07

# Faça o que eu fiz na aula

Quebrar Variável Temporária

Na pasta quebrar_variavel_temporária, temos uma função que imprime a área e o perímetro de um retângulo, e essa função utiliza dois parâmetros de tipo inteiro, a altura e a largura do retângulo.

Para fazermos o cálculo do perímetro, somamos a altura com a largura e multiplicamos o resultado por dois, armazenamos isso em uma variável com o nome de `temp`, que significa temporário e depois imprimimos na tela do navegador.

Depois fazemos o cálculo da área, que é a altura vezes a largura e armazenamos nessa mesma variável temporária o resultado.

Porém, temos um problema, como vimos no vídeo estamos usando uma variável temporária, e o valor dentro dela está sendo sobrescrito quando fazemos o cálculo da área.

Para refatorarmos isso, vamos utilizar a refatoração Quebrar Variável Temporária, e com isso, vamos trocar o nome da segunda atribuição da variável `$temp` para `$area`.

```
$area = $altura * $largura;
```

E para termos um nome mais descritivo, ao invés de `$temp` vamos mudar o nome da primeira variável atribuída para `$perimetro`

```
$perimetro = 2 * ($altura + $largura);
```

Agora se voltarmos ao navegador e recarregarmos a página, vamos ter o mesmo comportamento que tínhamos no início, porém com uma maior flexibilidade e legibilidade no nosso código por causa da refatoração.

Remover Atribuição à parâmetros

Na pasta remover_atribuicao_a_parametros temos um arquivo index.php que dentro dele é definida uma função que dá desconto para um cliente e aumenta esse desconto se algumas condições forem satisfeitas. Porém, vimos na aula que temos um problema se precisarmos imprimir o valor inicial do nosso desconto, pois estamos sobrescrevendo a variável e perdemos o valor inicial.

Podemos fazer a refatoração Remover Atribuição à Parâmetros, onde no começo da função, declaramos uma variável para receber o parâmetro e fazemos a atribuição nela, deixando o parâmetro intacto.

```
$resultado = $descontoInicial;
```

A partir daí, utilizamos esta variável para fazer as atribuições:

```
    if ($resultado > 200) {
        return;
    }
```

```
    if ($ehPremium === true) {
        $resultado = $resultado * 1.1;
    }

    if ($quantidade > 50) {
        $resultado = $resultado * 1.2;
    }

    if ($anosCliente > 3) {
        $resultado = $resultado * 1.1;
    }
```

E para imprimirmos na tela, agora conseguímos imprimir o desconto inicial e o desconto final:

```
echo <<<EOF
--------------------
Inicial: $descontoInicial
Desconto: $resultado
--------------------
EOF;
}
```

Assim, nosso código ficou mais simples, e não perdemos o valor do desconto inicial que enviamos para a função.

Substituir Algoritmo

Na pasta substituir_algoritmo, temos um arquivo index.php com a lista de strings contendo as aulas disponíveis em um curso, e logo embaixo temos uma declaração de função que é responsável por fazer a busca de uma string em uma lista de aulas.

Nesta função, são executados três laços foreach percorrendo cada elemento da lista de aulas que recebemos, no primeiro laço, para cada elemento é executada uma expressão regular para verificar se a string recebida como parâmetro está presente na lista de aulas.

Se ela estiver presente, elas são adicionadas ao array de palavras encontradas.

No entanto, temos um problema que esta palavra será procurada exatamente como foi digitada, e temos outros laços e expressões regulares que vão percorrer a lista múltiplas vezes, isso tem tanto um problema de performance quanto de usabilidade.

Às vezes, não adianta tentar ficar refatorando um algoritmo ruim, vamos utilizar uma técnica de refatoração chamada Substituir Algoritmo.

Olhando a documentação do PHP, vemos que depois do delimitador de padrão da função `preg_match`, podemos utilizar a letra i para procurar tanto letras maiúsculas quanto minúsculas, vamos modificar a primeira iteração:

```
        if (preg_match("/$pesquisa/i", $aula) !== 0) {
```

Com isso, podemos simplificar e muito o nosso algoritmo, podemos remover as outras duas iterações.

E como não vamos também encontrar duplicados, podemos remover a linha com a chamada de `array_unique`

```
$unicas_encontradas = array_unique($encontradas);
```

E agora temos um código mais legível e performático utilizando uma funcionalidade da linguagem de programação que estamos usando.