01

Inicialização atrasada com lateinit

Transcrição

[00:00] Como vimos anteriormente, essa abordagem na qual, a gente está utilizando tipos, que podem receber "null", tende a ser trabalhosa, aqui, dentro do Kotlin, porque a gente tem que sempre garantir que aquele valor não é "null", para evitar o famoso Null Pointer Exception.

[00:13] Só que aqui no Kotlin, a gente está sendo obrigado a inicializar a nossa variável e foi, justamente, por esse motivo, que a gente chegou nesse caso do "null", porque a gente não tem um valor específico para colocar agora, então a gente colocou o "null" e a gente teve que colocar com que a nossa variável pudesse receber esse tipo, que no caso, é o "null", esse tipo de valor.

[00:32] Só que agora, eu havia comentado com vocês, que existem outras técnicas, que permitem com que a gente evite esses tipos de cenários, que é justamente, ter que lidar com o "null" que, como a gente viu, é trabalhoso. Então a gente vai ver, justamente, nesse momento, essa técnica, como que funciona essa técnica, como que o Kotlin permite que a gente evite esse tipo de situação?

[00:50] Ele fala o seguinte, que, por padrão, ele vai exigir que a gente inicialize as variáveis, só que, também, ele vai nos dar uma outra alternativa, como seria essa alternativa? Essa alternativa é a seguinte, ele vai falar olha, você pode fazer com que, essa variável, seja inicializada depois, uma inicialização posterior e, como que a gente pode indicar que essa inicialização vai ser feita depois? Como a gente pode fazer isso no Kotlin?

[01:12] Basicamente, a gente pode chegar na nossa property e escrever essa Key Word, chamada de "lateinit", essa inicialização vai ser feita depois. Dessa maneira, a gente está indicando para o Kotlin que a gente não quer fazer essa inicialização neste momento, a inicialização não pode ser feita nesse momento, a gente vai fazer em algum outro momento e a gente vai se responsabilizar por isso, então é, justamente isso, que o "lateinit" nos permite.

[01:36] Agora, a gente não precisa ter que, primeiro, inicializar nossa variável, não precisamos disso e, também, a gente não precisa indicar que esse valor é um valor que pode receber "null". No caso, o "lateinit" nem permite que a gente coloque esse valor recebendo "null", ele não permite Isso, ele já até nos protege desse tipo de abordagem, é bem bacana.

[01:57] E agora que a gente fez essa abordagem do "lateinit", estamos nos responsabilizando em colocar esse valor, sabe se lá em algum momento. Essa que é proposta do "lateinit", é permitir que a gente evite essa abordagem de inicializar a property, logo no momento que a gente declara ela e inicializar em algum momento, sabe-se lá quando. Assim, evita essa abordagem de, também, ter que colocar ou lidar com valores "null".

[02:22] Vamos fazer um primeiro teste, que é ver o que acontece quando a gente está utilizando essa "lateinit" e a gente não inicializa. Esse é o caso mais interessante da gente verificar qual a diferente entre o "lateinit" e o que a gente fazia quando o valor era apenas "null". Vamos verificar, vamos agora executar Alt+Shift+F10, veja o Android Studio conseguiu executar nossa App, vamos ver o que acontece. Reparem que ele deu um problema, ele deu um crash, vamos ver qual foi o motivo para ele ter dado crash.

[02:51] Ele deu uma Exception, vamos ver onde ele indicou, ele indicou justamente nesse momento que a gente tentou fazer o uso da nossa property, ele deu essa Exception, vamos ver qual foi a Exception.

[03:04] A Exception foi essa "UnitializedPropertyAccessException", ele tá falando que essa property que foi feita com a inicialização posterior, que a gente vai fazer depois da nossa "viewDaActivity", ela não foi feita, ela não foi inicializada.

Em outras palavras, ao invés de dar um Null Pointer Exception, ele está indicando que a Exception ocorreu porque, o valor que agente indicou, que a gente iria inicializar, não foi inicializado.

[03:29] A abordagem é até diferente, não temos que tomar um Null Pointer Exception em algum momento muito profundo na nossa aplicação. Em algum momento que a gente executa, a gente tenta fazer o primeiro uso da property que a gente falou que iria inicializar. Essa que é uma diferença da abordagem do "lateinit" em relação ao que a gente utilizava com as variáveis que podem ser nulas, porque, a qualquer momento, ela poderia mudar o valor para "null" e a gente tomaria aquele Null Pointer Exception, nesse caso não.

[03:55] A primeira, das coisas que o "lateinit" faz é, justamente, proteger a gente do Null Pointer, então ele não recebe "null" e o que ele faz agora é, se você não me inicializar, eu simplesmente vou falar que você não me inicializou e vou quebrar sua aplicação. Essa que é a abordagem do "lateinit". Agora vamos ver o que acontece se a gente inicializar, eu vou tirar esse comentário aqui no momento que a gente inicializa e vamos ver o que acontece.

