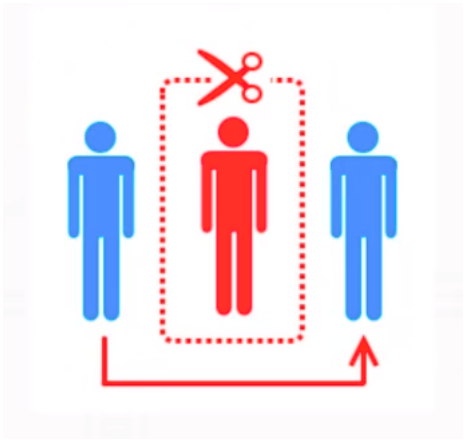
04

Remover Intermediário

Transcrição

A próxima refatoração desse capítulo é a ***Remove Middle Man*** ou *Remover o Intermediário*.

Imagine que temos duas classes representadas em azul e - por uma questão de arquitetura - para chamar o método de uma classe para outra, passaremos por um *intermediário*.



Como vemos, a classe intermediária não agrega valor a essa comunicação. E o intermediário acaba sendo redundante e ele não faz praticamente nada. Podemos refatorar eliminando-o nesse caso.

Abriremos o projeto `refatoracao`, que contém o arquivo `Escola.cs` na pasta "Aula07 > R15.RemoveMiddleMan > depois". Temos as seguintes classes nesse 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; }
    public Funcionario Diretor{...}

    private decimal salario;
    public decimal Salario{...}
}

class Empregado
{
    private readonly Funcionario funcionario;
    public Empregado(Funcionario funcionario)
    {
```

```
            this.funcionario = funcionario;
        }

        public Funcionario Diretor
        {
            get    { return funcionario.Diretor; }
        }

        public decimal Salario
        {
            get    { return funcionario.Salario; }
        }
    }
```

Para executar o código, temos a classe `Teste`, ainda no mesmo arquivo:

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

Observe que dentro da classe `Teste`, temos uma instância chamada `maria`, do tipo `Empregado()`. Ela recebe uma nova instância de `Funcionario()`. No final, obtemos na variável `diretorDaMaria`, o diretor da Maria chamando uma propriedade `Diretor` a partir da instância `maria`.

O que a classe `Empregado` faz? Ela recebe uma instância de `Funcionario` que é armazenada como um *campo local* do empregado. Ela também tem uma propriedade `Diretor` que obtém, a partir de um `funcionario`, o `Diretor` desse funcionário. Mais para frente, temos uma outra propriedade chamada de `Salario`.

Conseguimos obter o `Salario` a partir do `funcionario.Salario`, o que significa que a classe `Empregado` está agindo como intermediária entre a classe `Teste` e a classe `Funcionario`.

Por isso, eliminaremos a classe `Empregado` e, depois, acessaremos diretamente a classe `Funcionario`.

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

Com isso, conseguimos eliminar o intermediário que será irrelevante em nossa aplicação.