



Effective Android Testing





Somkiat Puisungnoen

Search

Somkiat | Home

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro

Software Craftsmanship

Software Practitioner at สยามชานาญกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ...

Java and Bigdata

This image shows a Facebook profile page for 'Somkiat Puisungnoen'. The top navigation bar includes the Facebook logo, the user's name, a search bar, and profile picture. Below the header, there's a collage of photos featuring the user and others. A prominent photo of the user, wearing a white t-shirt with a Superman logo, is displayed. Below the collage are tabs for Timeline, About, Friends (3,138), Photos, and More. On the left, there's a sidebar with a pending post about working at Opendream and sections for Intro, Software Craftsmanship, and professional experience as a Software Practitioner and Agile Practitioner. On the right, there's a 'What's on your mind?' section with a placeholder image of the user, and a post from the user himself about Java and Bigdata.



somkiat.cc

Page Messages Notifications 3 Insights Publishing Tools Settings Help ▾

somkiat.cc
@somkiat.cc

Home Posts Videos Photos

Liked Following Share ...

+ Add a Button



<https://github.com/up1/course-effective-android-testing>



Agenda

- Introduction of testing
- Why we need to test ?
- Types of tests
- Testing pyramid concept
- Android testing
- Workshop (step-by-step)



Agenda

- UI testing with Espresso and Kakao
- Espresso workshop
- Testable application
- Code coverage



Testing for Android app



**"Program testing can be used
to show the presence of bugs,
but never their absence."**

Edsger Dykstra, 1970, Notes on Structured Programming



Why we need to test ?

Help you to **catch bugs**

Develop features **faster**

Enforce **modularity** of your project



But,
**It's take time to learning and
practice !!**



Goals

How to **THINK** when and where
you should test ?



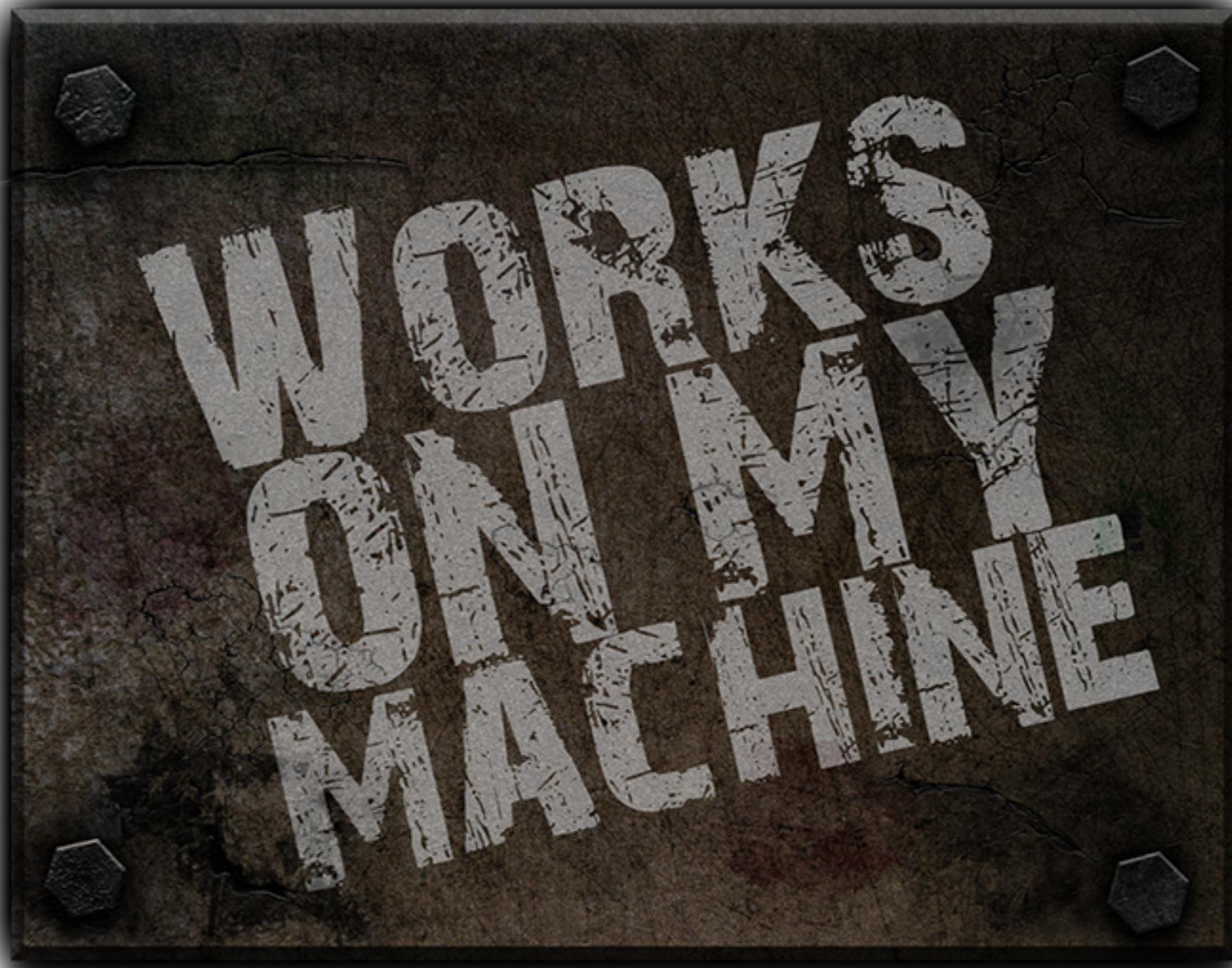
What you need to know ?

