01

Conhecendo as Higher Order Function

Transcrição

[00:00] Conseguimos deixar o nosso código bem mais flexível utilizando a herança, aqui no Kotlin, mas vamos dar uma olhada como é que a gente está utilizando, por exemplo, os nossos Dialogs, lá na nossa Activity. Vamos lá, o Ctrl+N aqui, "ListaTransacoesActivity", e vamos entrar aqui para ver o que acontece, primeiro eu vou aumentar o código para vocês, Ctrl+Shift+F12.

[00:18] reparem que a gente tem os nossos membros, as funções, as properties e, aqui, a gente tem essa função "chamaDialogDeAdicao", que é justamente aquela função que fica responsável em fazer a instância da "AdicionaTransacaoDialog", que a gente manda os parâmetros de ViewGroup, manda o contexto e, então, a gente faz a chamada da chama, que, justamente, abre aquele nosso dialog para gente.

[00:39] Nessa chamada aqui, olha como a gente está fazendo atualmente. Reparem que aqui a gente manda o tipo, e aqui a gente faz a implementação da interface "transacaoDelegate", a gente faz uma implementação dessa interface, que só tem apenas uma única função a ser implementada.

[00:52] A gente viu anteriormente, que quando a gente tem esse tipo de comportamento, de uma interface que tem apenas uma única função, a gente pode fazer uma conversão para uma expressão lambda, como foi o caso que a gente fez aqui no "setonClickListener", em outras palavras, vamos tentar converter aqui, essa "TransacaoDelegate" em uma expressão lambda.

[01:12] Vamos chegar aqui, o primeiro passo que a gente vai fazer é tirar esse object aqui, que é a parte do Object Expression, tirou ele aqui, reparem que agora, aqui a gente não precisa mais colocar a função que está sendo implementada, porque a ideia da expressão lambda é implementa-la, por debaixo dos panos. E agora a gente está colocando aqui a implementação da expressão lambda.

[01:31] Percebam, também, que aqui, a gente tem essa referência da transação, que sabe-se lá de onde tá vindo, ou seja, a gente precisa indicar, via parâmetro da expressão lambda, o que é essa transação. A gente vai chegar aqui e vai colocar agora como "transacao", e aqui a gente vai colocar a flechinha da expressão lambda, mas reparem que, ao fazer esse tipo de implementação de expressão lambda, nessa interface que a gente tem, que é a "transaçãoDelegate", não foi possível fazer isso.

[01:55] Só que agora, porque isso acontece? Lembra daquela vez que eu tinha comentado com vocês, que a gente consegue fazer a expressão lambda por que é uma interface que foi feita em Java. Aquela informação não foi falada apenas por falar, isso significa o seguinte, que realmente, aqui dentro do Kotlin, a gente consegue só consegue fazer esse tipo de conversão, de implementação de object expression, para uma expressão Lambda, por meio das interfaces do Java, a gente pode até fazer um teste para isso.

[02:22] É isso que a gente vai fazer agora, para a gente tentar entender os motivos de isso acontecer, vamos começar primeiro com esse teste, o primeiro passo que a gente vai fazer aqui, é justamente criar uma interface, que vai ter exatamente o mesmo comportamento da "TransacaoDelegate", só que agora vai ser feita em Java. A gente vai tentar fazer essa implementação, primeiro via object expression, e depois, via expressão lambda, e a gente vai ver isso acontecendo.

[02:42] Vamos lá, Alt+1 aqui, eu vim em Delegate, Alt+Insert, da mesma maneira como a gente fazia lá no Kotlin, e aqui encher a gente tem a opção de Java Class, que é a opção padrão que o Android Studio nos dá. Reparem que aqui a gente vai colocar a opção como "TransacaoDelegateJava", e aqui no tipo a gente vai colocar agora como uma Interface. Eu vou

dar um "Ok", ele criou para gente. Reparem que ele colocou esse comentário, a gente pode apagar sem nenhum problema, e agora vamos começar com a implementação dessa interface.

[03:12] Qual que vai ser essa implementação? A gente vai fazer mais ou menos uma réplica do item que a gente fez, que é aquela que foi feita em Kotlin, a gente vai colocar um "void" aqui e vai deixar o "Delegate". Dentro desse Delegate a gente vai colocar o tipo de transação que a gente espera, "Transacao transacao", e aqui coloca um ponto e vírgula. Dessa maneira a gente criou exatamente uma interface que tem o mesmo comportamento da nossa Delegate, só que agora ela foi feita em Java.

