Refatoramento de Código

Exemplo extraído do livro *Refactoring: Improving the Design of Existing Code* (Fowler, 1999)

Parte 3: De quem é a responsabilidade de calcular o valor de um locação?

I. Delegando responsabilidades para a classe Locação

No exercício anterior, movemos parte do código do método **extrato()** da classe **Cliente** para um método a parte denominado **valorDeUmaLocacao()**. O código é exibido a seguir:

```
private float valorDeUmaLocacao(Locacao umaLocacao) {
            float valorDaLocacao = 0.0;

            // determina valores para cada linha
            switch(umaLocacao.getCarro().getCodigoDoPreco()) {
            case Automovel.BASICO: // R$ 90.00 por dia
                  valorDaLocacao += umaLocacao.getDiasAlugado() * 90.0;
                  break;

            case Automovel.FAMILIA: // R$ 130.00 por dia
                  valorDaLocacao += umaLocacao.getDiasAlugado() * 130.0;
                  break;

            case Automovel.LUXO: // R$ 200.00 por dia
                  valorDaLocacao += umaLocacao.getDiasAlugado() * 200.0;

                  // Adiciona um desconto de 10% se alugar o carro por mais
                  // de 4 dias
                  if(umaLocacao.getDiasAlugado() > 4) {
                        valorDaLocacao *= 0.9;
                  }
                  break;
            } //switch

          return valorDaLocacao;
      }     // fim do método valorDeUmaLocacao()
}
```

Ao analisarmos o código com mais atenção, podemos perceber que o método **valorDeUmaLocacao()** usa informação de um objeto da classe **Locacao** e não usa nada da classe **Cliente**. De acordo com a regra **Expert Information**, *"devemos colocar as responsabilidades com os dados"*. Por isso, desconfiamos que o referido método não está definido na classe correta, pois faz mais sentido estar presente na classe **Locacao**.

II. **Tarefa** (refactoring2-1)

Mova o método **`valorDeUmaLocacao()`** para a classe **`Locacao`**. Essa mudança de código causou impacto na codificação de outras classes?

Na prática, houve uma **delegação** do cálculo do **`valorDeUmaLocacao`** de **`Cliente`** para a classe **`Locacao`**. No método **`extrato()`** da classe **`Cliente`**, podemos remover a chamada do método privado **`valorDeUmaLocacao(cada)`** e deixar que a classe concreta **`Locacao`** assuma a responsabilidade de calcular o preço da locação.

III. **Remoção de variáveis temporárias desnecessárias: Parte I**

Após realizar as modificações necessárias, podemos identificar algumas variáveis no método **`extrato()`** que são desnecessárias. Por exemplo, não há necessidade de manter a variável **`valorCorrente`**, haja vista que seu resultado pode ser usado diretamente na linha de instrução que acumula os valores dos aluguéis. Além disso, ter menos variáveis significa que mitigamos situações para dar errado por não receber o valor correto, não ser inicializada, etc.

IV. **Tarefa** (refactoring2-2)

Refatore o método **`extrato()`** da classe **`Cliente`**, de modo que as variáveis sem serventia sejam eliminadas. Depois das mudanças, compile e rode o código novamente. A saída deve ser a mesma e a classe **`Locadora`** não deve ser afetada pelas mudanças. Reflita sobre como foi o impacto desta mudança no seu código.

V. **Extração do cálculo de Pontos de Alugador Frequente**

A questão neste momento, ao examinar como ficou o método **`extrato()`** é refletir sobre **de quem é a responsabilidade de calcular os Pontos de Alugador Frequente (PAF)**. Pergunta básica:

"Quem detém (qual classe) a informação para cálculo do PAF?"

VI. **Tarefa** (refactoring2-3)

De acordo com a pergunta acima, mova o bloco de código de cálculo do PAF para a classe que deve ter essa responsabilidade (que detém a informação para cálculo). Depois das mudanças, compile e rode o código novamente. A saída deve ser a mesma e a classe **`Locadora`** não deve ser afetada pelas mudanças. Reflita sobre como foi o impacto desta mudança no seu código, no geral.

VII. **Remoção de variáveis temporárias desnecessárias: Parte II**

Ótimo! O código está ficando muito interessante. Vamos olhar para o método **`extrato()`** e refletir sobre a existência de variáveis temporárias. Será que não podemos remover a variável **`valorTotal`**? Embora elas possam ser úteis, deixam evidências de indicativo de "mau cheiro" (*bad smell*).

Se pararmos para analisar bem, a variável **`valorTotal`** é utilizada para acumular dados e calcular o valor total do extrato enquanto o **loop** está funcionando. Este loop está servido para 3 coisas:

a) Montar a string referente ao extrato (variável **`resultado`**);
b) Calcular o valor total dos aluguéis (variável **`valorTotal`**);
c) Calcular os PAF (variável **`pontosDeAlugadorFrequente`**);

Entretanto, o cálculo do valor total faz sentido ser deslocado para outro lugar se considerarmos essa linha de raciocínio: posso querer saber o valor total em outro método como, por exemplo, **`extratoHTML()`**, e terei portanto que repetir o cálculo do preço total neste lugar se não houver um método que faça esse cálculo.

Sendo assim, faz sentido criar um método **`getValorTotal()`**. Se ste método faz algo que podemos resumir em uma frase curta, como "calcule o valor total", então podemos criar o método. Foi o mesmo que aconteceu com a variável temporária **`pontosDeAlugadorFrequente`**, criar um método **`getPontosDeAlugadorFrequente()`.**

Embora esses dois passos devam ser feitos separadamente com testes a cada passo, vamos logo ver o resultado dos dois passos, criando também o método **`getPontosTotaisDeAlugadorFrequente()`** e chamando-o dentro do método **`extrato()`**.

VIII. **Tarefa** (refactoring2-4)

Mova o bloco de código que calcula o valor total no **`extrato()`** para o método **`getValorTotal()`** na classe **`Cliente.`** Também, defina na classe cliente o método **`getPontosTotaisDeAlugadorFrequente()`**. Se houver modificações em outras classes ou métodos, faça os ajustes. Depois das mudanças, compile e rode o código novamente. A saída deve ser a mesma e a classe **`Locadora`** não deve ser afetada pelas mudanças. A modificação foi simples? Reflita sobre como foi o impacto desta mudança no seu código.