

# برنامه نویسی دستگاه های سیار (CE364)

جلسه هجدهم:  
ذخیره تنظیمات برنامه

**سجاد شیرعلی شمرضا**

**پاییز 1401**

**شنبه، 10 دی 1401**

● بخش مرتبط با این جلسه:

- Unit 5: Data persistence:
  - Pathway 2: Use Room for data persistence



سوال؟

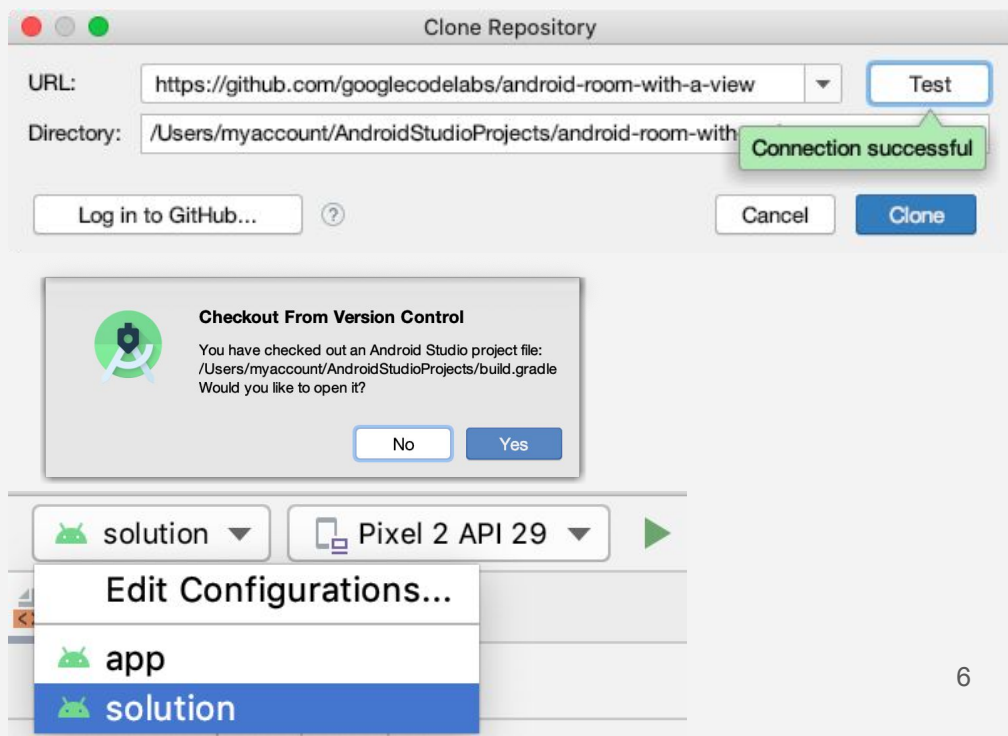
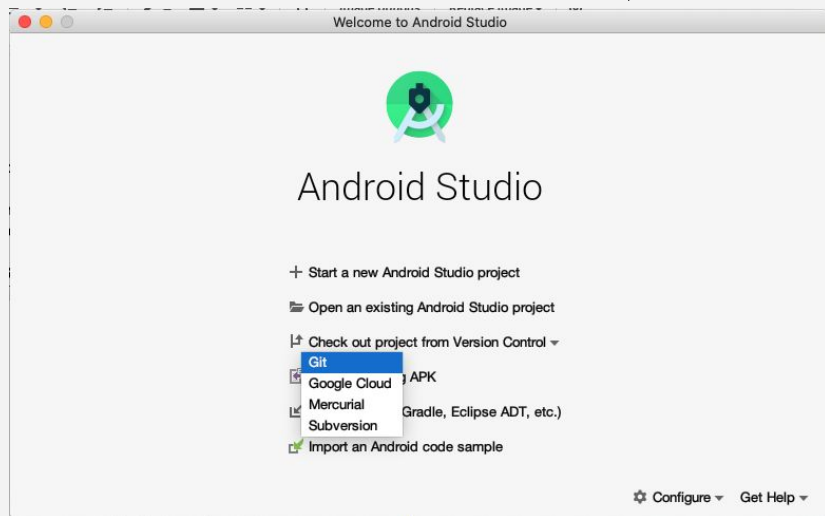
داده خانه