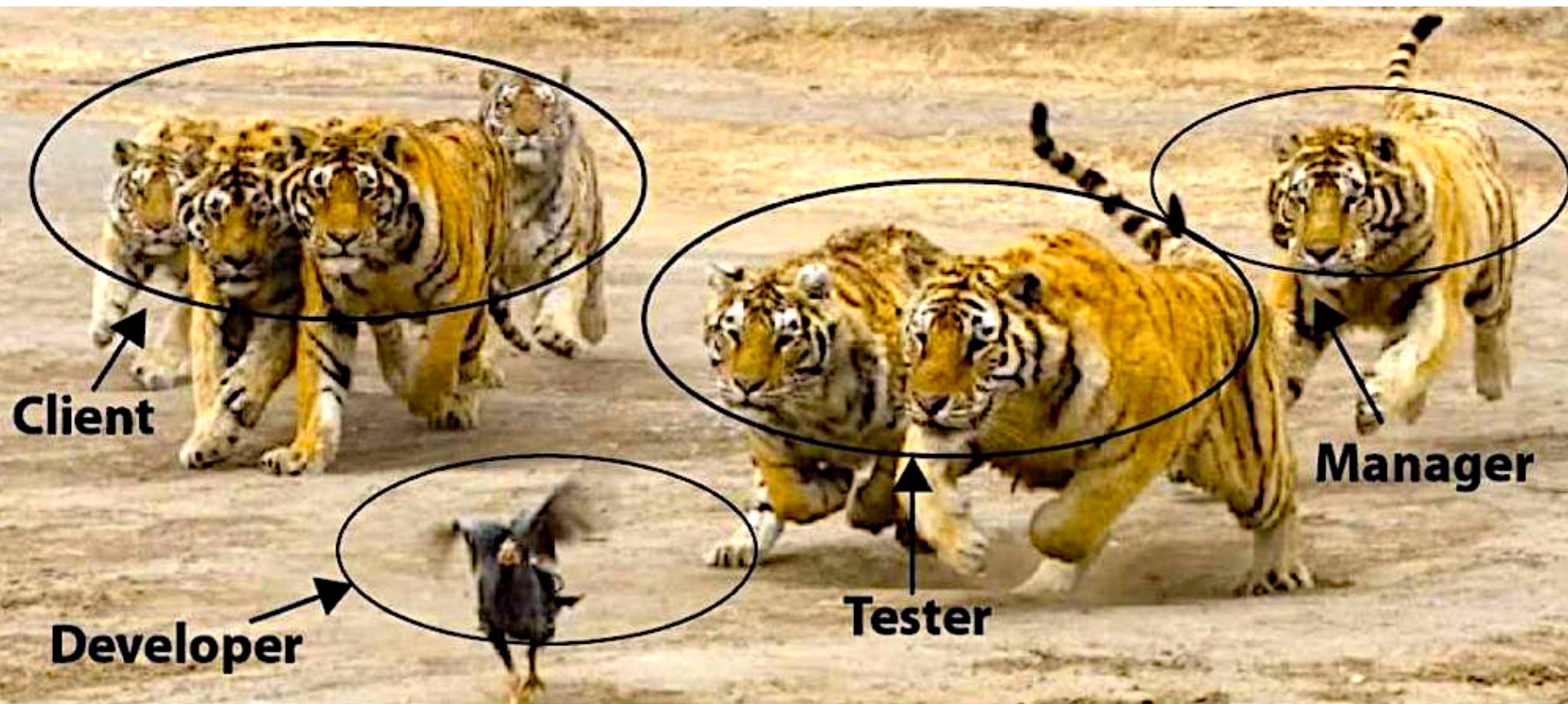
Java/Android/Kotlin
Android Studio
JUnit 4
Espresso/Kakao



