08

Faça o que eu fiz na aula

Temos dois arrays, um deles tem os nomes dos correntistas de um banco que são pagantes, e no outro, temos os correntistas não-pagantes do banco.

```
$correntistas = [
    'Giovanni',
    'João',
    'Maria',
    'Luis',
    'Luisa',
    'Rafael',
];

$correntistasNaoPagantes = [
    'Luis',
    'Luisa',
    'Rafael',
];
```

E nos foi solicitado que fizessemos um array que possui apenas os correntistas pagantes, como podemos fazer isso?

No PHP, existe uma função chamada `array_diff` , que gera um novo array com a diferença entre dois arrays.

```
$diferentes = array_diff($correntistas, $correntistasNaoPagantes);
```

Agora temos um array de correntistas e precisamos fazer com que eles sejam relacionados com o seu saldo.

```
$correntistas = [
    'Giovanni',
    'João',
    'Maria',
    'Luis',
    'Luisa',
    'Rafael',
];

$saldos = [
    2500,
    3000,
    4400,
    1000,
    8700,
    9000,
];
```

Podemos utilizar uma função do PHP que chama `array_merge` ! vamos utilizá-la?

```
$relacionados = array_merge($correntistas, $saldos);

var_dump($relacionados);
```

Mas não foi muito bem o que queríamos, vejam que o `array_merge` coloca um array no final do outro, e isso não dá a impressão de que os correntistas estão relacionados com o saldo.

Ao invés do `array_merge` , vamos utilizar o `array_combine` , como a assinatura do método é a mesma, não precisamos alterar os parâmetros.

```
$relacionados = array_combine($correntistas, $saldos);
```

O array_combine utilizou um array como chave do outro, isso é chamado de array associativo, ou também de Map.

Vamos mudar o nome da variável de `$relacionados` para `$correntistas_saldos` , para descrever melhor o que esta variável está representando.

Para acessarmos uma posição de um array associativo, fazemos algo parecido com os arrays, porém, ao invés de um número inteiro, digitamos a string usada como chave entre colchetes:

```
echo $correntistas_saldos[“Giovanni”];
```

Para inserirmos um elemento no array associativo, usamos a mesma sintaxe:

```
$correntistas_saldos[“Marcio”] = 6600;
```

Para utilizarmos um array associativo, usamos da mesma maneira do que um array, a diferença é que utilizamos strings como chave ao invés de números inteiros de maneira ordenada.

Quando precisamos fazer a interpolação de um array associativo, como o array associativo possui uma string entre aspas como sua chave, precisamos utilizar essa sintaxe especial e colocar a variável entre chaves.

```
echo “<p>O saldo do Giovanni é {$saldos[”Giovanni”]}</p>”;
```

Podemos realizar uma checagem antes de acessarmos a posição de um array associativo, para verificarmos se uma chave existe em um array, podemos utilizar a função do PHP chamada `array_key_exists` , que retornará um valor booleano caso seja, ou não, encontrada no array.

```
if (array_key_exists("Joao", $correntistasContas)) {
    echo $correntistasContas["Joao"];
} else {
    echo "Não existe no array";
}
```

Podemos também inicializar um array associativo utilizando uma forma mais curta, assim:

```
$correntistas = [
    “Giovanni” => 2500,
    “João” => 3000,
    “Maria” => 4400,
    “Luis” => 1000,
    “Luisa”, => 8700,
    “Rafael” => 9000,
];
```

Legal! agora precisamos verificar neste array associativo, todas as pessoas que possuem mais que 3000 reais como saldo.

Vamos para a classe `ArrayUtils` e criar o método encontrarPessoasComSaldoMaior:

```
public static function encontrarPessoasComSaldoMaior(int $saldo, array $array): array
{

}
```

Vamos criar um array para armazenar os maiores:

```
public static function encontrarPessoasComSaldoMaior(int $saldo, array $array): array
{
    maiores = [];
}
```

Agora podemos fazer uma iteração sobre esse array associativo, que é parecido com o que fazemos com o array, podemos utilizar o `foreach`, mas com uma sintaxe um pouco diferente:

```
public static function encontrarPessoasComSaldoMaior(int $saldo, array $array): array
{
maiores = [];
            foreach ($array as $chave => $valor) {
            if ($valor > $saldo) {
                $maiores[] = $chave;
            }
        }
return $maiores;
}
```

Dentro desse loop, teremos acesso a chave e ao valor sendo iterado pelo array associativo, fazemos uma comparação do valor e se for maior, guardamos a chave, que será o nome.

E por fim, retornamos o array contendo os nomes.

Podemos verificar se está tudo correto chamando o método:

```
$pessoas = ArrayUtils::encontrarPessoasComSaldoMaior(3000, $correntistasContas);

var_dump($pessoas);
```