ESPRESSO TESTING

INTENTS, CUSTOM MATCHER

ESPRESSO INTENTS

- Android Intent is used to open new activity, either internal (opening a product detail screen from product list screen) or external (like opening a dialer to make a call). Internal intent activity is handled transparently by the espresso testing framework and it does not need any specific work from the user side.
- However, invoking external activity is really a challenge because it goes out of our scope, the application under test.
- Once the user invokes an external application and goes out of the application under test, then the chances of user coming back to the application with predefined sequence of action is rather less.

two options

- Espresso provides two options to handle this situation. They are as follows,
- intended
  - This allows the user to make sure the correct intent is opened from the application under test.
- intending
  - This allows the user to mock an external activity like take a photo from the camera, dialing a number from the contact list, etc., and return to the application with predefined set of values (like predefined image from the camera instead of actual image).

Setup

- Espresso supports the intent option through a plugin library and the library needs to be configured in the application's gradle file. The configuration option is as follows,

```
        dependencies { // ...
        androidTestImplementation 'androidx.test.espresso:espresso-
intents:3.1.1' }
```

intended()

- Espresso intent plugin provides special matchers to check whether the invoked intent is the expected intent. The provided matchers and the purpose of the matchers are as follows,
  - hasAction
    - This accepts the intent action and returns a matcher, which matches the specified intent.
  - hasData
    - This accepts the data and returns a matcher, which matches the data provided to the intent while invoking it.
  - toPackage
    - This accepts the intent package name and returns a matcher, which matches the package name of the invoked intent.

intending()

- Espresso provides a special method – *intending()* to mock an external intent action. *intending()* accept the package name of the intent to be mocked and provides a method *respondWith* to set how the mocked intent needs to be responded with as specified below,
  - intending(toPackage("com.android.contacts")).respondWith(result);

Custom View Matchers

- Espresso provides various options to create our own custom view matchers and it is based on *Hamcrest* matchers. Custom matcher is a very powerful concept to extend the framework and also to customize the framework to our taste. Some of the advantages of writing custom matchers are as follows,
  - To exploit the unique feature of our own custom views
  - Custom matcher helps in the *AdapterView* based test cases to match with the different type of underlying data.
  - To simplify the current matchers by combining features of multiple matcher

- We can create new matcher as and when the demand arises and it is quite easy. Let us create a new custom matcher, which returns a matcher to test both id and text of a *TextView.*
- Espresso provides the following two classes to write new matchers –
  - TypeSafeMatcher
  - BoundedMatcher
- Both classes are similar in nature except that the *BoundedMatcher* transparently handles the casting of the object to correct type without manually checking for the correct type. We will create a new matcher, *withIdAndText* using *BoundedMatcher* class. Let us check the steps to write new matchers.
  - Add the below dependency in the *app/build.gradle* file and sync it.

```
dependencies { implementation 'androidx.test.espresso:espresso-core:3.1.1' }
```

Matcher

- fun matchesSafely(val fruit: Fruit) : Boolean – matcher’s logic,
- fun describeTo(val description : Description) : Unit – description of the expected result,
- fun describeMismatchSafely(val item : Fruit , val mismatchDescription : Description) – description of actual value.