## هدف

- آشنایی با کتابخانه DataStore (داده خانه) از مجموعه جت پکی
- آشنایی با امکان داده خانه اولویت ها (Preferences DataStore)
  - ذخیره مجموعه ای کلید-مقدارها

# کار با انبارهای کنترل نسخه

- امکان دریافت پروژه از انبار کنترل نسخه و مدیریت مستقیم پروژه در آن ها




# یادآوری برنامه لغت نامه

- نمایش لیست حروف و امکان انتخاب یک حرف
- نمایش کلمات مختلفی که با آن حرف شروع میشوند
- تغییر نحوه نمایش حروف
- اضافه کردن امکان ذخیره نحوه نمایش حروف

Words
A
B
C
D
E
F
G
H
I
J
K
L
M

< Words That Start With A
ABOUT
ACNE
ALPHABET
ANCHOR
ANECDOTE

Words



A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P
Q	R	S	T
U	V	W	X
Y	Z		

# کاربرد Preferences DataStore

- مناسب برای نگه داری پایگاه های داده ساده
  - مانند: اطلاعات ورود به سامانه، تنظیمات برنامه
- برای پایگاه داده های پیچیده تر: استفاده از Room
  - مثلاً اطلاعات دانشجویان، موجودی انبار
- امکان تعریف یک رابط برنامه نویسی (API) ساده، ایمن، و غیر همزمان برای ذخیره سازی داده
- اضافه کردن پیش نیاز به برنامه (در فایل build.gradle)

```
implementation "androidx.datastore:datastore-preferences:1.0.0"  
implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.3.1"
```



## دو روش ذخیره اطلاعات

- بدون تعریف ساختار اولیه (schema)
  - شناخته شده به نام داده خانه (Preferences DataStore)
  - ذخیره بر اساس نام کلید
- بر اساس داده دارای نوع قوی
  - شناخته شده به نام Proto DataStore
  - واحد ذخیره سازی پروتو است (Protocol buffers و یا proto)

# تعریف کلاس داده خانه

- تعریف یک کلاس جدید به نام `SettingsDataStore` در داخل بسته `data`
- گرفتن یک پارامتر از نوع `Context` در سازنده آن
- تعریف یک نام برای داده خانه و شیء داده خانه خارج (قبل) از کلاس در فایل
  - استفاده از نوع اولویت (`Preferences`) برای نوع داده خانه

```
private const val LAYOUT_PREFERENCES_NAME = "layout_preferences"
```

```
// Create a DataStore instance using the preferencesDataStore delegate, with the Context  
// receiver.
```

```
private val Context.dataStore : DataStore<Preferences> by preferencesDataStore(  
    name = LAYOUT_PREFERENCES_NAME  
)
```

## تعریف کلید ها

- تعریف نوع کلید ها (مثلا عدد صحیح یا رشته)
  - استفاده از توابعی مانند `intPreferencesKey` و یا `stringPreferencesKey`
- مثال: تعریف یک مقدار درست/غلط برای تعیین استفاده از مدیر نمای خطی
  - تعریف نوع و نام کلید مورد نظر

```
private val IS_LINEAR_LAYOUT_MANAGER = booleanPreferencesKey("is_linear_layout_manager")
```

# ذخیره اطلاعات جدید در داده خانه

- استفاده از تابع edit داده خانه
  - اجرای یک مجموعه از دستورات
  - اجرای توسط Dispatcher.IO

```
suspend fun saveLayoutToPreferencesStore(isLinearLayoutManager: Boolean, context: Context) {  
    context.dataStore.edit { preferences ->  
        preferences[IS_LINEAR_LAYOUT_MANAGER] = isLinearLayoutManager  
    }  
}
```

# خواندن مقدار از داده خانه

- فراهم آوردن کل اطلاعات ذخیره شده در قابل `Flow<Preferences>`
- فقط در دسترس قرار دادن اطلاعات مورد نظر (و نه کل داده خانه)
  - رسیدگی کردن به استثناءها

```
val preferenceFlow: Flow<Boolean> = context.dataStore.data
    .catch {
        if (it is IOException) {
            it.printStackTrace()
            emit(emptyPreferences())
        } else {
            throw it
        }
    }
    .map { preferences ->
        // On the first run of the app, we will use LinearLayoutManager by default
        preferences[IS_LINEAR_LAYOUT_MANAGER] ?: true
    }
```

