03

Conhecendo o null safety

Transcrição

[00:00] Como vimos anteriormente, no momento que a gente utilizou a nossa "viewDaActivity", que é property que a gente quem criou, inicializando com o "window.decorView", o nosso código quebrou, porque isso aconteceu?

[00:11] Se a gente reparar aqui no erro que aconteceu, a gente percebe que a gente tomou um Null Pointer Exception, como a gente tinha visto aqui, e ele tomou esse Null Pointer Exception, justamente, no momento que ele chamou esse objeto "window" e depois o "getDecorView". Então, nesse momento, ele tomou um Null Pointer Exception, porque isso aconteceu?

[00:27] Se a gente lembrar, logo do princípio das Activitys do Android, a gente percebe que elas são classes um pouquinho especiais, porque são um pouquinho especiais? Porque, além de só construí-las, além daquela questão de instância, construtor, entre outras coisas que a gente conhece do Java, elas também possuem essa questão de ciclo de vida, que é, justamente, o "onCreate", o "onResume", o "onStop", entre outros existentes, o que isso significa então, comparado com essa questão do "window.decorView"?

[00:55] Significa o seguinte, para a gente conseguir pegar o valor da "view" da nossa Activity, sem tomar um Null Pointer Exception, a gente precisa, de fato, ter essa view criada, mas quando que ela é criada aqui na nossa Activity? É, justamente, quando a gente coloca essa função "setContentView", mas quando que essa "setContentView" é chamada? No "onCreate", e o "onCreate", quando ele é chamado? Em algum momento, sabe-se lá quando, mas não é no Construtor e nem antes do Construtor.

[01:22] Ou seja, quando a gente faz essa inicialização aqui, a gente está fazendo essa inicialização no momento em que a nossa Activity está sendo construída como uma classe do Java, e não como uma Activity do Android e, justamente, por esse detalhe, a gente está tomando o Null Pointer Exception. Então, como a gente poderia lidar com esse tipo de situação?

[01:40] Como que a gente poderia, por exemplo, utilizar uma property para poder ter o mesmo comportamento que a gente tem, para poder entregar uma view, que é a própria view da Activity? Como a gente pode fazer isso?

[01:51] A princípio, a gente pode fazer da seguinte maneira, logo depois que a gente "seta" a view, a gente pode vir aqui e falar, a view da Activity, o valor dela é a "window.decorView", essa é uma das abordagens que a gente pode fazer, aqui, novamente, por ser uma property que vai ser inicializada, em algum momento, sabe-se lá quando, mas não é no Construtor, a gente vai lá e deixa como "var", para poder alterar esse valor.

[02:19] Só que ainda não resolvemos o problema necessariamente, porque, aqui, nessa "window.decorView" a gente ainda está com um objeto que é nulo, então a gente vai tomar Null Pointer Exception novamente. A gente pode até testar e ver se realmente acontece, vamos ver, Alt+Shift+F10, eu vou dar um "Enter" e vamos ver o que acontece.

[02:42] Veja que o Android Studio executou, mas vamos ver o que acontece na nossa App, vou pedir para abrir novamente, porque não tinha clicado nisso anteriormente, e ele quebra de novo, e se a gente ver, a gente toma novamente o mesmo Null Pointer Exception, aqui, falando da "window.decorView". Nesse momento, a gente pode até ver aqui, ele até indica onde foi, que é justamente aqui nessa inicialização.

[03:03] Se a gente perceber o que a gente está fazendo aqui, a gente não pode inicializar essa variável que a gente tem aqui, com o nosso "window.decorView". Então, como a gente pode lidar com esse tipo de situação, qual o valor que a gente pode colocar, apenas para poder inicializar a property? Porque, se a gente deixar desta maneira, aqui, olha o que

acontece, o Kotlin fala que a gente precisa inicializar, a gente precisa inicializar essa property para ele permitir que a gente compile o nosso código.

[03:26] Qual é o valor que a gente pode colocar aqui, que ele não diz nada, ele não atribui o valor que a gente espera? Um valor que a gente pode colocar aqui é, justamente, o valor do "null". A gente pode colocar esse valor, agora, olha o que acontece, no momento que a gente colocou esse "null", olha o que que ele inferiu como valor desse "null", ele inferiu que esse "null" está sendo esse cara chamado de "Nothing".

[03:48] A gente não tem mais a nossa "view", a gente não pode mais utilizar, por exemplo, a nossa "window.decorView" para reatribuir o valor, porque ele já assumiu que nossa property está sendo um "Nothing", que é justamente uma classe que representa nada, o valor que não tem nada, o que a gente pode fazer para poder reatribuir?

[04:05] A gente tem que falar que a gente não espera o "Nothing", a gente não quer que ele atribui isso como "Nothing", a gente espera esse cara seja uma "view", é o que a gente espera que ele seja, só que agora, olha o que aconteceu, no momento que a gente colocou uma "view", o próprio Kotlin já nos alarmou e deu um erro aqui no "null", porque que ele fez isso?

