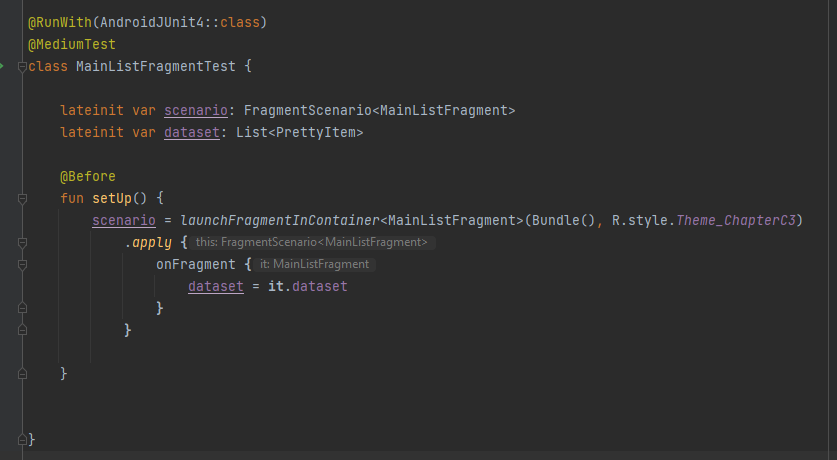
# D.3 – Mockito e End-To-End test

Abbiamo convertito l’activity in un contenitore di fragment, e abbiamo spostato le funzionalità in MainFragment. Ora i nostri test non funzioneranno più, quindi aggiorniamoli

## Testare i Fragment

Similmente all’activity, andiamo a creare uno scenario di test per il Fragment.   
Dopo aver creato la classe di test, utilizza la funzione *launchFragmentInContainer* per creare un’activity temporanea che conterrà il nostro Fragment



Poi portiamo da MainActivityTest a MainFragmentTest le funzioni di test che avevamo scritto

Text

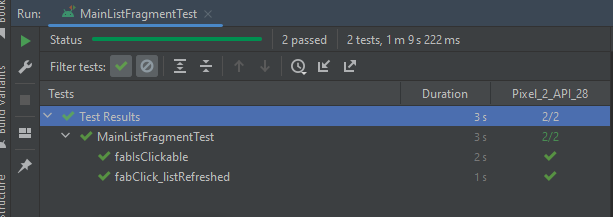
Description automatically generated

Rimuoviamo *dataset* da MainActivityTest, non è piu’ presente e quindi genera errore

Text

Description automatically generated

Ora i nostri test funzionano sul fragment



## Mock e Mockito

Come abbiamo visto i test si occupano di verificare che il codice che scriviamo abbia il comportamento corretto. Gli integration test pero’ spesso richiedono di verificare che il codice che scriviamo produca una interazione con Android corretta. Per questo utilizzeremo un Mock, tramite la libreria **Mockito**

Un test double sono oggetti che sembrano e agiscono come componenti della tua app, ma vengono creati nel tuo test per fornire un comportamento o dati specifici.

Un mock è un tipo di test double che ha aspettative sulle sue interazioni. È un oggetto utilizzato per verificare se vengono applicate le tecniche e i percorsi corretti, ma senza una reale implementazione di metodi o valori.

Testiamo la funzione di navigazione

Testeremo la capacità del nostro codice di inviare al Navigation Component la destinazione corretta con i dati corretti, quando la riga viene cliccata.

Crea un nuovo test dove imposterai un navController *mock* al fragment

Text

Description automatically generated

Ora utilizza Espresso con le RecyclerViewActions per cliccare sull’oggetto della lista che abbia come testo “ID: itemId”

Text

Description automatically generated

Infine, utilizza la funzione *verify* di Mockito per verificare che il navcontroller abbia ricevuto il corretto input di navigazione che ci aspettiamo da questa interazione

Text

Description automatically generated

*Verify* ritorna l’oggetto mock che stiamo analizzando (navController), del quale possiamo utilizzare tutte le funzioni pubbliche, come *navigate*.

