12

Para saber mais: funções Lambda

As funções lambda são um recurso muito útil na linguagem de programação Java. Elas podem ajudar a simplificar nosso código e torná-lo mais fácil de ler e compreender. Mas antes de avançarmos, vamos começar pelo início.

O que são funções Lambda?

As funções Lambda - chamadas de funções anônimas - são uma maneira de definir funções que são frequentemente usadas uma única vez, direto no local onde elas serão usadas.

Na programação convencional, normalmente definimos uma função em algum lugar do nosso código e, em seguida, chamamos essa função por seu nome em outro lugar, sempre que precisamos usá-la.

Porém, às vezes precisamos de uma função que seja usada em apenas um lugar no nosso código. Para esses casos, pode ser mais simples e direto definir essa função diretamente no lugar onde ela será usada, sem lhe dar um nome, ou seja, usando uma função Lambda.

A principal vantagem do uso de funções lambda é a simplificação do código e a melhora na legibilidade, já que a função é definida logo onde será usada.

Como usamos funções Lambda no Java?

Em Java, uma função lambda é definida da seguinte maneira:

```
(argumentos) -> { corpo-
```

COPIAR CÓDIGO

Por exemplo, podemos definir uma função lambda que adicione dois números da seguinte maneira:

```
(a, b) -> { return a + b
```

COPIAR CÓDIGO

Aqui, `a` e `b` são os argumentos da nossa função. O corpo da função, que é o código que será executado quando a função for chamada, está entre chaves `{}`. Neste caso, o corpo da função é apenas uma linha que retorna a soma de `a` e `b`.

Em Java, as funções lambda são, geralmente, usadas com interfaces funcionais. Uma interface funcional é uma interface que contém apenas um único método. A função lambda então fornece a implementação desse único método.

Exemplos de uso de funções Lambda

Vamos a um exemplo concreto para entender melhor. Suponhamos que temos uma lista de números e queremos

printar apenas os números pares dessa lista. Sem o uso de funções lambda, poderíamos fazer algo assim:

```
List<Integer> lista = Ar

for(Integer i: lista) {
  if(i % 2 == 0) {
    System.out.println(i
  }
}
```

COPIAR CÓDIGO

Porém, com o uso de funções lambda, podemos simplificar esse código:

```
List<Integer> lista = Ar

lista.stream().filter(i
```

COPIAR CÓDIGO

No código acima, criamos um stream da nossa lista, filtramos esse stream para incluir apenas os números pares (isso é feito pela função lambda `i -> i % 2 == 0`), e finalmente usamos o

método `forEach` para printar cada elemento do stream filtrado.

Agora ficou mais claro como as funções Lambda em Java podem nos ajudar a simplificar nosso código e torná-lo mais legível!