03

Implementando a Higher Order Function na superclass

Transcrição

[00:00] Agora que a gente conheceu a High Order Function no Kotlin, vamos fazer o seguinte, aquele nosso delegate que a gente está implementando com interfaces no Dialogs, vamos substituir para que agora eles utilizem a High Order Function. Porque, dessa maneira, a gente vai conseguir fazer essa implementação utilizando a expressão Lambda, sem exigência de utilizar um Object Expression. Vamos fazer isso agora.

[00:19] Primeiro, vamos apagar aqui esse teste que a gente colocou aqui, deixa eu apagar aqui, eu vou apagar essa função de teste que a gente estava fazendo, que era só para a gente ver como era o funcionamento, por isso eu estou apagando, a gente não vai reutilizar ele. Agora que a gente vai fazer? Vamos entrar agora nas nossas classes de dialog, aqui eu vou entrar primeiro na "FormularioTransacaoDialog", e vamos agora começar com essa conversão.

[00:40] O primeiro passo que eu vou tirar aqui é justamente esse comentário que eu acabei deixando um pouquinho para trás, mas agora a gente já pode tirar, não tem nenhum problema não impacta nada. Agora que a gente tirou esse comentário, vamos começar com esse passo de implementar a High Order Function. Reparem que aqui, a princípio, o que a gente está fazendo, a gente está mandando um parâmetro que representa a interface, a "transacaoDelegate", portanto, a gente pode fazer aqui agora.

[01:03] Ele vai ser uma função, que vai receber via parâmetro, ou seja, High Order Function, uma função de alta ordem, e agora a gente precisa definir o que é esperado dessa função. O que é esperado dessa função? A princípio, a gente pode definir um retorno, retorno a gente não espera nada, a gente não espera retornar algo daqui, portanto, a gente pode deixar como "unit", agora que a gente define o que o retorno é o "unit".

[01:25] A gente precisa de fato, indicar, que essa função que a gente está usando, que a gente vai delegar para alguém, vai enviar uma transação. Portanto, a gente precisa chegar aqui e falar que ela vai receber uma transação, porque é exatamente isto que a gente está utilizando no nosso delegate. Por isso, faz todo sentido a gente chegar aqui e mandar essa transação aqui para ele. Inclusive, reparem que antes a gente costumava colocar esse nome "TransacaoDelegate", por causa do nome da interface.

[01:51] Agora a gente quer apenas delegar, não precisa falar que é uma transação delegate, a gente vai delegar, ou seja, colocar o nome apenas como delegate, então Shift+F6 para renomear. Reparem que aqui delegate, a gente sabe que a gente está delegando uma transação, a gente não precisa deixar tão verboso assim, a gente pode diminuir essa verbosidade. A gente fez a primeira conversão aqui na função chama, que é a primeira função que recebe a interface.

[02:14] Mas repare, que aqui embaixo, quando a gente chama essa "configuraFormulario", a gente tem um problema de compilação, por quê? Porque a gente está mandando uma High Order Function, sendo que ele está esperando aqui, uma interface que é a "transacaoDelegate", então a gente também tem que fazer essa conversão. Como a gente pode fazer isso? Até mesmo o Android Studio nos ajuda nesse aspecto, porquê? Porque agora, a gente vai apagar todo esse parâmetro, e a gente vai começar a escrever, por exemplo, delegate.

[02:35] Reparem que ele é inteligente o suficiente para entender que a gente está querendo mandar o delegate que a gente declarou lá em cima, então a gente pode dar um "Enter", ele já faz isso para a gente.

[02:43] Percebam que aqui, a gente não está utilizando ele, porque esse nome não está sendo utilizado em nenhum momento dentro da função "configuraFormulario". A gente pode fazer o seguinte, a gente pode apagar essa referência da "transacaoDelegate", e simplesmente chamar este delegate, que a nossa High Order Function, enviando a transação que foi criada.