# ایجاد نمونه از کلاس جدید در برنامه

- تعریف یک نمونه از کلاس دسترسی ایجاد شده در کلاس قطعه LetterListFragment

```
private lateinit var SettingsDataStore: SettingsDataStore
```

- مقداردهی اولیه آن در هنگام ایجاد نما

```
override fun onCreateView(view: View, savedInstanceState: Bundle?) {  
    ...  
    // Initialize SettingsDataStore  
    SettingsDataStore = SettingsDataStore(requireContext())  
}
```

## متصل کردن داده به نما

```
override fun onCreateView(view: View, savedInstanceState: Bundle?) {  
    recyclerView = binding.recyclerView  
    // Initialize SettingsDataStore  
    SettingsDataStore = SettingsDataStore(requireContext())  
    SettingsDataStore.preferenceFlow.asLiveData().observe(viewLifecycleOwner, { value ->  
        isLinearLayoutManager = value  
        chooseLayout()  
    })  
}
```

## ذخیره تغییر در مقدار

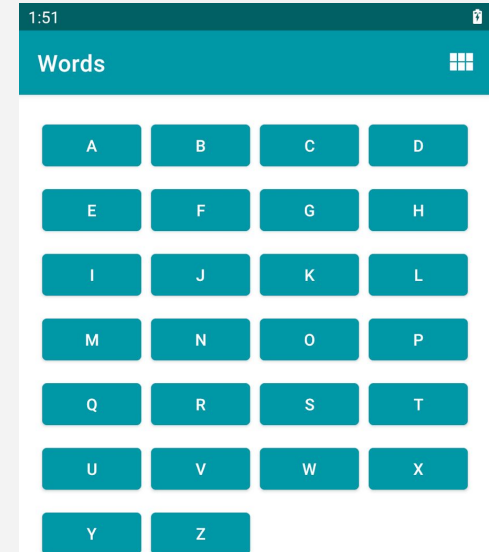
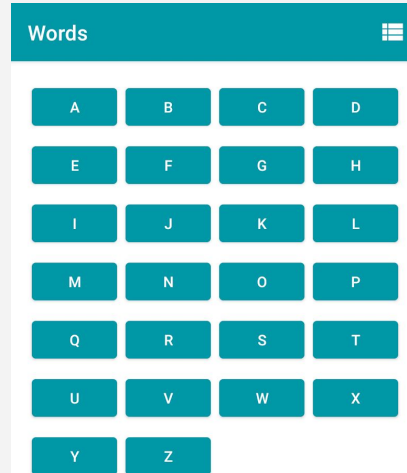
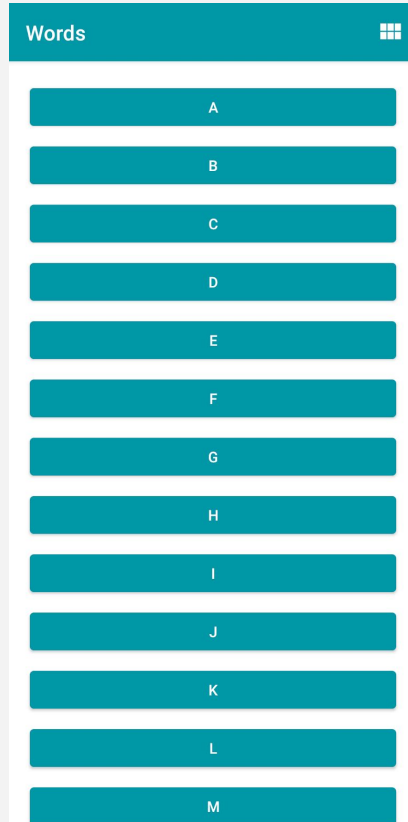
```
override fun onOptionsItemSelected(item: MenuItem): Boolean {  
    return when (item.itemId) {  
        R.id.action_switch_layout -> {  
            ...  
            // Launch a coroutine and write layout setting in preference Datastore  
            lifecycleScope.launch {  
                SettingsDataStore.saveLayoutToPreferencesStore(  
                    isLinearLayoutManager, requireContext())  
            }  
            ...  
            return true  
        }  
    }  
}
```



# آزمایش برنامه

- اجرای برنامه
  - تغییر حالت مدیر نما
  - بستن (خارج شدن) برنامه
  - اجرای دوباره برنامه
- مشاهده استفاده از مدیر نما در حالت قبل از بستن برنامه

