private Departamento Departamento { get; }
    public Funcionario GetDiretor()
    {
        return this.Departamento.Escola.Diretor;
    }
}
```

Desta forma, retornaremos o funcionário `Diretor` em `GetDiretor()`. Trocaremos o código na classe `Teste` para ocultar essa delegação:

```
class Teste
{
    public Teste()
    {
        var maria = new Funcionario();
        var diretorDaMaria = maria.GetDiretor();
    }
}
```