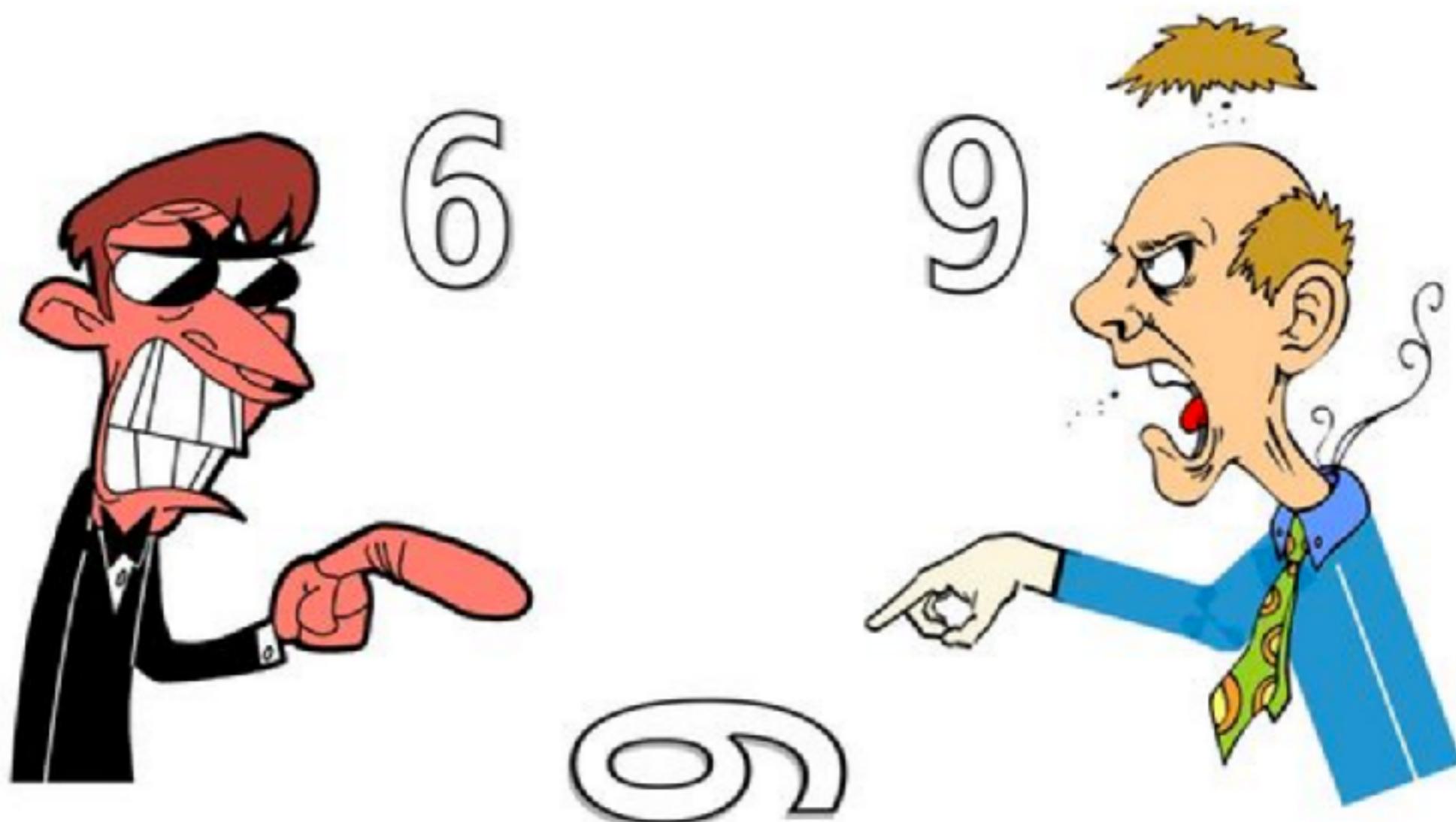
All about Testing







Developer vs Tester

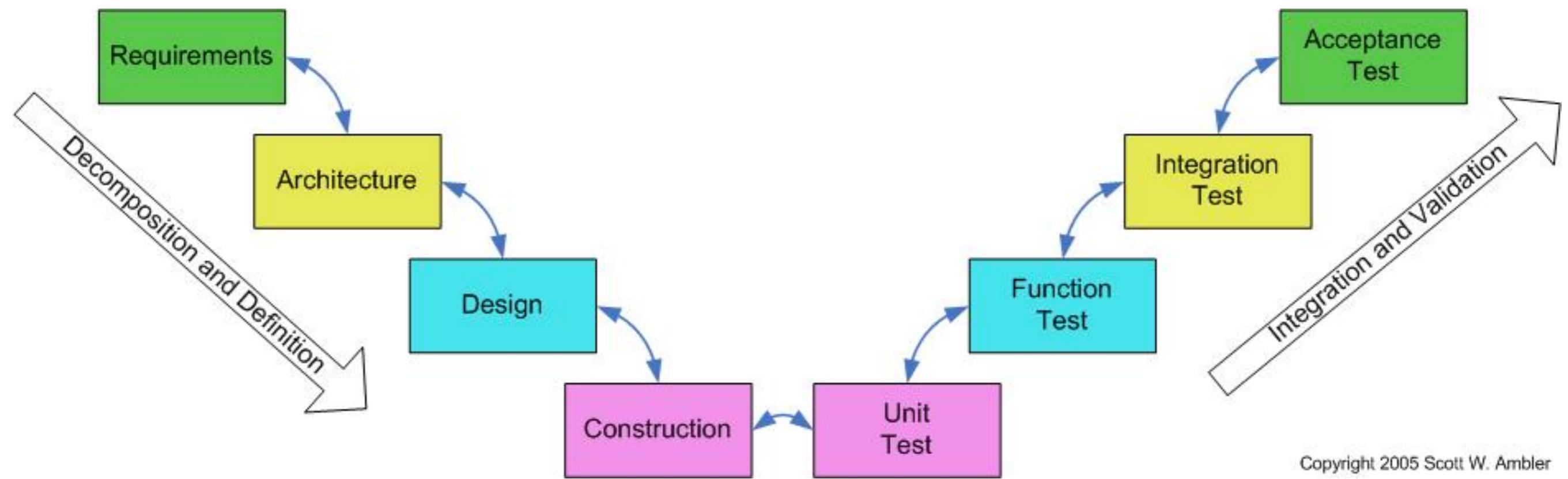


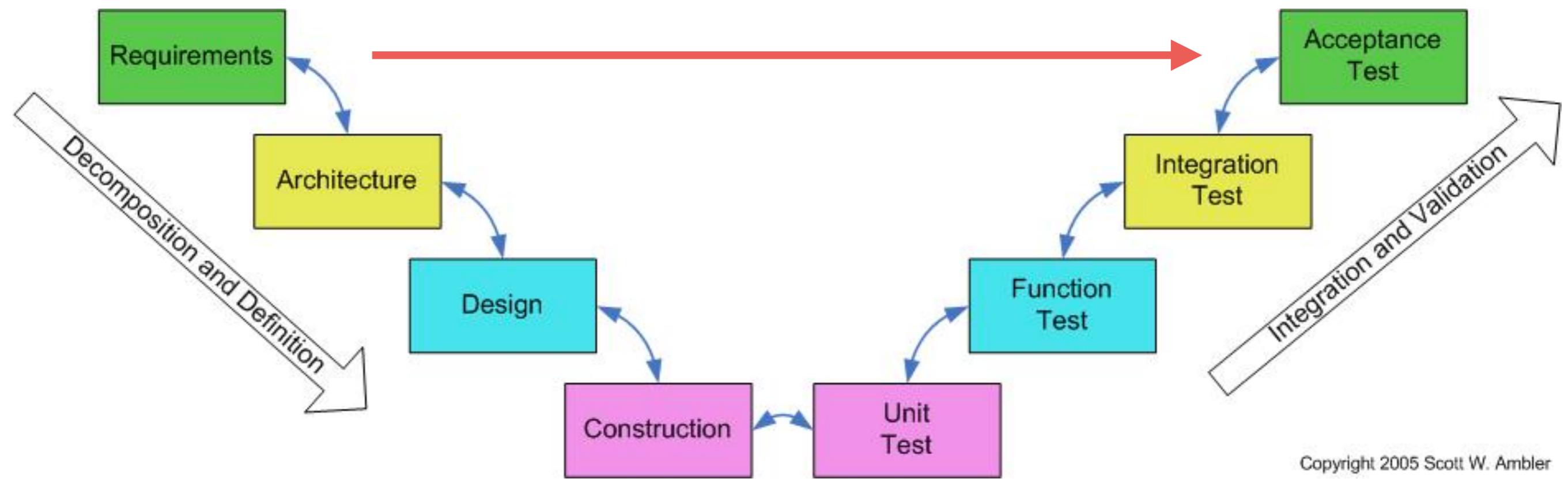