# مشکل آیکن منو پس از اجرای مجدد



# درست کردن مشکل آیکن منو پس از اجرای مجدد

- منو در هر تغییر برنامه دوباره رسم نمی شود
  - رسم دوباره آن با فراخوانی تابع `invalidateOptionsMenu`
- فراخوانی تابع پس از تعیین مقدار (نوع) آیکن

```
override fun onCreateView(view: View, savedInstanceState: Bundle?) {  
    ...  
    SettingsDataStore.preferenceFlow.asLiveData().observe(viewLifecycleOwner, { value ->  
        ...  
        // Redraw the menu  
        activity?.invalidateOptionsMenu()  
    })  
}
```



سوال؟

تست کردن پایگاه داده

- اضافه کردن تست برای ارزیابی درستی و کارایی نحوه استفاده از پایگاه داده
- انواع تست:
  - تست واحد (unit test)
    - تست بخشی از کد
    - اجرا در داخل ماشین مجازی جاوا (JVM) و بر روی کامپیوتر
  - تست ابزاری (instrumentation test)
    - اجرا بر روی دستگاه (گوشی و یا شبیه ساز)

# اجرای تست ابراری

- کمک گرفتن از هماهنگ کننده تست (Test Orchestrator)
  - پاک کردن وضعیت برنامه در بین اجرای دو تست
  - جلوگیری از وابستگی یک تست به تست اجرا شده قبلی
- تعریف شده در فایل app/build.gradle

```
android {  
    ...  
    defaultConfig {  
        ...testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
    }  
}
```

## اضافه کردن پیش نیازها

```
dependencies {  
    ...  
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'  
    androidTestImplementation 'com.android.support.test.espresso:espresso-contrib:3.0.2'  
}
```



# هدف تست شماره 1

- اضافه کردن شاخه androidTest و کلاس DataTransactionTests
- هدف تست شماره 1 (برای برنامه انبارداری)
  - اجرای فعالیت اصلی (ActivityScenarioRule)
  - کلیک کردن بر روی دکمه اضافه کردن عنصر
  - وارد کردن متن برای بخش های لازم
  - کلیک کردن بر روی دکمه ذخیره
  - تایید اینکه برنامه عنصر جدید را اضافه کرده و اطلاعات به درستی در صفحه نمایش داده میشود

# تست شماره 1

```
@RunWith(AndroidJUnit4::class)
class DataTransactionTests {

    @get:Rule
    val scenario = ActivityScenarioRule(MainActivity::class.java)


    private val testItemName = "Test Item"
    private val testItemPrice = "5.00"
    private val testItemInitialCount = "1"
    private val quantityHeader = "Quantity\nIn Stock"


    @Test
    fun added_item_displays_in_list() {
        onView(withId(R.id.floatingActionButton)).perform(click())
        onView(withId(R.id.item_name)).perform(typeText(testItemName))
        onView(withId(R.id.item_price)).perform(typeText(testItemPrice))
        onView(withId(R.id.item_count)).perform(typeText(testItemInitialCount))
        onView(withId(R.id.save_action)).perform(click())










        // Make sure we are back in the list fragment by checking that a header is display
        onView(withText(quantityHeader)).check(matches(isDisplayed()))

        // Make sure item is displayed
        onView(withText(testItemName)).check(matches(isDisplayed()))
        onView(withText("$testItemPrice")).check(matches(isDisplayed()))
        onView(withText(testItemInitialCount)).check(matches(isDisplayed()))
    }
}
```

# اجرا کردن تست

Run:  added\_item\_display...() ×

▶ Status  1 passed | 1 tests, 6 s 214 ms

🔧 Filter tests:         

Tests	Duration	Pixel_3a_API_30
✓ Test Results	4 s	1/1
✓ DataTransactionTests	4 s	1/1
✓ added_item_displays_in_list	4 s	✓

# اجرای دوباره تست

Run: `added_item_display...()` 1 failed 1 tests, 15 s 992 ms

Filter tests:

Tests	Duration	Pixel_3a_API_30
Test Results	13 s	0/1
DataTransactionTests	13 s	0/1
added_item_displays_in_list	13 s	

com.example.inventory.DataTransactionTests.added\_item\_displays\_in\_list

androidx.test.espresso.AmbiguousViewMatcherException: 'an instance of android.widget.TextView and view.getText() with or without transformation to match: is "Test Item"' matches multiple views in the hierarchy.  
Failed on Pixel\_3a\_API\_30