[02:59] A gente conseguiu fazer este primeiro passo, agora a gente conseguiu fazer com que a nossa Super Class, trabalhe com a High Order Function, o que a gente precisa fazer agora? Nosso próximo passo é verificar as classes filhas, verificar se elas estão atendendo o mesmo comportamento de High Order Function. Vamos entrar primeiro, na mais fácil, que é a "adicionaTransacaoDialog".

[03:19] Reparem que, ela utiliza a mesma função do chama, portanto, a gente não precisa declarar nada aqui dentro dela, porque ela só tá declarando apenas, os membros que ela precisa sobrescrever, o resto ela está usando da nossa Super Class. Agora a gente pode voltar a, ir para a "alteraTransacaoDialog", nela, agora a gente precisa chegar e também modificar, por que a princípio, ela está lidando com a interface "transacaoDelegate".

[03:42] Agora a gente pode colocar aqui como delegate, da mesma maneira como a gente fez lá na nossa "FormularioTransacaoDialog", e aqui a gente vai declarar a High Order Function, que vai receber uma transação, que é do tipo transação e, aqui, o retorno dela, vai ser o nosso "unit", aqui a gente vai lá e apresenta o Delegate. Só uma observação, que eu acabei não deixando tão claro assim.

[04:03] Só para vocês terem como feedback, para a gente mandar aqui uma High Order Function, os parâmetros que a gente tá mandando aqui, tanto nessa parte do tipo que a gente está colocando que tá recebendo via parâmetro, como também o retorno, precisam ser os mesmos, o que isso significa? Se a gente tentar retornar uma String, por exemplo, ele não vai compilar aqui, por que ele não vai compilar?

[04:21] Porque ele espera, de fato, uma High Order Function, que tenha uma transação via parâmetro e o retorno do "unit". Esse é um aspecto bem importante para a gente considerar no momento que a gente está colocando uma High Order Function, porque, tanto os parâmetros recebidos, como também o retorno, precisa ser igual.

[04:35] Esse é um feedback bem importante, para vocês saberem, no momento que vocês estiverem considerando uma High Order Function e uma implementação da mesma. A gente definiu agora aqui, nas nossas classes do "Dialog", portanto, a gente pode voltar na nossa Activity, e fazer a implementação dela, Ctrl+N aqui, agora eu vou para Activity, então lista, "ListaTransacoesActivity", e agora reparem que, no momento que a gente faz a chamada do chama, aqui nas nossas classes de Dialog, ele começa a quebrar.

[05:03] Porque ele está na implementação, do Object Expression, e a gente não faz implementação via Object Expression, para as High Order Function, a implementação somente via expressão Lambda. Portanto, esse código aqui, fazendo o Object Expression, não vai existir mais, a gente pode deixar a nossa expressão lambda, é claro, não tem esse "overhide" aqui dessa função, porque não existe ela. Agora percebam que a gente tem aqui a nossa expressão lambda, percebam também, que aqui ele não está conseguindo resolver a transação.

[05:28] Por que, a princípio, a gente não está definindo para ele o que é uma transação. O que a gente pode fazer? Como a gente sabe, a gente tem apenas um único parâmetro, então, a gente tem duas técnicas para fazer isso, a primeira delas, como a gente já viu, é justamente enviando o apelido por meio de transação, pode ser outro apelido também se quiser, "T", ou qualquer coisa que vocês preferirem. Essa é a primeira abordagem que a gente tem na expressão Lambda, só para poder recapitular.

[05:49] A outra abordagem que a gente tem é, justamente, a seguinte, o objeto subentendido, já que é apenas um único parâmetro, a gente pode utilizar o nosso "it", o "it" é a própria transação que a gente está recebendo. Então, é o objeto sub entendido, a gente pode fazer isso desta maneira, só para ordenar um pouquinho o código, vou utilizar o atalho Ctrl+Alt+L, ele ordenou aqui, só subir mais um pouco, e a gente fez a implementação da expressão Lambda.

[06:11] No caso, a implementação da High Order Function, com a expressão Lambda, aqui na função chama da "AdicionaTransacaoDialog", portanto, o que a gente precisa fazer agora, é só também converter a implementação da expressão Lambda, da High Order Function, aqui na nossa função, que faz alteração, da "AlteraTransacaoDialog", então

nessa chama aqui, a gente também vai fazer essa conversão, a gente pode até apagar esse código aqui, deixar apenas os parênteses aqui, aliás, as chaves.