[04:16] Alt+Shift+F10, veja que o Android Studio conseguiu executar. Vamos ver agora, a nossa App continua funcionando. A gente consegue, por exemplo, adicionar transições aqui, a gente consegue fazer, exatamente, tudo que a gente fazia antes, quando a gente realmente tinha o valor da nossa "view", da "viewDaActivity", a gente consegue fazer a mesma coisa. Só que agora, a diferença, é que a gente não tem que lidar com valores que recebem "null".

[04:46] A gente não precisa lidar com isso, porque o "lateinit" nos permite que a gente inicialize as nossas variáveis, em algum momento, sabe-se lá quando, porque a gente não pode inicializar nesse momento, como a gente viu anteriormente, e se a gente não inicializa, ele já dá um crash, falando, você precisa inicializar, senão, você não vai conseguir executar a sua aplicação e, quando a gente inicializa, ele vai lá e funciona tudo.

[05:06] Em outras palavra, a gente acabou de fazer aqui é, evitar que a gente tenha que lidar com valores nulos. Agora, pensando justamente nesse detalhe, se a gente voltar na nossa resumo "view", a gente percebe que a gente não precisa mais ter esse cenário, de ficar recebendo um valor pode receber "null".

[05:21] A gente pode voltar com aquela abordagem, na qual a gente estava apenas utilizando aqui o nosso objeto, porque a gente tinha a garantia de que ele já não ia receber "null", e a gente não precisa preocupar com isso. Olha que interessante, veja que apareceu esse "it", por causa do escopo do "let", vou voltar aqui com a "view", voltei com a "view", deixa eu também fazer isso para os outros caras, eu fiz, primeiro para o "adicionaReceita", agora para o "adicionaDespesa".

[05:42] A gente não precisa mais lidar com essa parte do "null", não precisa mais nos responsabilizar em ter que verificar, esse valor aqui pode ser nulo, deixa eu ter certeza que ele não é, a gente perde toda essa responsabilidade, simplesmente mudando a variável que poderia ser nula, para poder usar uma inicialização que vai ser feita depois, com o "lateinit". Deixa eu só dar um Ctrl+Alt+L para ele poder organizar, ajustar essas linhas aqui, que é uma questão de TOC mesmo, vamos lá, aqui ajustei a linha.

[06:11] Agora a gente não tem que lidar mais com valores nulos e deixamos o nosso código bem mais simples, reparem que com essa abordagem que a gente fez do "lateinit", a gente deixou o nosso código bem mais simples. A gente não tem mais que tem que lidar com valores nulos, a gente não precisa mais se preocupar com isso, e quando ocorrer aquela situação que a gente não inicializa, ele vai lá, e avisa isso para a gente.

[06:32] Só que agora tem alguns detalhes dessa parte do "lateinit", por mais que ele tenha resolvido essa parte a gente. Reparem que aqui, a gente fez esse □lateinit", utilizando uma property "var", vamos ver o que acontece quando a gente tenta utilizar uma "val", vamos lá. Olha o que acontece, o □lateinit" não permite com que a gente utilize variáveis "val", ou seja, ele vai sempre ser uma variável que a gente vai ter a possibilidade de mudar o valor dela. Então existe esse risco que a gente utiliza essa abordagem do "lateinit".

[06:58] A gente vai ter sempre que lidar com valores "val", porque ele vai mudar em algum certo momento, por isso, ele vai fazer com que a gente tenha que utilizar o "var". Uma outra abordagem, também, que é importante saber quando a gente está utilizando esse "lateinit", é que, aqui a gente está utilizando apenas uma única property, a gente não sabe exatamente qual é o momento que ela vai ser inicializada.

[07:18] É claro, por ser uma única property, por estar bem perto aqui do "onCreate", a gente tá vendo que o "onCreate" está inicializando isso para a gente, só que, se caso, a gente tivesse mais properties que fossem "lateinit" também, olha o tanto de responsabilidade que a gente teria que ter, em analisar, qual é o momento que essa variável está sendo inicializada, ou onde ela precisa ser inicializada, entre outras coisas que acabam tendo complicações.

[07:40] Por mais que o "lateinit" nos ajude nesse momento, existem casos que ele pode ser um pouco mais complicado. Então, considerando esse contexto, logo mais a gente vai ver uma outra técnica que a gente tem, para poder, também, ajustar esse tipo de inicialização, que não pode ser feita nesse momento, mas que vai fazer com que evite essa situação em que ela vai ser inicializada depois e a gente não sabe exatamente aonde está essa inicialização. A gente vai ver essa técnica logo mais, até já.