05

Utilizando membros abstratos

Transcrição

[00:00] Como vimos anteriormente, a gente conseguiu fazer a herança, no "AlteraTransacaoDialog", só que ainda tinha algumas peculiaridades que a gente tinha que resolver, que é justamente, a parte do título do dialog, e também do título do botão positivo do dialog. Então, pensando justamente, em resolver esse detalhe, a gente vai conseguir, agora, ver, como a gente pode fazer, para fazer uma implementação do título, tanto para Altera, como, também, uma implementação para Adiciona.

[00:22] A gente vai ver como a gente pode fazer isso. Só que agora, vamos poder fazer uma análise, para ver como que tá funcionando atualmente. Reparem que aqui na "FormularioTransacaoDialog", que é a nossa super classe, ela tem aqui uma implementação já da "tituloPor", que ele tá considerando a "adicionaReceita", como também, a "adicionaDespesa", que é justamente para o dialog e adição, então ele está fixo nesse momento.

[00:42] Como a gente já percebeu, a gente precisa fazer com que cada um desses membros, aqui que são as classes filhas da "FormularioTransacaoDialog", elas façam uma implementação própria delas, e não fique como responsabilidade da "FormularioTransacaoDialog", porque isso tem de a mudar de acordo com a ação. Então, qual que é a técnica, que a gente pode utilizar, para tá permitindo esse tipo de comportamento, isto é, fazendo com que essa responsabilidade não seja mais da "FormularioTransacaoDialog".

[01:06] E essas classes filhas, que é a "AlteraTransacaoDialog" e a "AdicionaTransacaoDialog", sejam responsáveis e obrigadas a implementar esse comportamento, que é o "tituloPor". A princípio, o que a gente pode fazer? Utilizando recursos da própria herança mesmo, qual é o recurso da herança que a gente pode obrigar que esse membro possa ser implementado pelas suas classes filhas? É tornando essa função abstrata, é isso, justamente, que a gente vai fazer agora. Como a gente pode fazer com que essa função seja abstrata aqui no Kotlin?

[01:35] Basicamente, o nosso primeiro passo para tornar qualquer membro abstrato é, justamente, fazendo com que ele seja abstrato por meio da Key Word "abstract". A partir dessa Key Word, a gente está falando o seguinte, essa função aqui vai ser abstrata, agora reparem, que a gente tem certas peculiaridades aqui, para serem solucionadas.

[01:53] A primeira delas é a seguinte, tudo que a gente torna como abstrato, não pode ter uma implementação. Portanto, esse corpo que a gente tem da "tituloPor" já não pode mais existir quando a gente torna ela abstrata, a gente vai dar um "Enter", e agora eu vou comentar o código, só para a gente ter, mais ou menos, como é a implementação aqui, para quando a gente for fazer nos outros, não se perder.

[02:12] Outra barra aqui e a gente comentou, então o primeiro paço a gente fez, agora a gente está indicando que as classes filhas vão ter que implementar uma função que, é chamada de "tituloPor", que vai receber um tipo e que vai devolver um "int", é isso que elas terão que fazer. Só que agora entram outros detalhes, toda vez que a gente torna alguém abstrato, o que acontece com uma classe que tá fazendo isso?

[02:33] Ela também precisa ser abstrata, já que ela vai manter menos abstratos, ela também não pode fazer uma instância, porque ela não tem implementação desse cara, quem tem a implementação desse cara é, justamente, quem está herdando dela, por isso ela, também, tem que se tornar abstrata, esse é o outro passo que a gente tem que fazer.

[02:50] Aqui a nossa classe "FormularioTransacaoDialog" vai receber o "abstract" também, agora, o que acontece? Quando a gente tem, aqui, um abstract de uma classe, a gente não precisa colocar mais, essa Key Word chamada de "open", porque a gente não precisa? Porque a "open" fica por debaixo dos panos, também, quando uma classe é abstrata,

ou seja, a classe quando é abstrata, já tem um comportamento padrão, de ser herdada por outras classes, justamente por isso, a gente não precisa colocar o "open".