[06:37] Aqui a gente vai ter exatamente o mesmo comportamento, só que agora a gente entra em um caso bem peculiar, por quê? Porque, a gente recebe uma transação, a gente está mandando a transação na função chama, e aqui dentro da altera, que altera a nossa transação, a nossa ListView, a gente está mandando novamente essa transação, mas qual é essa transação que a gente está mandando?

[06:57] A princípio, se a gente for olhar assim de cara, por cima parece que está vindo aqui, da nossa implementação da High Order Function, só que na verdade, não está, essa transação aqui, o que está acontecendo? Ela está sendo enviada a partir desse parâmetro, ou seja, a gente está enviando novamente o mesmo parâmetro. Então, aquela transação que a gente pegou lá da lista de transações, está sendo enviada para ser alterada novamente.

[07:17] Ou seja, o comportamento que a gente vai ter de alterar uma transação lá no nosso Dialog de transação, não vai acontecer, se a gente deixar o código desse jeito, porque, a gente vai mandar exatamente a mesma transação que não vai ser modificada. Portanto, nesses casos, a gente corre bastante riscos quando a gente utiliza esse tipo de abordagem, que é, justamente, por meio da expressão Lambda, que tem a capacidade de pegar todos os membros que estão um nível acima dela, de quem a chamou.

[07:39] Qual é a ideia que a gente pode, por exemplo, lidar com esse tipo de caso? Uma das ideias que vai nos proteger bastante, no caso uma boa prática para lidar com esse tipo de situação, é justamente, definindo bem um apelido, um apelido que tenha um nome bem distinto do que a gente está acostumado, do mundo afora que a chamou. Portanto, a gente poderia fazer um "transacaoCriada", por exemplo, ou alterada, "transacaoAlterada".

[08:01] Para poder indicar que, essa transação aqui a gente está recebendo é justamente, a transação que a gente quer mandar para esse altera. Essa é uma das abordagens que a gente pode fazer, e vai nos manter seguros, do nosso código, evitar esta questão de a gente estar mandando uma transação que a gente não quer, um objeto que a gente não quer, por questões de nome. Essa é uma das abordagens, a outra, novamente, usando o objeto subentendido.

[08:24] A gente pode vir aqui, e colocar o "it", porque, usando o "it", a gente garante, que essa transação que a gente está mandando para ser alterada, é justamente a transição que a gente está recebendo. Esse é um cuidado que a gente tomar, quando a gente está aqui na expressão Lambda, quando a gente vai fazer uma implementação de uma High Order Function, só para a gente ver se realmente está tudo funcionando, vamos testar o nosso código e ver o que acontece Alt+shift+F10.

[08:48] Veja que o Android Studio conseguiu executar aqui para a gente, vamos ver como ficou, a princípio, tudo funcionando, vamos testar aqui. Vou adicionar uma receita, uma receita de R$100, adicionei, tudo funcionando normalmente, agora eu vou alterar, apareceu para alterar, eu vou alterar o valor apenas, e ele foi lá e alterou. A nossa expressão Lambda, a nossa High Order Function, tudo que a gente converteu da interface para a High Order Function, para poder utilizar esta sintaxe aqui, mais resumida de expressão Lambda, funcionou.

[09:16] A gente conseguiu fazer com que nosso código funcionasse, sem depender daquela interface que a gente estava colocando, antes, que é maneira que a gente fazia no Java. A gente não precisa mais fazer desta maneira, criar novas interfaces, para fazer o mesmo comportamento do Delegate, portanto, o que isso significa? Significa que a gente não precisa mais desses Delegates, que a gente criou, a gente pode até removê-los, vou dar um delete aqui, agora a gente removeu.