Con questa definizione stiamo chiedendo a mockito di **verificare** che **navController** abbia ricevuto l’input di **navigate** a una specifica **direction.**

Questo ci basta come condizione di test, perché il nostro test non deve validare Android, ma l’interazione della nostra app con i componenti Android.

Il test completo:

****

## E2E Testing

Mentre gli Integration Test validano una funzionalità, gli E2E test validano un flusso composto da una serie di funzionalità.

Gli End-To-End test validano una serie di funzionalità che lavorano insieme.

Testano grandi porzioni dell'app, simulano da vicino l'utilizzo reale e quindi sono generalmente lenti. Hanno la massima fedeltà e validano che l’applicazione funzioni effettivamente nel suo insieme.

Creiamo un nuovo test dove verifichiamo che i dati mostrati nella pagina di dettaglio siano quelli della riga cliccata

Effettueremo questo test nella MainActivity, per farlo dovremo indicare a Espresso cosa cliccare. Ora la main activity non ha piu’ un riferimento a dataset, quindi non ci è possibile identificare un elemento della lista da cliccare per il test.

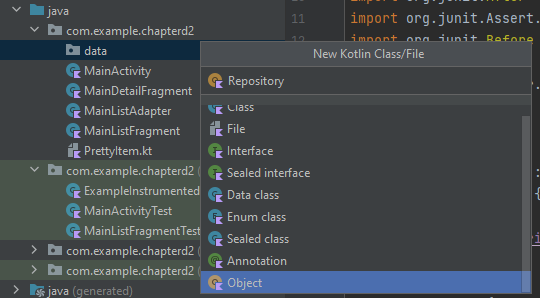
Per rimediare, facciamo in modo che sia MainActivity che MainListFragment possano accedere allo stesso *dataset*

Crea un nuovo package chiamato *data*

*A screenshot of a computer

Description automatically generated with medium confidence*

Al suo interno, crea un nuovo *object* chiamato **Repository**



Al suo interno:

* Sposta da MainListFragment il valore *dataset* e la funzione *generatePrettyItems*
* Aggiungi una funzione per rigenerare il dataset

Text

Description automatically generated

Ora *dataset* e le funzioni relative sono disponibili a chiunque implementerà *Repository*

Modifica MainListFragment in modo che possa ottenere il dataset e la rigenerazione della lista da Repository

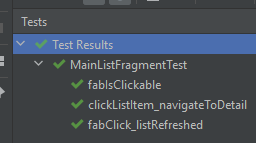
Text

Description automatically generated

Graphical user interface, text

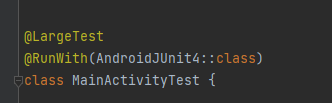
Description automatically generated

Lancia l’app e i test, per controllare che funzioni tutto correttamente.

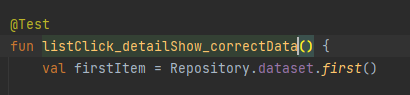


Passiamo ora al test E2E

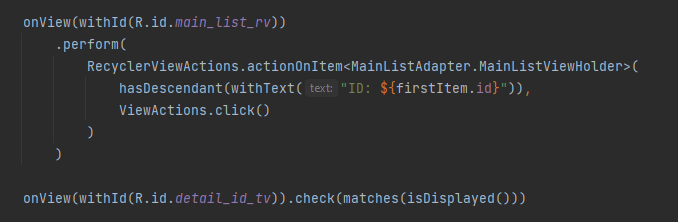
Modifica MainActivityTest con @LargeTest per notificare che queste operazioni possono prendere molto tempo. Questa annotazione è di solito usata solo per gli E2E tests.



Crea un nuovo test e prendi il primo oggetto della lista dal Repository



Effettua il click, poi controlla che la pagina di dettaglio venga visualizzata, utilizzando la textView dell’id come check



Infine controlla che le view abbiano i corretti dati inseriti



## Esercizio: Test Card collection

Testa Card Collection ora che è stata convertita a Fragments, assicurati di effettuare tutti i seguenti test:

* Unit Tests
* Integration Tests
* Navigation Test
* E2E tests