03

Simplificando o cálculo com lambda

Antes de adicionarmos uma nova feature na nossa App, vamos aplicar o processo de refatoração na classe `Resumo` para que fique fácil de compreender o que acontece em cada uma das funções dela.

Vamos começar com o processo de refatoração na função `receita()`. Veja que dentro dela temos um código um tanto quanto grande que tem a finalidade de passar por todas as transações, **filtrar** as de receita e **somá-las** utilizando estruturas como `for` e `if`.

Em outras palavras, vamos utilizar funções prontas que fazem isso.

Filtrando a as transações com o Filter

Primeiro, vamos começar pelo filtro, ou seja, utilize o objeto `transacoes` e chame a função `filter()`. Ela recebe uma função que espera um teste que retorna um `boolean` para poder filtrar as transações.

Para poder enviar essa função, utilize a expressão lambda **apelidando** o item da lista como `transacao`. Em seguida, considere o **teste** em que o tipo de transação seja `Tipo.RECEITA`.

Dica: A expressão lambda tem a seguinte syntax: `{//apelido -> //teste}`.

Somando todas as transações

Após a chamada do `filter()`, precisamos também somar cada uma das transações que foram filtradas.

Para isso, chame também a função `sumByDouble()` enviando a expressão lambda para somar os valores de cada uma das transações.

Lembre-se que nesta chamada é preciso converter a property `valor` da `Transacao` para `Double`.

Depois de finalizar a expressão lambda, retorne a função `sumByDouble()` para a variável `somaDeReceita`, então, para utilizar a soma que foi realizada via expressão lambda, atribua o retorno da função `receita()` a partir de uma instância de `BigDecimal` recebendo como parâmetro do construtor a variável `somaDeReceita`.

Por fim, execute a App e veja se ainda está funcionando como esperado.