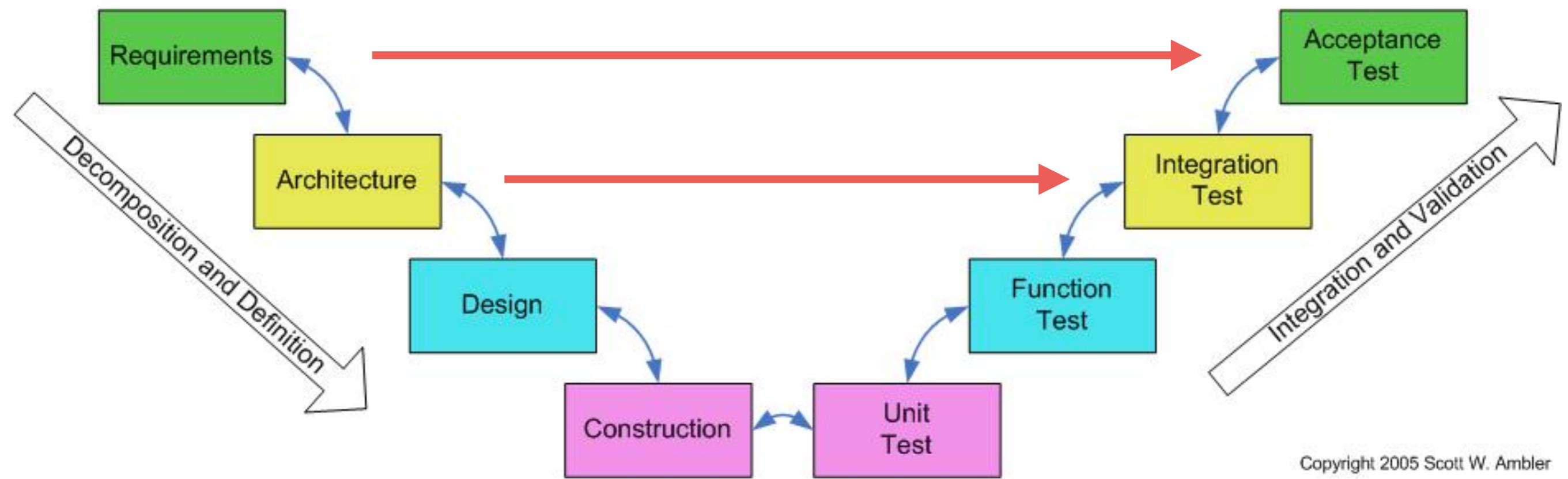
www.cartoonistshilpa.com

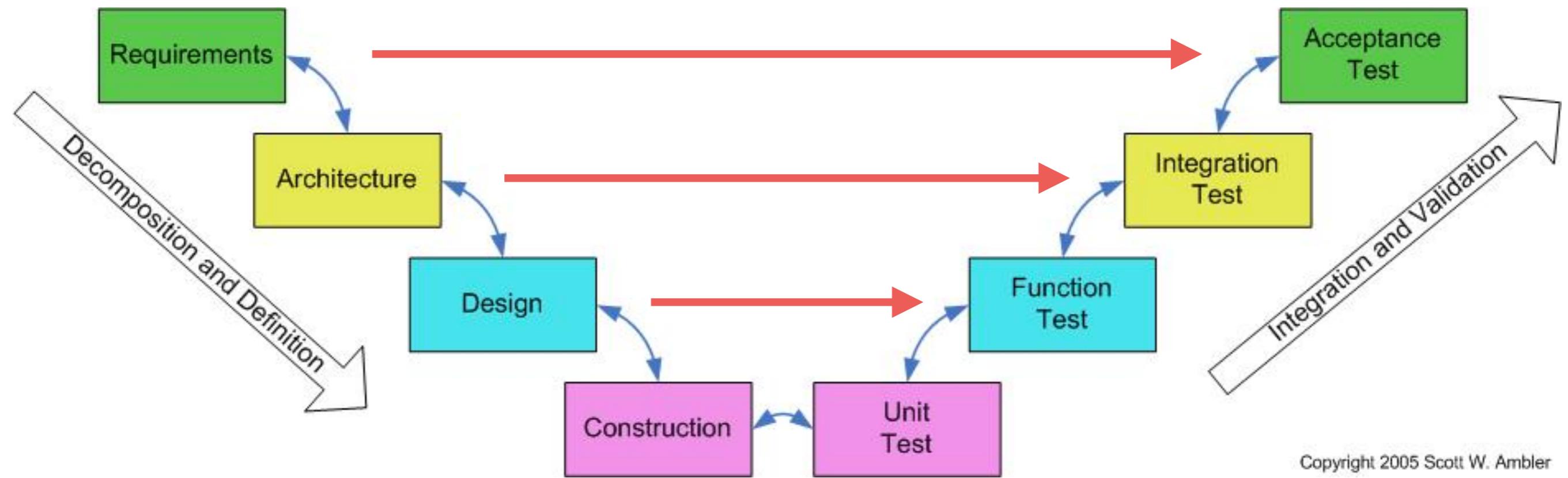


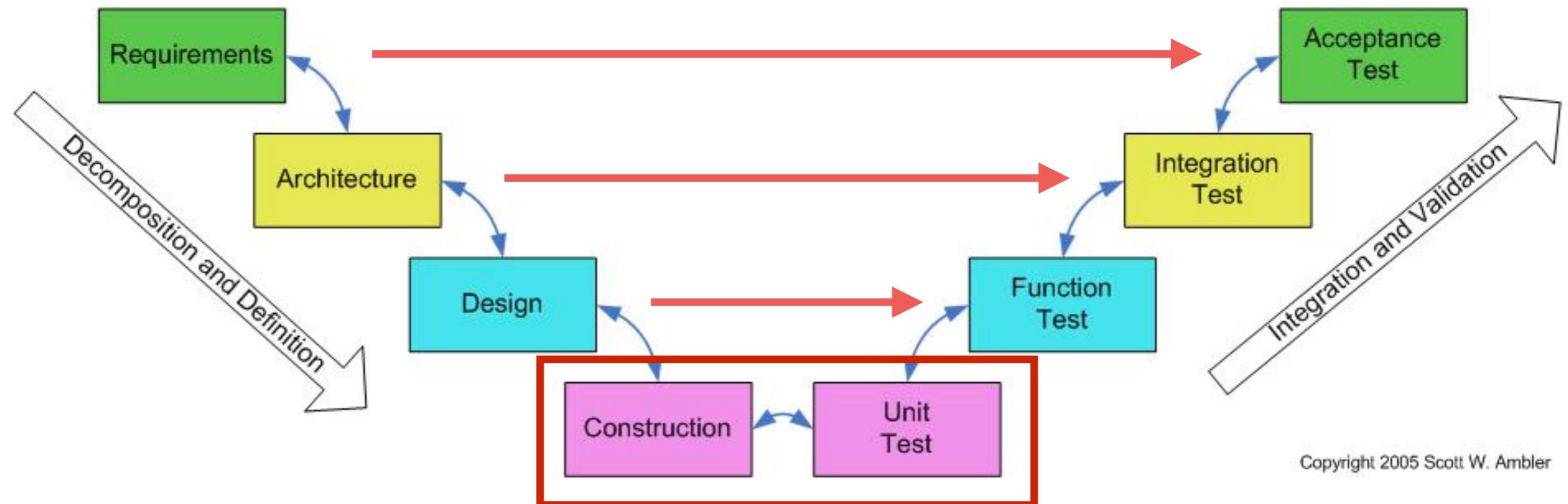








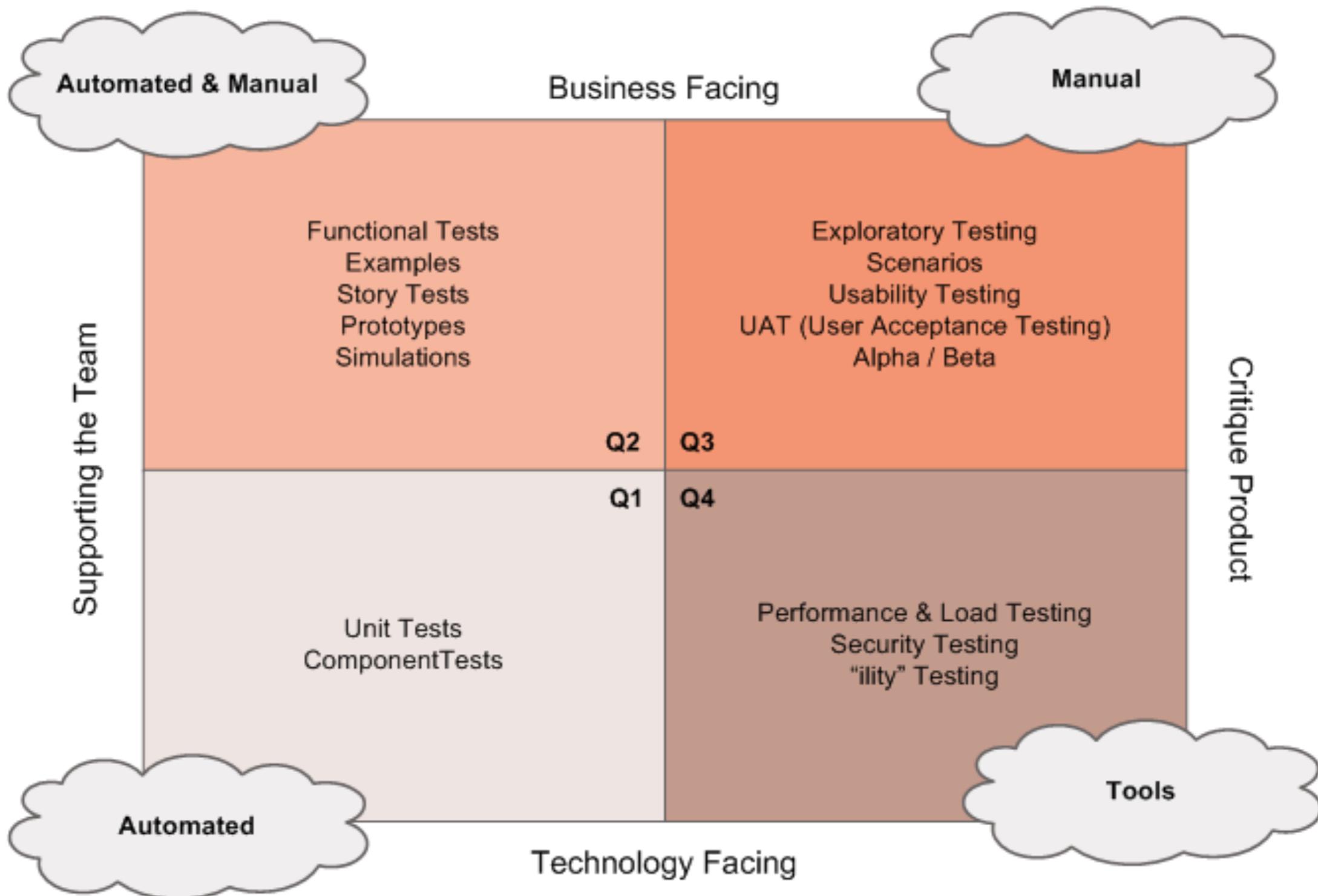