[03:35] Vamos agora testar, para ver se realmente, aquilo que eu tinha comentado com vocês, sobre a conversão, acontece? Vamos lá. Vamos pegar aqui dentro do nosso código, e vamos criar aqui uma função, só para a gente testar de fato, a gente vai fazer um teste para aquela interface do Java, vamos fazer uma função, "testeInterface", a gente vai testar interface agora.

[03:57] O que a gente vai mandar como parâmetro, inicialmente vamos mandar a interface do Java, e ver o que acontece. A "transacaoDelegateJava", a gente vai colocar ela agora, deixa eu aumentar novamente o código, Ctrl+Shift+F12, colocar aqui o corpo dela, e a gente fez aqui uma função que vai testar a interface, "testeInterface", é só para a gente testar mesmo. E agora, vamos fazer as implementações que a gente conhece, vamos chamar aqui essa teste.

[04:32] Aqui dentro dessa teste interface, vamos fazer primeiro, a Object Expression, da mesma maneira como a gente viu anteriormente, só para a gente passar por todos esses pontos que a gente já conhece. Então object, agora "TransacaoDelegateJava", agora a gente coloca o corpo e a gente vai lá Alt+Enter, e a gente coloca os membros, reparem que a gente tem exatamente o mesmo comportamento.

[04:57] Só que agora, quando a gente coloca essa interface, reparem que o próprio Kotlin, no caso, o Android Studio que está alarmando, mas por debaixo dos panos, é o Kotlin, ele indica que isso aqui pode ser convertido, "convert to Lambda". A gente consegue fazer a conversão dele sem nenhum problema, a gente consegue fazer isso, agora vamos fazer esse mesmo comportamento, mandando como parâmetro apenas a Delegate, como a gente tá acostumado.

[05:18] Aqui eu vou voltar o código, para gente voltar exatamente como a gente viu, exatamente como a gente tinha feito antes, "transacaoDelegate", normal, e agora vamos fazer novamente, Object Expression, "transacaoDelegate ", só para a gente testar exatamente o que a gente fez nos dois, aqui é o Delegate normal. Ele permite fazer o Object Expression, a gente faz até lá em cima não tem problema.

[05:46] Só que agora, olha a diferença que a gente tem aqui, ele não fala que isso aqui pode ser convertido para uma expressão lambda. Se a gente quiser colocar como uma expressão lambda da mesma maneira como ele colocou, a interface do Java, não é possível também. Reparem que a gente não tem capacidade de fazer aquela mesma conversão que a gente fez, aqui na interface do Kotlin, e por que isso acontece?

[06:07] Para a gente entender o motivo de isso acontecer, a gente precisa entender como que a interface do Java, é convertida para expressão lambda, como que acontece isso, desde quando isso surgiu e o porque? Essa feature de converter uma Interface para a expressão lambda, ela vem de uma featuring do Java 8, ou seja, as interfaces que são conhecidas por possuírem um método abstrato único, ou seja, de um modo técnico, single abstract method, que é justamente as interfaces, que tem apenas uma única assinatura.

[06:43] Elas podem ser convertidas para a expressão lambda, então no Java 8, isso é possível. E o que isso significa, o que isso tem a ver com o Java 8? Justamente, por questões de compatibilidade, interoperabilidade, justamente aquela feature que o Kotlin permite a gente a gente usar tudo que a gente tem no Java dentro do código Kotlin, o Kotlin também manteve essa abordagem.

[07:02] É tanto que, o Kotlin, ele chama essa abordagem de converter uma Object Expression da interface do Java para a expressão Lambda de "SAM Conversion", que vem de Single, Abstract Method, e que vem de conversão. Essa conversão, só é possível realmente nas interfaces do Java, justamente por questões de interoperabilidade. Agora a gente começa a vir naquela dúvida, a gente nunca vai conseguir fazer isso apenas com o código do Kotlin?

[07:30] Não necessariamente, no caso, o que acontece é o seguinte, para interfaces do Kotlin, ele realmente não vai fazer essa conversão, e por que ele não vai fazer? Porque já existe um recurso exclusivo para esse tipo de comportamento a gente vai ver agora, é um recurso, no qual, a gente consegue mandar um uma função dentro de outra função, e consegue, fazer a implementação via expressão lambda.

[07:49] Sem ter a necessidade de mandar uma interface que foi, justamente, o caso que a gente mandou, para poder ter um comportamento do Delegate, é, justamente, isso que a gente vai ver agora. Deixa eu pegar todo esse teste que a gente fez e, agora, vamos testar, por exemplo, vamos criar esta outra função, que vai testar, "testaFuncaoDoKotlin", essa é a função que eu falei para vocês que é uma novidade agora.