[04:22] Ele diz o seguinte, que quando a gente tá querendo colocar em valor "null", numa classe que não pode receber "nul", isso não vai ser possível. Então o Kotlin vem novamente com aquela abordagem de proteger nós programadores, isso é, ele não permite, a princípio, que a gente coloque valores "null" em alguma classe que a gente está querendo colocar, no caso, uma property que vai ter o tipo "view" ele não permite isso, por que ele não permite?

[04:47] Porque ele considera que todas as classes que a gente está utilizando, são classes que não podem receber "null". Então por padrão o Kotlin trabalha dessa maneira, mas, agora, a gente entrou num caso em que "null" se faz necessário para a gente, ele é necessário neste momento, apenas para a gente ter a nossa property, para ela ser inicializada em algum outro momento, que a gente vai inicializar.

[05:09] Como a gente pode chegar no Kotlin e falar o seguinte, olha Kotlin, para esse momento eu quero que essa variável, aqui, possa receber valores "null", eu sei que você, por padrão, você não você não quer que essa variável receba "null", mas eu quero que ela tenha a possibilidade, como a gente pode fazer isso? A gente pode chegar e falar o seguinte, a gente pode colocar esse operador de interrogação.

[05:29] A partir do momento que a gente coloca esse operador de interrogação e uma classe, quando a gente está colocando o tipo dela, a gente está falando que essa classe vai ter esse tipo, que é no caso, o tipo "view", só que ela também pode receber um valor "null". A gente está permitindo essa possibilidade, então, aquele segredo que eu deixei para você para falar mais para frente, que no caso está sendo agora, que é justamente, aqui, no nosso "Bundle" e o interrogação, é que esses tipos podem receber valores "null".

[05:56] É dessa maneira que o Kotlin trabalha e ele permite que a gente trabalhe com valores "null", para nos proteger, justamente, daqueles códigos que podem receber Null Pointer Exception. Essa é a ideia do próprio Kotlin de evitar esse tipo de situação, tanto que, essa abordagem de contexto geral, é conhecida como Null Safety, que é, justamente, a segurança em relação a valores "null", a gente vai ver agora algumas abordagens que a gente pode considerar nesse tipo de situação aqui no Kotlin.

[06:22] O princípio dessa parte do Null Safety é, justamente, indicar que uma variável, no caso, um tipo dessa variável pode receber "null", aqui, ele permitiu, só que, agora, começam a vir alguns detalhes. Quando a gente tá indicando que essa variável pode receber o valor "null", em todos os pontos do código, que a gente utiliza ela, vai começar a dar alguns pepinos, como assim dar alguns pepinos?

[06:45] Porque, perceba, aqui, por exemplo, quando a gente está utilizando essa property para, por exemplo, colocar uma variável que é do tipo "view", reparem que ele não está permitindo, por que não tá permitindo? Porque, aqui, a

gente tá esperando que essa variável que a gente está atribuindo, seja uma valor que não é nulo, por padrão, ela não pode ser nulo, ou seja, nesse momento que a gente está atribuindo essa variável, a gente precisa, no caso, ou falar que ela pode receber nulo, ou a gente precisa tratar essa nossa property.

[07:15] A gente precisa tratar ela e indicar que, ela pode, ou não, ser nulo, nesse primeiro momento, vamos tentar apenas enviar do jeito que está aqui e ver o que acontece, a gente vai ver, passo a passo, como a gente pode lidar que esse tipo de situação, a gente até ver que a gente não precisa desse objeto, a gente pode mandar até o próprio "viewDaActivity" aqui dentro, a gente só está atribuindo aqui porque, antes, a gente pegava o "decorView", deixa eu pegar aqui, copiar, eu vou apagar primeiro essa linha.

[07:40] E, agora, eu tô mandando a "viewDaActivity", diretamente, reparem que, novamente, ele vem com aquela ideia, de que a gente precisa mandar uma "view", só que a gente está mandando uma "view" que pode ser nula, vamos entrar agora no "ResumoView" e vamos ver o que a gente pode fazer para lidar com esse tipo de situação.

[07:58] Ctrl+V, vamos lá, a princípio a gente tem esse "context", deixa eu até tirar porque a gente não está utilizando essa property, vou deixar só como parâmetro normal, a princípio, reparem que a gente está recebendo a "view". A gente recebe uma "view" que a gente espera que é um valor que não vai ser nulo mesmo, então, nesse momento, só para a gente ver o que acontece, vamos falar com que agora esse valor pode receber nulo, ou seja, lá na Activity, agora, está tudo certo.