[09:39] Mas é claro, por a gente ter usado ele em algum momento aquilo nosso código, a nossa aplicação vai quebrar, porque tem alguns imports que foram feitos, por exemplo, que fazem uso dele, a gente pode usar alguns atalhos para poder nos ajudar nisso, a gente utiliza o atalho Ctrl+Alt+O, e ele vai tentar ordenar para a gente, ele está pedindo para

fazer este processo por causa de um VSC, não precisa rodar, Ctrl+Alt+O de novo, provavelmente eu vou fazer alguma coisa aqui, a mais, Ctrl+Alt+O ele conseguiu ajustar sem nenhum problema.

[10:07] Agora que a gente fez essa parte aqui, a gente precisa verificar outros pontos, como que a gente pode verificar? A gente pode verificar da seguinte maneira, a gente pode utilizar um atalho, Ctrl+F9, o que é este atalho? É um atalho que faz um Build do projeto, tanto que, se você nunca o viu, fica aqui nessa parte do Build, "Make Project", Ctrl+F9.

[10:25] Ele vai lá e faz o Build para a gente, ele vai fazer todas as tasks do gradle, para verificar se não tem algum problema de compilação, ou alguma referência que não está sendo reconhecida, ele vai lá e nos alarma. Reparem que, aqui, ele está falando o que na "AlteraTransacaoDialog", a gente está usando ainda o Delegate, que nem existe mais.

[10:41] Então Ctrl+Alt+O, em seguida ele fala aqui também, nessa outra classe aqui, "FormularioTransacaoDialog", ele também está usando a mesma referência, então Ctrl+Alt+O de novo, agora ele está falando que não tem mais nenhuma, mas nenhuma achou com problema de referência. Então vamos executar novamente o Ctrl+Alt+F9 e ver o que acontece, veja que ele executou sem nenhum problema.

[10:03] Portanto, a gente pode até testar novamente, para ver se, esta modificação de remover uma classe, que é uma modificação bem perigosa mesmo, no nosso projeto, ela ainda está mantendo o nosso projeto intacto,"Alt+Shift+F10, veja que ele conseguiu executar, está funcionando normalmente, vamos testar novamente, um teste rápido, vou colocar uma transação de R$50, agora vou alterar aqui, agora R$150, ele está funcionando. Então está mantendo exatamente o mesmo comportamento.

[10:29] A gente conseguiu agora tirar aquelas interfaces que a gente tinha criado, para poder fazer o comportamento de Delegate, a gente não precisa mais dela, a gente vai estar agora utilizando a High Order Function, que é bem mais resumida de se utilizar, e não precisa dessa criação de arquivos extras, que foram as interfaces.

[11:43] Só que agora, por mais que essa implementação que a gente fez, é uma implementação que a princípio, parece bem legal, por que é uma novidade da linguagem, tem alguns detalhes que a gente precisa ficar muito atento, quando a gente implementa este tipo de implementação.

[11:56] Quando a gente implementa este tipo de solução na nossa aplicação, vamos entrar na nossa "ListaTransacoesActivity", e vamos ver como ficou essa chamada aqui, do nosso Dialog, perceba que ficou bem mais resumido, só que se, agora, a gente olhar de primeira e ler chama, ele manda tipo, e depois tem uma expressão Lambda.

[12:18] A princípio, para quem não conhece sobre o projeto, para quem não implementou esta classe, não é tão claro o que está acontecendo aqui e o porquê isto acontece. Portanto, este tipo de implementação, é legal, por ser resumido, mas ao mesmo tempo, ele sacrifica um pouco de leitura. Ao mesmo tempo, uma pessoa que vai começar agora no nosso projeto, não terá a capacidade de apenas ler este código, e saber o que está acontecendo aqui.

[12:41] A gente que implementou, a gente sabe, que aqui, ele está sendo executado, no momento em que ele consegue trazer uma transação para a gente, é bem claro que está acontecendo. Só que, para quem é novo, não sabe que isso está acontecendo, não é tão claro. Então, por mais que isso facilite a nossa vida, na hora de escrever código, ao mesmo tempo ele também sacrifica essa parte de leitura, como também, veja que a gente também não sabe com que tipo de informação a gente está lidando, apenas lendo esse código, não é tão claro assim.