[08:12] Como que vai funcionar este teste? Basicamente, vou deixar esse "transacaoDelegate" para a gente ter, mais ou menos o mesmo comportamento, o nome do parâmetro, para a gente entender que a ideia é a mesma. Basicamente, a gente está indicando que a gente quer mandar uma função, para mandar uma função aqui via parâmetro, a gente vai lá e coloca aqui os parênteses.

[08:29] Então a partir do momento que a gente está colocando esses parênteses, a gente está indicando que a gente está mandando uma função, só que o Kotlin fala o seguinte, se você quer mandar uma função via parâmetro, você também precisa indicar o que essa função vai retornar. Então, não somos obrigados a indicar isso, diferentemente, da função normal que a gente coloca aqui, que a gente subir entende que já tem o "unit", a gente precisa declarar um retorno por padrão.

[08:49] Independente, se a gente não quer retornar nada, a gente precisa colocar, como a gente faz isso? Coloca a flechinha aqui, da mesma maneira como a gente faz na expressão lambda, e agora a gente indica o retorno, no caso, a gente pode indicar o retorno do jeito que a gente quiser. A gente pode por exemplo, colocar uma String aqui, a gente pode por exemplo colocar o "int", qualquer tipo de valor aqui dentro, inclusive, a gente pode falar que não tem retorno nenhum, como a gente faz isso?

[09:10] Com o unit, como a gente sabe que tem em todas as funções aqui, por debaixo dos panos, aqui a gente já declarou uma função que está sendo recebida, via parâmetro aqui da função "testaFuncaoDoKotlin", Esse tipo de abordagem, ele também é conhecido como função de alta ordem, ou então, da forma técnica, ele é conhecido como Higher Order Function. Essa é a feature do Kotlin, que permite com que a gente vive um sonho aqui dentro de outras funções, e a gente consiga executar.

[09:41] Da mesma maneira como a gente fazia aqui com o "transacaoDelegate", só que desta maneira, a gente faz expressão lambda, sem ter necessidade de fazer o Object Expression, é justamente isso que a gente vai ver agora. Como que a gente pode fazer esse teste? Primeiro, deixa eu mostrar para vocês esse comportamento de quando a gente tenta implementar esta função, que é a "testaFuncaoDoKotlin", a gente vai colocar "testaFuncaoDoKotlin", e reparem, eu completei sem querer, de propósito.

[10:05] Mas, olha a sugestão que ele já dá para a gente implementar, já com a expressão o que é implementar com uma expressão Lambda é, justamente, dessa maneira que a gente implementa esse tipo de funções. Reparem que aqui a gente já tem, exatamente, esse tipo de comportamento, que é implementar com uma expressão Lambda, quando a gente tem esse tipo de parâmetro.

[10:19] Agora para a gente testar todo esse comportamento que eu te falei, para mostrar que é bem similar ao que a gente fez aqui no Delegate, vamos fazer o seguinte, vamos colocar alguns logs aqui, tanto aqui dentro da

"testaFuncaoDoKotlin", como também dentro da função Lambda, vamos ver como é a sequência de execução deles. O primeiro log que a gente vai colocar, vai ser justamente um log que vai testar a High Order Function, aqui como tag eu vou deixar a "hof", é isso que eu vou fazer.

[10:48] Agora que eu coloquei aqui a "hof", vamos colocar aqui uma mensagem que vai indicar que a "testaFuncaoDoKotlin", vai ser executada. Então eu vou colocar aqui como "testaFuncaoDoKotlin esta sendo executada", é a primeira mensagem que a gente vai esperar aqui, em seguida, o que a gente vai fazer? A gente vai lá e vai mandar aqui a nossa "transacaoDelegate", e vamos executar ela.

[11:14] A princípio não vamos mandar uma transação, é só para a gente fazer esse teste, não precisa se preocupar, que logo mais a gente vê como mandar esses parâmetros aqui, só para a gente fazer esse teste, a gente não vai mandar. Agora que a gente está chamando aqui essa High Order Function, que a gente declarou aqui em cima, vamos agora entrar aqui dentro da expressão Lambda, e ver quando é executado a expressão Lambda.