[03:17] Claro, a gente pode deixar o "open" aqui, sem nenhum problema, só que até a própria linguagem, fala que isso daqui, é uma atribuição, uma declaração, ambígua, redundante, ou seja, não precisa disso daqui. Então a gente vai lá e vai apagar, a gente está tornando ela, apenas abstrato e, por debaixo dos panos, ela também já é aberta, para poder fazer a herança.

[03:36] Agora que a gente fez essa outra modificação, a gente também tem mais uma restrição, quando a gente tá fazendo o uso de abstração, dentro de algum membro. Essa outra restrição, trata-se justamente do nível de acesso, que a gente tem aqui, na nossa função, porque uma função abstrata não pode ser privada, o mínimo que ela pode ser, é só protegida, portanto, a gente vai deixar como "protected", essa é uma outra restrição que a gente tem.

[03:59] Agora a gente tem uma função que é abstrata, ou seja, esse comportamento aqui, ele precisa ser implementado, pelas classes filhas. Então, vamos fazer isso agora, vamos lá. A primeira que eu vou fazer é, justamente, copiar esse código aqui que trata-se, justamente da implementação da "AdicionaTransacaoDialog ", eu vou copiar ele aqui, agora eu vou até apagar, apaguei, agora ele já não está mais aqui dentro do "FormularioTransacaoDialog".

[04:22] Reparem que ele não tá mais compilando, porque ele exige que agora a gente implemente os membros abstratos, que nesse caso, é a função "tituloPor", Alt+Enter, o Android Studio já nos ajuda, "Implement Members". Reparem que ele até já mostra o que precisa ser implementado, que é a "tituloPor". Agora a gente fez essa implementação aqui, ele vem que aquele "TODO" normal, vamos apagar que a gente já vai implementar aqui agora, e aqui, eu vou colar aquele código, eu só vou descomentar aqui, Ctrl+/.

[04:48] Eu já tenho aqui implementação, eu até tinha copiado, mais do corpo aqui que não era necessário, porque eu copiei o corpo da função também, e aqui só falta importar a classe "R", e a gente já tem a implementação do "AdicionaTransacaoDialog". A gente já tem essa implementação aqui. Inclusive, a gente pode até meio que copiar, e mudar os parâmetros do nosso "AlteraTransacaoDialog", então, lá no "AlteraTransacaoDialog", a gente vai vir aqui agora e implementar esse membro.

[05:11] "Implemente Member", "tituloPor", o mesmo comportamento, a gente vai vir e vai colar aqui, ele vai pedir para importar a classe "R", e agora só modificar para "altera_receita" e "altera_despesa", também a gente já tem a nossa função implementada. Vamos ver o que acontece quando a gente tenta usar o "AlteraTransacao" e "AdicionaTransacao", agora que elas estão implementando o "tituloPor", vamos lá, Alt+Shift+F10, veja que o Android Studio conseguiu executar para gente, vamos ver o que acontece.

[05:46] A gente vai tentar adicionar, ele está funcionando normalmente, "Adiciona receita" e o adicionar aqui, vamos colocar aqui um valor, só para poder preencher, e adicionar. Ele adicionou sem nenhum problema, vamos clicar para tentar alterar. Reparem que agora o título do dialog foi para "Alterar receita", já não tá mais como "Adicionar receita", mas ainda assim o botãozinho aqui embaixo está como adicionar.

[06:09] Ou seja, o mesmo comportamento que a gente fez para permitir que a gente alterasse aqui esse título, de acordo com a classe que está sendo instanciada, a gente também vai fazer aqui para o adicionar. A gente vai fazer o mesmo comportamento, reparem que funciona, justamente, fazer com que essa implementação seja abstrata e, quem for implementar a transação, aliás, o "FormularioTransacaoDialog", vai ter que fazer essa implementação também, desse membro abstrato.

[06:32] É justamente isso que a gente vai fazer agora, também, para esse botão do adicionar. Como que a gente pode estar fazendo isso? Vamos lá no "FormularioTransacaoDialog" e vamos ver como está funcionando essa parte do botão positivo aqui. Reparem que aqui, ele tá sendo uma String bem solta aqui no meio, a princípio a gente não tem como fazer com que isso daqui seja abstrato, porque ele já tá dentro dessa função, qual é o passo que a gente pode fazer?

