02

Criando matcher personalizado

> Caso você precise do projeto com todas as alterações realizadas na aula passada, [você pode baixá-lo neste link (https://github.com/alura-cursos/android-teste-espresso/archive/aula-5.zip)](https://github.com/alura-cursos/android-teste-espresso/archive/aula-5.zip).

Faça com que seja possível distinguir as views dos leilões criando um matcher personalizado que vai utilizar a posição do leilão no Adapter.

Para isso é necessário implementar um membro que implemente a interface `Matcher<? super View>`. Na aula utilizamos o `BoundedMatcher<T, S extend T>` (https://developer.android.com/reference/android/support/test/espresso/matcher/BoundedMatcher), porém, pode ser implementado com outras classes, como o `TypeSafeMatcher` (http://hamcrest.org/JavaHamcrest/javadoc/1.3/org/hamcrest/TypeSafeMatcher.html) ou o `BaseMatcher` (http://hamcrest.org/JavaHamcrest/javadoc/1.3/org/hamcrest/BaseMatcher.html).

A grande diferença é que o `BoundedMatcher` impede a referência nula e já realiza o cast automaticamente de acordo com a classe enviada via argumento de sua assinatura.

Para esse matcher personalizado, considere o ViewMatcher do `onView` sobre o RecyclerView, para utilizá-lo dentro do matcher personalizado e realizar os passos necessários para fazer o matcher das views internas do `ViewHolder`.

> Lembre-se que é uma implementação mais complexa, pois, internamente, ela vai garantir o conteúdo tanto da descrição quanto do maior lance do leilão.

Por fim, comente todo o código que faz matcher apenas com os matchers do Espresso e rode o teste, confira se ele passa.