[11:34] A gente vai colocar um log aqui, e aqui a gente vai colocar "hof", que é como a Tag que a gente vai utilizar, e aqui, a mensagem que a gente vai deixar é "entrei na expressão Lambda", a gente vai ver o momento em que essa expressão lambda é chamada, a gente vai ver isso. Reparem, que só para a gente fazer esse teste, e ser executado rapidamente, vamos só corrigir alguns pontos que a gente deixou para trás.

[11:58] O primeiro deles, é justamente, esse problema de compilação que a gente deixou, que antes era implementado com a Object Expression, vamos tentar consertar ele aqui rapidamente. Aqui eu vou recortar esse carinha e agora eu vou apagar esse parâmetro da expressão lambda, que a gente tentou implementar, e novamente eu vou estar com o Object Expression, só para poder manter compilado, aqui, o código, Alt+Enter aqui, implementar os membros, vou lá e colo todo o código aqui.

[12:20] O outro ponto, só para poder testar rapidamente também, essa chamada da "testaFuncaoDoKotlin", eu vou colocar lá no "onCreate", que ele já executa tudo para nós, é aqui que eu vou deixar o teste. E agora a gente pode executar a nossa aplicação, Alt+Shift+F10, reparem que ele está colocando aqui o emulador, vamos verificar aqui. Veja que o Android Studio conseguiu executar, só para ver se está funcionando a nossa App, vamos abrir aqui o emulador, reparem que ela está funcionando normalmente.

[12:45] Agora a gente pode ver lá no nosso Logcat, para ver se realmente a mensagem apareceu. Entrando aqui no nosso Logcat, reparem que eu já até tinha feito um filtro antes, para poder colocar essa parte do "hof", para poder colocar aqui para a gente o nosso teste, e veja só o que aconteceu. Vamos recapitular aqui. Reparem que nesse momento aqui do "hof", a gente tem aqui o "testaFuncaoDoKotlin está sendo executada".

[13:09] Realmente, no momento em que a gente chamou o nosso "testaFuncaoDoKotlin", ele executou esse log que está dentro da função, ele executou esse log aqui, depois que ele executou a High Order Function, olha para onde ele veio, ele veio, justamente, para nossa expressão Lambda. Aconteceu exatamente aquilo que nós estávamos vendo, que é o mesmo comportamento que a gente teve no Delegate, ou seja, agente executou tudo que a gente precisava executar.

[13:34] No momento, que a gente precisava delegar para alguém, para fazer uma execução para que implementar a expressão Lambda, a gente conseguiu fazer enviando a High Order Function, exatamente o mesmo comportamento do Delegate. Inclusive, agora a gente entra naquele caso, e se eu quisesse mandar uma transação, por exemplo, a gente simplesmente, chegava aqui, e colocava a transação, então "transação", "transação", aqui a gente poderia criar uma transação, agora ele está pedindo ali.

[13:56] Vamos criar uma transação aqui, rapidamente, transação, a gente coloca o valor, "BigDecimal" de 100, o tipo eu vou colocar, "tipo=Tipo.RECEITA", agora vamos mandar essa transação aqui para dentro do High Order Function, primeiro pegando o objeto dela e depois mandando aqui para dentro. A gente está mandando a transação, agora na expressão lambda, da mesma maneira como a gente já viu anteriormente.

[14:26] A primeira das formas é mandando a transação aqui em cima, colocando apelido nela, eu estou colocando o apelido de transação mesmo, então a gente pode chegar agora aqui e fazer uma impressão dela, "log.i". Novamente, a gente coloca a Tag do "hof", para a gente ver que a gente está fazendo o teste da expressão lambda aqui, por meio da High Order Function e aqui a gente começa a colocar, "transação recebida da hof", e vamos colocar o valor das transações.

[14:50] Vamos imprimir aqui com o String Template, "transacao.valor", que vai ser o primeiro parâmetro, agora o "transacao.tipo", para a gente pegar os tipos que a gente colocou aqui. Veja que a gente está conseguindo aqui, fazer a impressão da transação a gente está recebendo da High Order Function. Vamos ver se isso funciona, Alt+shift+F10, veja que o Android Studio conseguiu executar, olha o que acontece aqui, "testaFuncaoDoKotlin" está sendo executada, nossa função foi executada.

[15:19] Agora, entrei na expressão Lambda, agora a transação recebida, a transação que tem o valor de 100 e uma receita. A gente conseguiu ter o mesmo comportamento, só que agora, a diferença, é que a gente consegue fazer a implementação da expressão lambda. Considerando esse aspecto, que a gente conseguiu colocar, para deixar o nosso código bem mais resumido, a gente vai fazer essa conversão logo mais nas nossas classes de Dialog, até já!