[08:23] Eu estou mandando uma variável que pode ser nula, a "ResumoView" vai lidar com isso, ela espera já esse tipo de valor, só que olha o que acontece aqui na nossa "ResumoView". Quando a gente está utilizando essa nossa property, que ela pode ser nulo, em todos ospontos que a gente está utilizando seus membros, eles quebraram, aqui, quando a gente pega a "view.resumo_card_receita", que é o componente da receita como também o "resumo_card_despesa", como também o "resumo_card_total", o nosso código não compila.

[08:53] Por que isso acontece? Porque o Kotlin vem com aquela abordagem, na qual, se o valor pode ser nulo, existe um risco grande de você tomar o Null Pointer Exception, portanto, você precisa lidar com esse tipo de situação, e como a gente pode, então, lidar com esse tipo de situação, aqui, no Kotlin?

[09:10] A primeira abordagem que a gente pode fazer é, a seguinte, a gente pode falar para o Kotlin que, quando a gente tá nesse tipo de situação, nós seremos responsáveis em falar, esse código pode ser executado, não se preocupe, eu estou lidando com isso, como que a gente faz isso no Kotlin? A gente vai lá e coloca "!!", por que a gente coloca "!!"?

[09:31] Porque, esse tipo de chamada, é muito arriscada, é uma chamada que a gente pode tomar Null Pointer Exception a qualquer momento, porque a gente não tem a garantia de que esse valor já não é nulo. A gente não tem mais essa garantia, por isso, a gente coloca "!!", porque a gente está se responsabilizando, mas é uma coisa muito arriscada, toda vez que você ver esse tipo de código, tome muito cuidado, a gente vai ver abordagens melhores de lidar com esse tipo de situação, mas vamos ver como funciona.

[09:57] Então, a gente está se responsabilizando por isso e, vamos ver, agora, os outros pontos, vamos, também, nos responsabilizar, só para a gente ver o que acontece, "!!", "!!". Reparem que agora, a nossa "ResumoView" voltou a compilar, por mais que ela receba um valor que pode ser nulo, a gente está falando que estamos nos responsabilizando por isso, que isso não vai acontecer.

[10:23] Agora que a "ResumoView" está compilando e a lista "ListaTransacoesActivity" também, vamos executar o nosso código e ver o que acontece, Alt+Shift+F10, vamos lá, vamos pegar o emulador, vamos esperar agora o Android Studio. Reparem que ele conseguiu executar, vamos verificar aqui na nossa App, ela conseguiu executar sem nenhum problema, então, agora, vamos tentar adicionar uma transação só para ver se, realmente, está funcionando.

[10:48] "Adiciona", agora eu vou alterar, aqui, só para testar também, reparem que está tudo funcionando, por mais que o valor pode ser nulo, a gente conseguiu com que o nosso código funcionasse numa boa, porque a gente está garantido que essa inicialização está sendo feita. Só que agora, vamos fazer uma brincadeira bem engraçada, vamos tirar essa inicialização, vou até comentar, vamos tirar essa inicialização e vamos ver o que acontece com o nosso código.

[11:16] Agora, Alt+Shift+F10, reparem que o Android Studio conseguiu executar, só que, olha o que aconteceu com a nossa App, quebrou, se a gente olhar novamente o Logcat, olha onde ele quebrou, olha exatamente o ponto que foi. Ele diz aqui que a gente tomou um Kotlin Null Pointer Exception, não é só um Null Pointer Exception que a gente vê no Java, é um Kotlin Null Pointer Exception, se a gente clicar na última parte onde começou o estouro disso, foi justamente onde?

[11:43] Justamente na primeira chamada que teve aquele "!!", ou seja, a gente se responsabilizou, só que a gente não garantiu, de fato, que o valor não era nulo, então, o próprio Kotlin, falou, você me falou que você se responsabilizou, mas isso causou Null Pointer Exception, por isso foi um Kotlin Null Pointer Exception, porque veio a partir de uma permissão, que o próprio Kotlin nos deu.

[12:07] Por mais que a gente tenha permitido que o nosso código executasse, ele compilasse, a gente percebe que essa abordagem não é tão segura assim para a gente, programador, porque as vezes pode estar funcionando, porque a gente lembrou de inicializar, mas existe os casos em que a gente não lembra, nos casos em que a gente não colocou valor e a gente vai estar lidando com valor nulo. Portanto, esse tipo de abordagem não é recomendado em nenhum momento aqui no Kotlin.

[12:29] A gente vai ver, logo mais, como a gente pode lidar com esse tipo de situação, para garantir que o nosso código não quebre, de um maneira estranha, como a gente viu agora, ele quebrou do nada, simplesmente porque a gente esqueceu de inicializar. Ele compilou, executou tudo, mas só por a gente não ter inicializado a variável, com o valor esperado, a gente tomou Null Pointer Exception, que é uma abordagem não tão legal. A gente,agora vai ver, como a gente pode lidar com esse tipo de situação, logo mais a gente vai ver isso.