Logs    Device Info

```
+----->ActionBarContextView{id=2131230777, res-name=action_context_bar, visibility=GONE, width=0, height=0, has-focus=false, has-focusable=
  at dalvik.system.VMStack.getThreadStackTrace(Native Method)
  at java.lang.Thread.getStackTrace(Thread.java:1736)
  at androidx.test.espresso.base.DefaultFailureHandler.getUserFriendlyError(DefaultFailureHandler.java:12)
  at androidx.test.espresso.base.DefaultFailureHandler.handle(DefaultFailureHandler.java:7)
  at androidx.test.espresso.ViewInteraction.waitForAndHandleInteractionResults(ViewInteraction.java:8)
  at androidx.test.espresso.ViewInteraction.check(ViewInteraction.java:12)
  at com.example.inventory.DataTransactionTests.added_item_displays_in_list(DataTransactionTests.kt:38)
```



# اجرای دوباره تست

Run: added\_item\_display...()

Status: 1 failed, 1 tests, 15 s 992 ms

Filter tests: [Icons]

Tests

Tests	Duration	Pixel_3a_API_30
Test Results	13 s	0/1
DataTransactionTests	13 s	0/1
added_item_displays_in_list	13 s	0/1

com.example.inventory.DataTransactionTests.added\_item\_displays\_in\_list

androidx.test.espresso.AmbiguousViewMatcherException: 'an instance of android.widget.TextView and view.getText() with or without transformation to match: is "Test Item"' matches multiple views in the hierarchy.  
Failed on Pixel\_3a\_API\_30

Logs    Device Info

```
+----->ActionBarContextView{id=2131230777, res-name=action_context_bar, visibility=GONE, width=0, height=0, has-focus=false, has-focusable=false}
  at dalvik.system.VMStack.getThreadStackTrace(Native Method)
  at java.lang.Thread.getStackTrace(Thread.java:1736)
  at androidx.test.espresso.base.DefaultFailureHandler.getUserFriendlyError(DefaultFailureHandler.java:12)
  at androidx.test.espresso.base.DefaultFailureHandler.handle(DefaultFailureHandler.java:7)
  at androidx.test.espresso.ViewInteraction.waitForAndHandleInteractionResults(ViewInteraction.java:8)
  at androidx.test.espresso.ViewInteraction.check(ViewInteraction.java:12)
  at com.example.inventory.DataTransactionTests.added_item_displays_in_list(DataTransactionTests.kt:38)
```

- اجرای برنامه و مشاهده خروجی
- دلیل؟ باقی ماندن اطلاعات از دفعه قبل

## Inventory

ITEM	PRICE	QUANTIT Y IN STOCK
Test Item	\$5.00	1
Test Item	\$5.00	1

# حل مشکل تست

- استفاده از هماهنگی کننده تست در فایل app/build.gradle

```
android {  
    ...  
    testOptions {  
        execution = "ANDROIDX_TEST_ORCHESTRATOR"  
    }  
}
```

- پاک کردن اطلاعات در بین اجراها

```
android {  
    ...  
    defaultConfig {  
        ...testInstrumentationRunnerArguments clearPackageData: "true"  
    }  
}
```

## تست شماره 2 - حذف عنصر

@Test



```
fun list_empty_after_item_deletion() {  
    onView(withId(R.id.floatingActionButton)).perform(click())  
    onView(withId(R.id.item_name)).perform(typeText(testItemName))  
    onView(withId(R.id.item_price)).perform(typeText(testItemPrice))  
    onView(withId(R.id.item_count)).perform(typeText(testItemInitialCount))  
    onView(withId(R.id.save_action)).perform(click())  
  
    // Make sure we are back in the list fragment by checking that a header is displayed  
    onView(withText(quantityHeader)).check(matches(isDisplayed()))  
  
    onView(withId(R.id.recyclerView)).perform(  
        RecyclerViewActions  
            .actionOnItemAtPosition<RecyclerView.ViewHolder>(0, click())  
    )  
    onView(withId(R.id.delete_item)).perform(click())  
    onView(withText("Yes")).perform(click())  
  
    onView(withText(testItemName)).check(doesNotExist())  
}
```