[13:07] Em outras palavras, às vezes para a gente evitar esta parte de sacrificar a leitura e compreensão do código, vale a pena a gente tentar forçar o máximo possível, para deixar o nosso código fácil de ler. Ao invés de utilizar esse "it", por exemplo, a gente pode até utilizar algumas técnicas para minimizar essa incompreensão do nosso código, a gente poderia falar assim, "transacaoCriada", e a gente manda a transação criada aqui para adicionar.

[13:33] Já dá até um pouco mais de ideia do que está acontecendo aqui, por que aconteceu essa expressão Lambda, e entender que aqui tem uma transação criada, e ele, mais ou menos, tenta interpretar, não ajuda, como ajudava o outro, que é o caso do Delegate, que já sabia que a ideia do Delegate, é porque alguém estava delegando uma transação para a gente, mas já dá uma ideia do que está acontecendo.

[13:50] Então, se alguém bater o olho aqui, é muito mais fácil de entender o que está acontecendo com essa transação criada, do que apenas o "it", não era tão claro para ele, essa implementação foi uma implementação nossa, foi apenas um chama, o chama não indica muitas das coisas, diferentemente, por exemplo, do "filter", que era bem claro que a gente estava fazendo filtro, que ele tinha uma condição, é uma coisa bem comum para muitas pessoas.

[14:11] Agora, quando a gente implementa a nossa própria High Order Function, a gente tem que tomar muito cuidado com essa parte de compreensão do código, inclusive, tem uma técnica bem legal, só para não falar as coisas que são peculiares ruins. A gente tem a parte, na qual, quando a gente tem a High Order Function no final, a gente não precisa colocar os parentes para poder declarar ela, ela não precisa estar dentro do conjunto de parâmetros ali da função.

[14:32] A gente pode deixar da mesma maneira como a gente fazer aqui no "setOnClickListener", quando a High Order Function está no final dos parâmetros, a gente consegue fazer isto, a gente só muda aqui o tipo, fecha os parênteses, e vai lá e implementa a expressão Lambda.

[14:44] Da mesma forma para poder sempre a gente ficar atento com essa parte da Leitura, se preocupar com isso, a gente vai lá e coloca aqui, "transacaoAlterada", ou então modificada, alguma coisa que dê a entender que você está recebendo uma transação que foi alterada.

[14:59] O comportamento que você quer que a pessoa que tá lendo esse código entenda é que, isto está sendo executado quando a transação é criada e você toma uma ação baseando-se nessa informação que foi criada, porquê? Por que a expressão lambda não diz, exatamente, o que está acontecendo aqui, apenas se a gente deixar com o "it", apenas se a gente deixar com a expressão Lambda, não é tão claro, a gente tenta melhorar um pouco a leitura, quando a gente está usando esse tipo de abordagem, já para evitar este tipo de incompreensão.

[15:25] Novamente, a gente pode usar o mesmo recurso, por estar no finalzinho aqui, a gente vai lá e consegue implementar, da mesma maneira, só para executar e verificar se está tudo funcionando,Alt+Shift+F10, veja que o Android Studio conseguiu executar. Vamos ver aqui, vamos só testar agora com a despesa, a gente nem testa com a despesa, sempre com a receita. Vamos testar aqui, agora eu vou alterar, então vou deixar R$1200, vou colocar agora uma categoria diferente, de Comunicações, ele ainda está funcionando.

[15:53] Reparem que a gente conseguiu agora implementar a expressão lambda, a gente conseguiu deixar o nosso código bem mais enxuto, só que a gente viu que tem alguns pontos de atenção, que a gente precisa tomar muito cuidado, é bem legal mesmo essa implementação, ela facilita bastante na parte de escrita de código, para a gente não ter que ficar interpretando muito código, só que a gente tem que tomar cuidado.

[16:11] É um feedback que eu gostaria de passar para vocês, que é bem importante vocês estarem sempre considerando, antes de vocês considerarem novas features de uma linguagem nova, é sempre importante a gente saber essas partes problemáticas dela para gente evitar alguns problemas futuros, era isso que eu queria passar para vocês e até mais!