[06:55] Um dos passos que a gente pode fazer? É fazer com que tenha uma função, que devolva esse título do botão, é uma das maneiras que a gente pode fazer, mas também tem uma outra técnica aqui no Kotlin, que permite também que a gente aplique esse mesmo conceito do abstrato para properties, a gente também tem esse tipo de conceito. Vamos fazer isso para a gente ver como funciona.

[07:14] Vamos transformar primeiro, essa String que a gente está vendo, que é a "adicionar" aqui, em uma "property", como a gente pode fazer isso? A gente pode até usar um atalho para isso, o atalho que a gente pode utilizar e o Ctrl+Alt+F, reparem que ele tá falando o que a gente quer transformar como uma property, no caso, a gente só quer adicionar essa String, vou dar um "Enter" aqui, e ele pergunta, você quer adicionar como uma property do arquivo, ou da classe, no caso esse primeiro é da classe, então é da classe que eu quero adicionar.

[07:40] Ele preenche esse campinho, vou dar um "Enter", porque não tá mostrando nada para gente, e ele já cria para gente, basta apenas a gente mudar o nome, Shift+F6, e agora a gente vai colocar como "tituloBotaoPositivo", para deixar bem descritivo o que ele significa para gente. Ele até pergunta se a gente quer fazer a refatoração, para aplicar também, lá embaixo, sim, a gente quer fazer, agora ele aplicou aqui para gente. Eu vou trazer aqui para cima essa property, Ctrl+Shift+(seta para cima) e ele pula todos esses membros aqui.

[08:11] Agora reparem que aqui, a gente deixou esse carinha como uma property da classe, e o que a gente pode fazer agora para permitir com que esse comportamento, seja implementado pelas suas filhas e, não pela "FormularioTransacaoDialog"? Basicamente, da mesma maneira que a gente fez lá embaixo, na nossa "tituloPor", a gente pode chegar aqui, e colocar a Key Word Art "abstract", e no momento que a gente coloca essa Key Word "abstract", a gente não implementa mais, a gente não inicializa mais esse valor.

[08:39] Quem vai se responsabilizar por isso, são as classes filhas, só que agora tem um detalhe, quando a gente faz esse tipo de implementação, esse tipo de declaração, utilizando properties abstratas, a gente precisa indicar qual é o valor esperado por ela, a gente precisa deixar explícito, o que a gente espera desse tipo de valor. Porque, senão, as filhas poderiam colocar qualquer valor, que não é o esperado nesse momento, a gente tem que especificar qual é o tipo que a gente quer, no caso é uma String.

[09:06] Agora a gente tá fazendo essa primeira parte, repare que ele não tá deixando aqui essa parte do privado, porque que não deixa como privado? É por causa do abstrato, porque o abstrato não permite que a gente utilize essas Key Words, no caso, esse modificador de acesso privado. Então a gente tem que restringir no mínimo só o "protected", no máximo, no caso, a restrição máxima é o "protected".

[09:29] Agora que a gente colocou aqui essa nossa property abstrata, a gente vai vir aqui nos nossos membros, novamente a gente tem que implementar aqui, Alt+Enter, reparem que agora a gente vai ter que implementar, aliás, ele até da outra opção, para implementar essa property via Construtor se a gente quiser, nesse caso a gente não quer que alguém mande via construtor esse título, a gente quer que a única responsabilidade desse título venha só de dentro do corpo da "AlteraTransacaoDialog", como também da "Adiciona".

[09:52] Então não vamos pegar essa primeira opção, a gente vai pegar essa segunda, porque a gente só vai fazer essa implementação internamente, ninguém tem que saber que isso existe. Vou dar um "Ok" aqui, ele já fez aqui a implementação para a gente do "TODO", reparem que ele fez aquele esquema que a gente já viu, lá no nosso resumo, que a gente implementou uma vez, converter uma Single Expression Function, para uma property, é justamente isso que ele tá fazendo aqui para a gente, ele está deixando como uma property.

[10:15] A gente está modificando quem? O "get" dela, é isso que a gente vai fazer agora. Para a "Altera", o que a gente vai deixar como título do botão positivo? Basicamente, a gente vai vir aqui e deixar como "Alterar", é esse o título que vai deixar aqui para ele, agora, vamos lá na "AdicionaTransacaoDialog", o que a gente vai fazer aqui? Alt+Enter, "Implement Members", a mesma coisa também, a gente está implementando, e aqui, o título dela vai ser "Adicionar".