# اجرای هر دو تست پشت سر هم

```
18 @RunWith(AndroidJUnit4::class)
19 class DataTransactionTests {
20
21     @get:Rule
22     val scenario = ActivityScenarioRule(MainActivity::class.java)
23
24     private val testItemName = "Test Item"
25     private val testItemPrice = "5.00"
26     private val testItemInitialCount = "1"
27     private val quantityHeader = "Quantity\nIn Stock"
28
29
30     @Test
31     fun added_item_displays_in_list() {
32         onView(withId(R.id.floatingActionButton)).perform(click())
33         onView(withId(R.id.item_name)).perform(typeText(testItemName))
34         onView(withId(R.id.item_price)).perform(typeText(testItemPrice))
35         onView(withId(R.id.item_count)).perform(typeText(testItemInitialCount))
36         onView(withId(R.id.save_action)).perform(click())
37
38         // Make sure we are back in the list fragment by checking that a header is displayed
39         onView(withText(quantityHeader)).check(matches(isDisplayed()))
40
41         // Make sure item is displayed
42         onView(withText(testItemName)).check(matches(isDisplayed()))
43         onView(withText(text: "$$testItemPrice")).check(matches(isDisplayed()))
44         onView(withText(testItemInitialCount)).check(matches(isDisplayed()))
45     }
```

```
18 @RunWith(AndroidJUnit4::class)
19 class DataTransactionTests {
20     ▶ Run 'DataTransactionTests...' ^⌘F10
21     🐞 Debug 'DataTransactionTests...' ^⌘F9
22     📄 Profile 'DataTransactionTests...'
23     ⚙️ Modify Run Configuration...
24     private val testItemName = "Test Item"
25     private val testItemPrice = "5.00"
26     private val testItemInitialCount = "1"
27     private val quantityHeader = "Quantity\nIn Stock"
28
29     @Test
```

Run: DataTransactionTests x

Status  2 passed 2 tests, 16 s 750 ms 

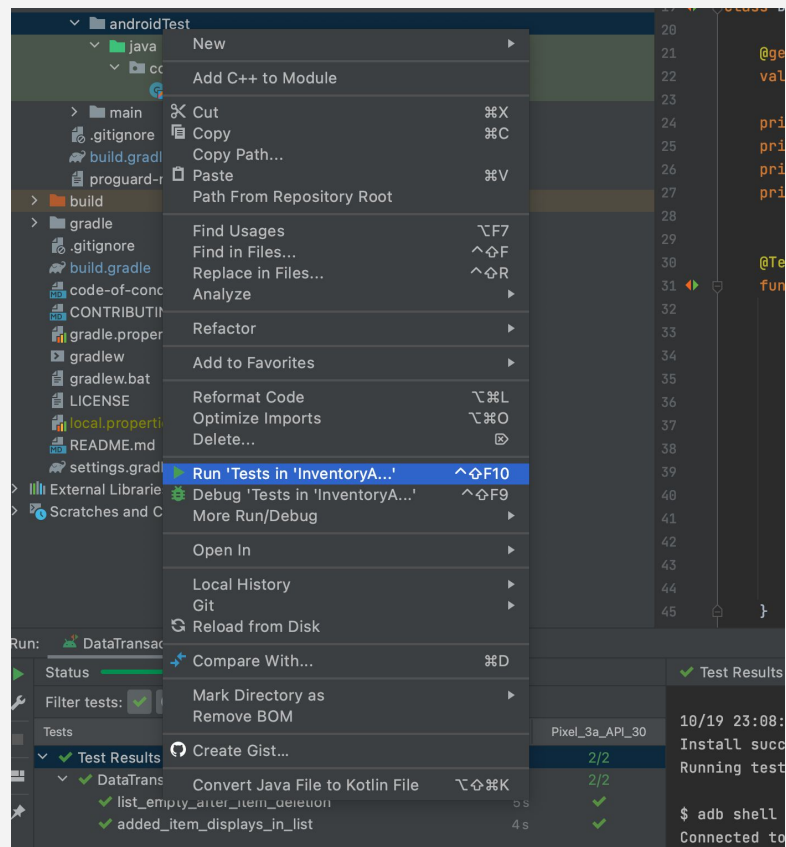
Filter tests:      

Tests	Duration	Pixel_3a_API_30	10/19
✓ Test Results	10 s	2/2	Insta
✓ DataTransactionTests	10 s	2/2	Runni
✓ list_empty_after_item_deletion	5 s	✓	
✓ added_item_displays_in_list	4 s	✓	\$ adb Conne

Tests Passed  
2 passed



# اجرای تمامی تست های یک پروژه





سوال؟