Type of testing



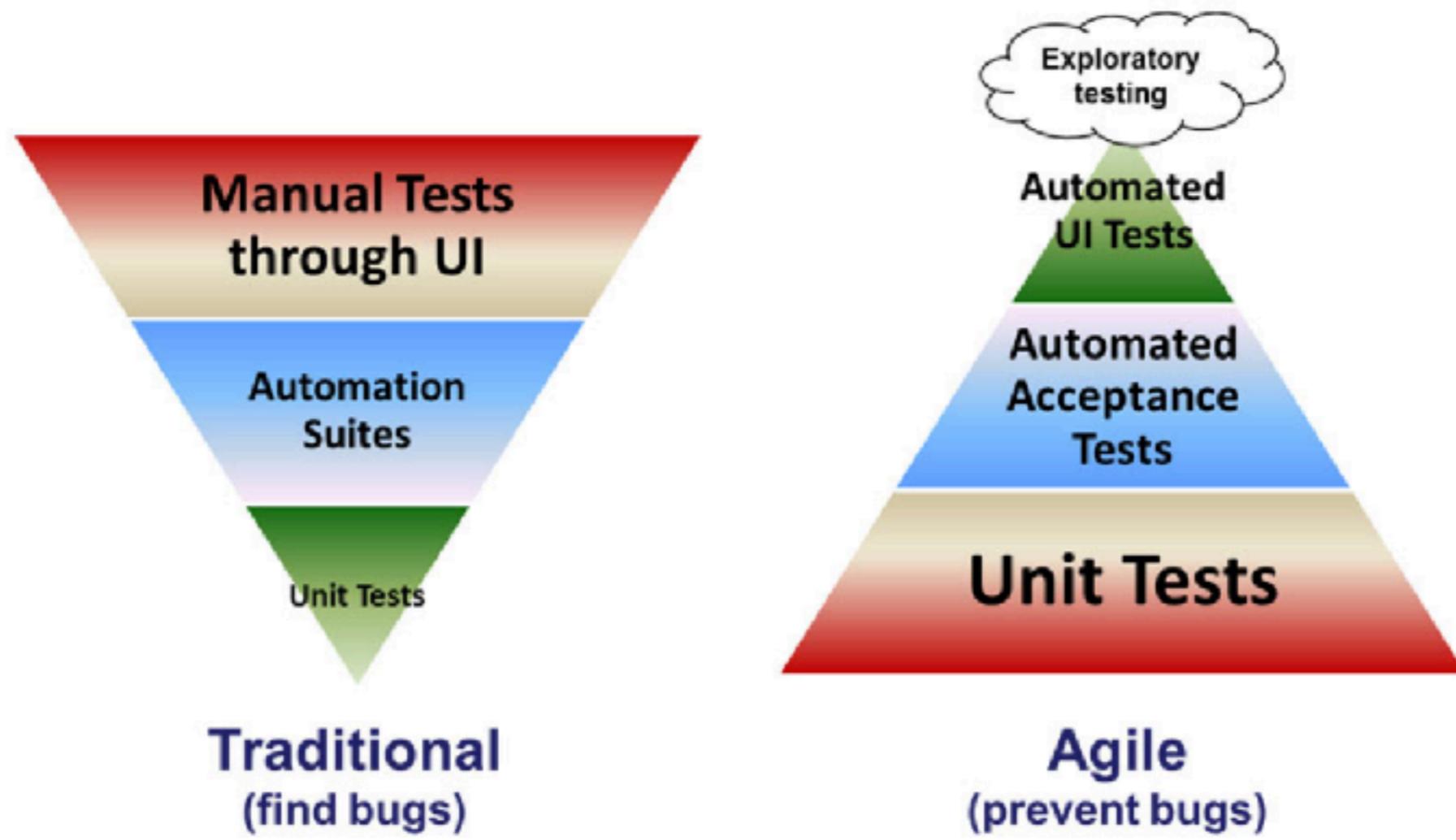
Agile Testing Quadrants



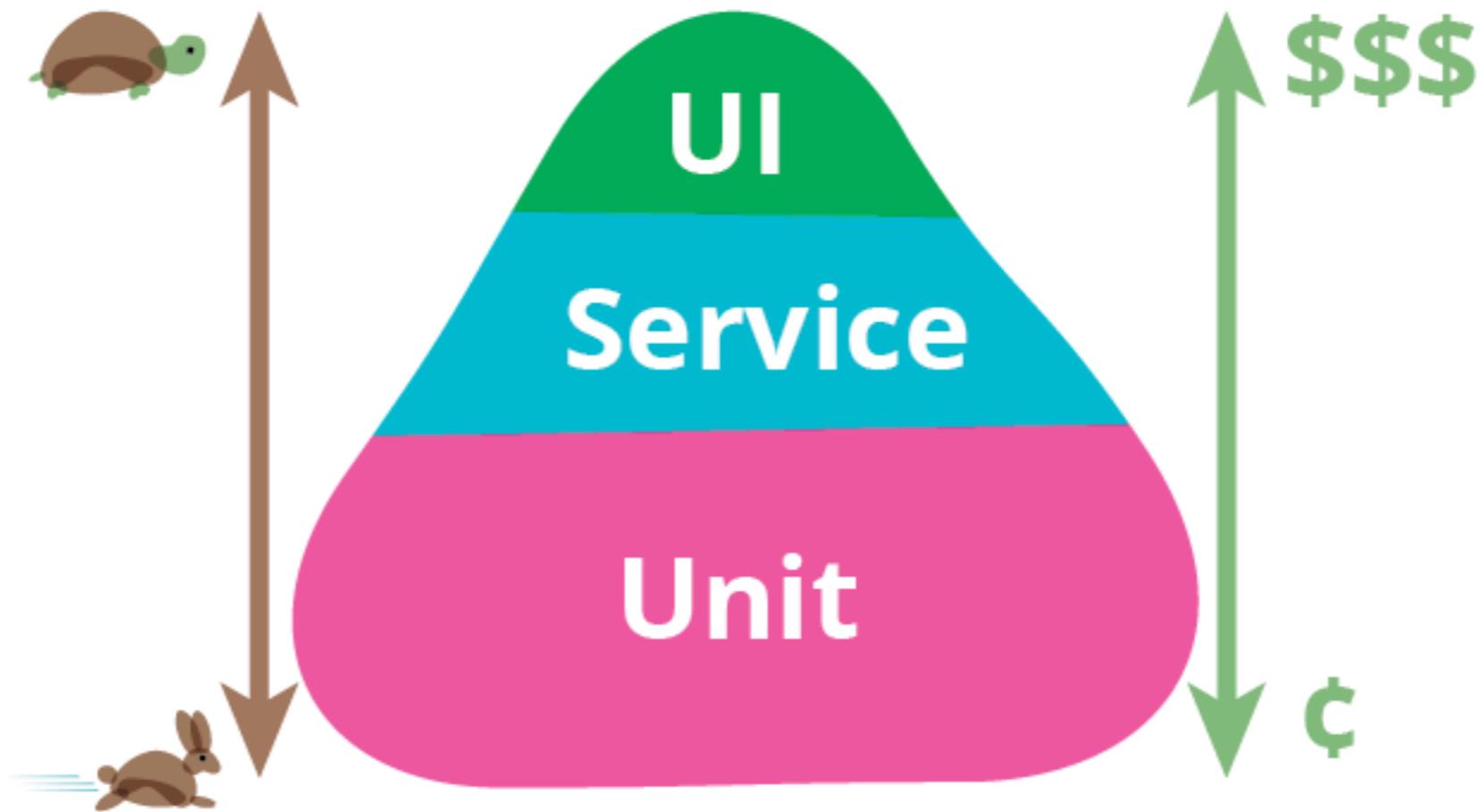
Testing pyramid



Testing Pyramid



Testing Pyramid



<https://martinfowler.com/bliki/TestPyramid.html>



Testing not a phase
Testing is activity



Android Testing

Android



Android Testing

Android

Java



Java run on JVM



JVM (Java Virtual Machine)



