06

Para saber mais - Técnica de implementação de Higher-Order Function

Quando estávamos implementando as Higher-Order Function com expressão lambda, vimos algumas alternativas de implementação, como por exemplo, na função `chama()` da `FormularioTransacaoDialog` :

```
fun chama(tipo: Tipo, delegate: (transacao: Transacao) -> Unit) {
    // restante do código
}
```

Fizemos a seguinte implementação:

```
AdicionaTransacaoDialog(viewGroupDaActivity, this)
      .chama(tipo) { transacaoCriada ->
          adiciona(transacaoCriada)
          lista_transacoes_adiciona_menu.close(true)
      }
```

Repare que enviamos o parâmetro `transacaoCriada` por 2 motivos:

- Para acessar o parâmetro que está sendo recebida.
- Para identificar dar significado ao parâmetro.

Uma outra alternativa é enviando o objeto subentendido `it` :

```
AdicionaTransacaoDialog(viewGroupDaActivity, this)
       .chama(tipo) {
           adiciona(it)
           lista_transacoes_adiciona_menu.close(true)
       }
```

Porém, por mais que tenhamos visto essas alternativas, apenas analisando o código por cima, fica difícil de compreender o que significa, certo?

Em outras palavras, quando utilizávamos a interface `TransacaoDelegate` era bem claro o objetivo da implementação, porém, agora que utilizamos a expressão lambda, precisamos entrar dentro da função `chama()` para compreender que o objetivo da Higher-Order Function é realizar um delegate.

Utilizando o Named parameter para dar mais significado

Considerando esse tipo de cenário, uma outra possibilidade de implementação que visa uma melhor compreensão da Higher-Order Function é utilizando o Named Parameter:

```
AdicionaTransacaoDialog(viewGroupDaActivity, this)
            .chama(tipo, delegate = {
        adiciona(it)
```

```
            lista_transacoes_adiciona_menu.close(true)
        })
```

Veja que dessa forma, é claro que a expressão lambda é destinada ao delegate. Claro, nesse tipo de situação, existem algumas alternativas para melhorar mais ainda a leitura, uma delas é colocando novamente o parâmetro para indicar o que está sendo delegado:

```
AdicionaTransacaoDialog(viewGroupDaActivity, this)
          .chama(tipo, delegate = { transacaoCriada ->
      adiciona(transacaoCriada)
      lista_transacoes_adiciona_menu.close(true)
  })
```

Ou então, para manter o objeto `it`, podemos mudar o nome da Higher-Order Function:

```
fun chama(tipo: Tipo, transadaoDelegate: (transacao: Transacao) -> Unit) {
    // restante do código
}
```

```
AdicionaTransacaoDialog(viewGroupDaActivity, this)
            .chama(tipo, transadaoDelegate = {
        adiciona(it)
        lista_transacoes_adiciona_menu.close(true)
    })
```

Ambas alternativas são bem válidas e já melhoram bastante a leitura e compreensão do código, mas perceba que ao optar por esse tipo de implementação, é necessário implementar a expressão lambda dentro dos parênteses.