[10:40] Reparem que, agora, essa responsabilidade do título do botão positivo, como também do título do Dialog, está sendo como responsabilidade das classes filhas, a Super Classe, a classe mãe, a classe Pai, ela não tá tendo mais essa responsabilidade, ela não implementa mais esse tipo de comportamento, no caso, que a nossa property e também a nossa função, ela não faz mais isso. Vamos testar para ver se funciona, vamos lá, Alt+Shift+F10, veja que o Android Studio conseguiu executar, vamos ver aqui.

[11:09] Vamos adicionar uma transação, R$ 100, agora vamos tentar alterar, quando a gente vem aqui no alterar, a gente está mantendo aqui o título, como a gente viu no dialog, e agora, olha aqui embaixo, a gente tem o nosso botãozinho de alterar. Vamos só testar para ver se realmente está alterando as informações, como esperado, vamos pegar uma categoria diferente, ele tá alterando, se a gente abre novamente ele mantém as informações.

[11:31] A gente conseguiu fazer essa implementação, a gente conseguiu colocar uma herança, na qual, a gente conseguiu reutilizar o máximo possível de membros, que eram comuns entre as classes filhas, e a gente fez com que os membros que não eram comuns, que cada uma tinha que implementar por si só, a gente conseguiu delegar para cada uma delas, utilizando o abstract da herança. Só para poder fechar essa parte aqui da herança, das nossas classes filhas, vamos fazer só uma refatoração dessa nossa "AlteraTransacaoDialog".

[11:58] Porque a gente têm esse código grande aqui, para tentar interpretar o que ele faz, e se a gente perceber, o que ele tá fazendo aqui para gente? Ele está inicializando os campos. Vamos fazer essa refatoração, extrair essas funções, só para deixar bem mais fácil o nosso código. O nosso primeiro passo é selecionar tudo isso daqui, Ctrl+Alt+M e, agora, a gente vai colocar um nome para todo esse código, que é justamente "inicializaCampos", é isso que ele está fazendo.

[12:22] Nesse "inicializaCampos", a gente vê que a gente já tem esse código grandão, e a gente pode diminuir ele mais ainda, que é justamente pegando, por exemplo, essa parte que inicializa o campo do valor, extraindo para uma função, "inicializaCampoValor". Da mesma forma, a gente pode fazer para o data, a gente seleciona aqui e coloca, vamos lá, selecionei, Ctrl+Alt+M, "inicializaCampoData", e da mesma maneira para esse campo da categoria, que a gente fez todo esse código aqui, que fica "inicializaCampoCategoria".

[12:57] Reparem que agora fica bem mais fácil da gente entender os passos que estão acontecendo aqui, que é, justamente, primeiro a gente pega o tipo, depois a gente faz a chamada da nossa "chama" da nossa superclass, e depois a gente inicializa os campos, por meio dessa transação e do tipo aqui que a gente está tendo. Reparem que ele tem até uma certa diferença da forma de chamar, porque a gente está usando o tipo da transação, ele tá mandando o tipo e a transação, juntos.

[13:19] A gente pode até fazer uma modificação aqui nessa parte, como assim uma modificação? O que isso significa? Significa o seguinte, a gente não precisa dessa parte do campo, deste tipo aqui, a gente pode fazer com que o tipo, também, venha a partir da transação. Então, aqui internamente da "inicializarCampos", a gente pode fazer o tipo venha aqui da transação, "transacao.tipo", a gente pode fazer dessa maneira, que não precisa ficar mandando dois parâmetros, só manda um único parâmetro, que já é a própria transação.

[13:42] A gente pode estar fazendo isso daqui também, só novamente para poder pegar, agora a gente tá mandando aqui o tipo, e agora a gente fez essa refatoração e ficou muito mais fácil da gente entender o que está acontecendo, inicializa os campos, agora inicializa o campo do valor, depois da data, depois da categoria. A gente até sabe os passos que estão acontecendo aqui.

[13:59] Agora a gente já fechou essa parte da herança e até mais!