Run android need device



Build app -> Install to device -> Test



Android Testing

JVM

Device

/src/test

/src/androidTest



JVM unit test

JVM

Device

JVM unit test

/src/test



Business logic with pure java code



Instrumentation unit test

JVM

Device

JVM unit test

Instrumentation unit
test

/src/test

/src/androidTest

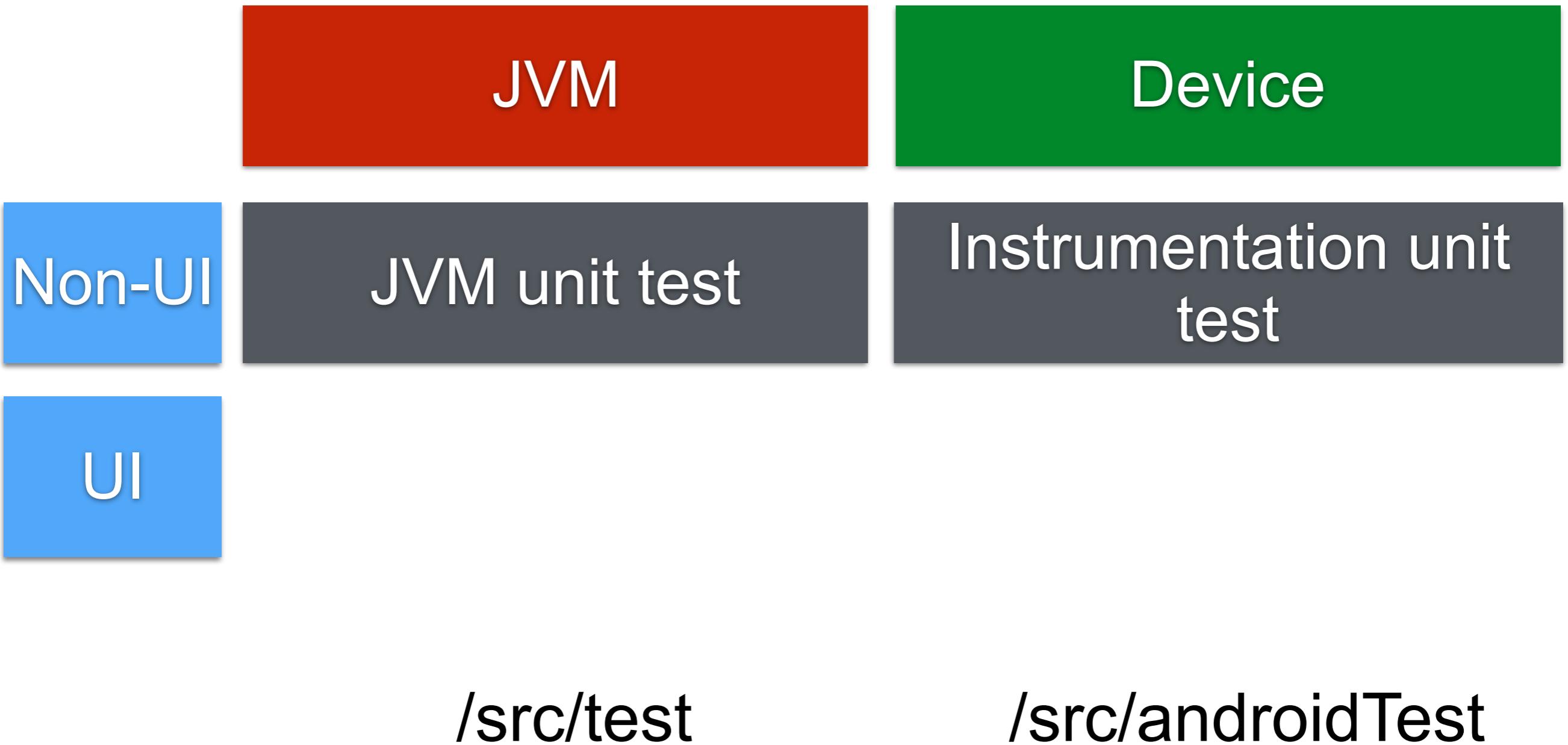
Working with Android specific code, you need run on device such as AssetManager, SharedPreference



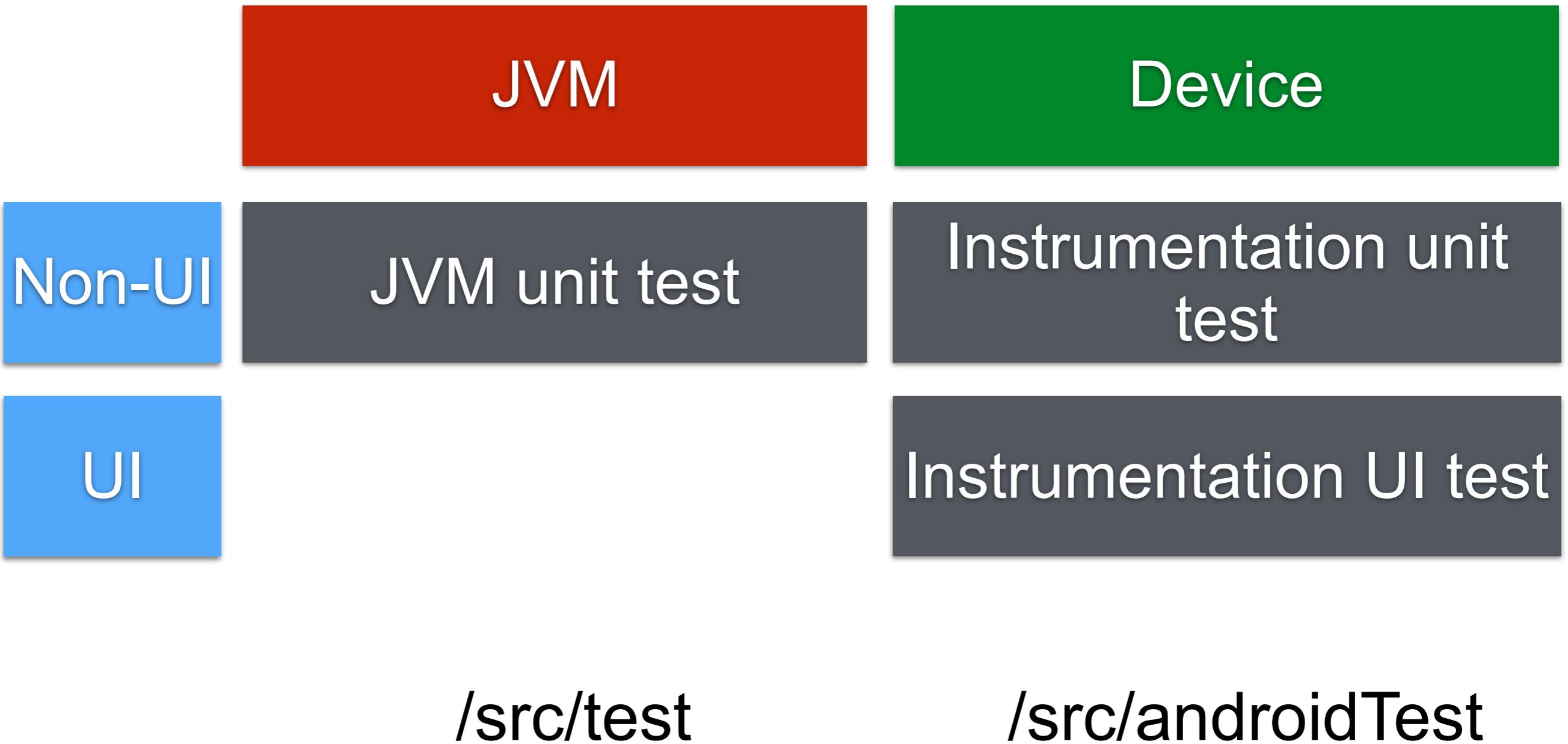
UI vs Non-UI



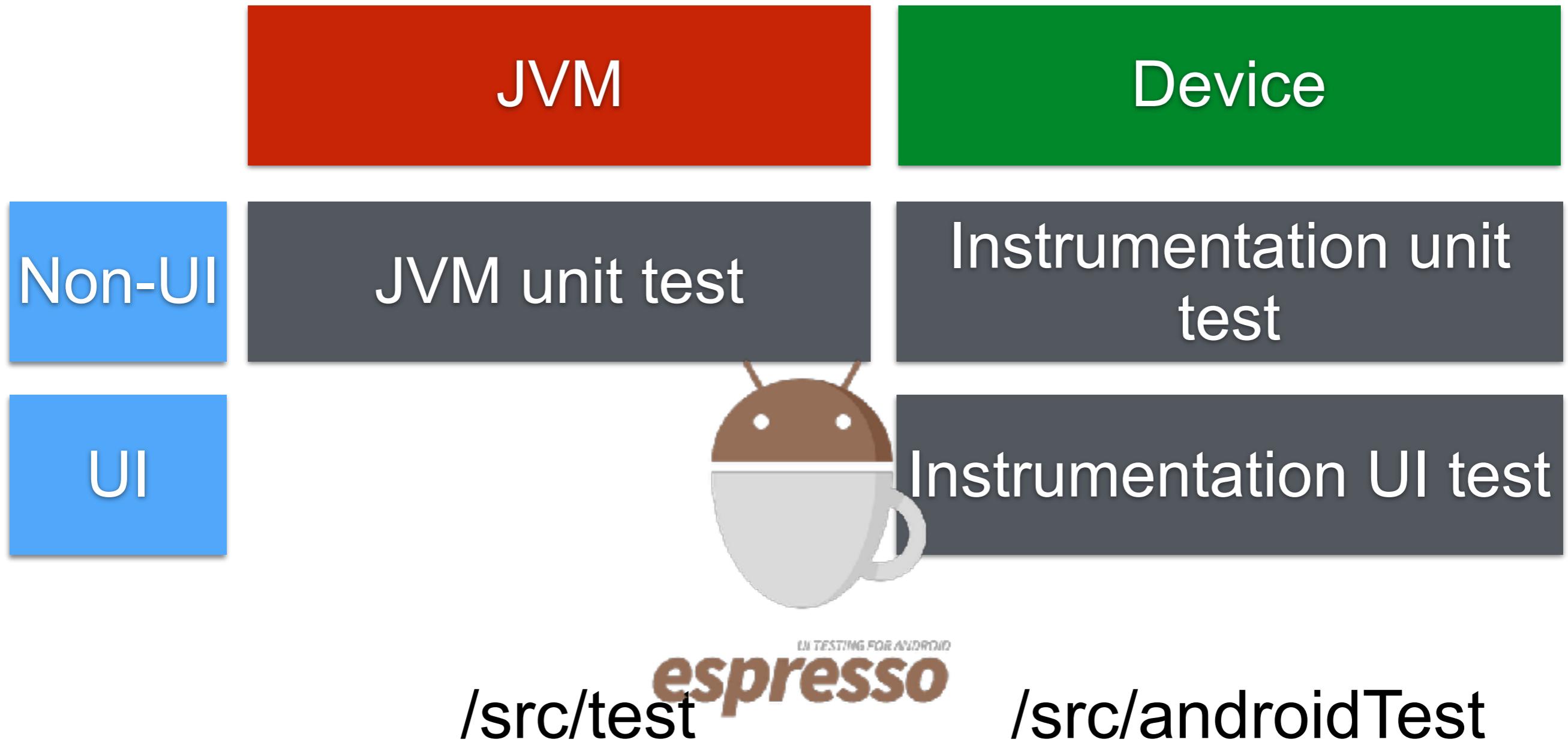
Android Testing



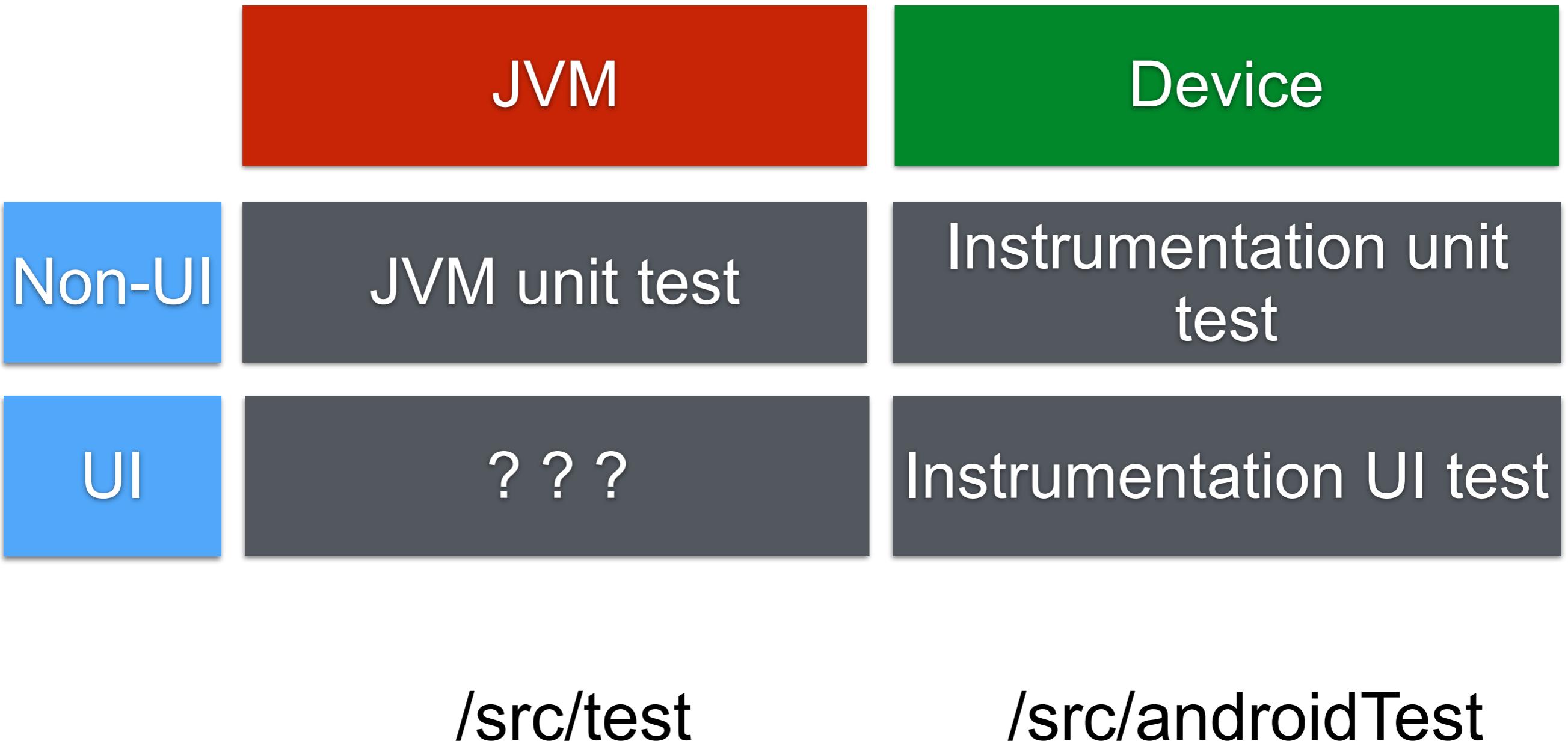
Instrumentation UI test



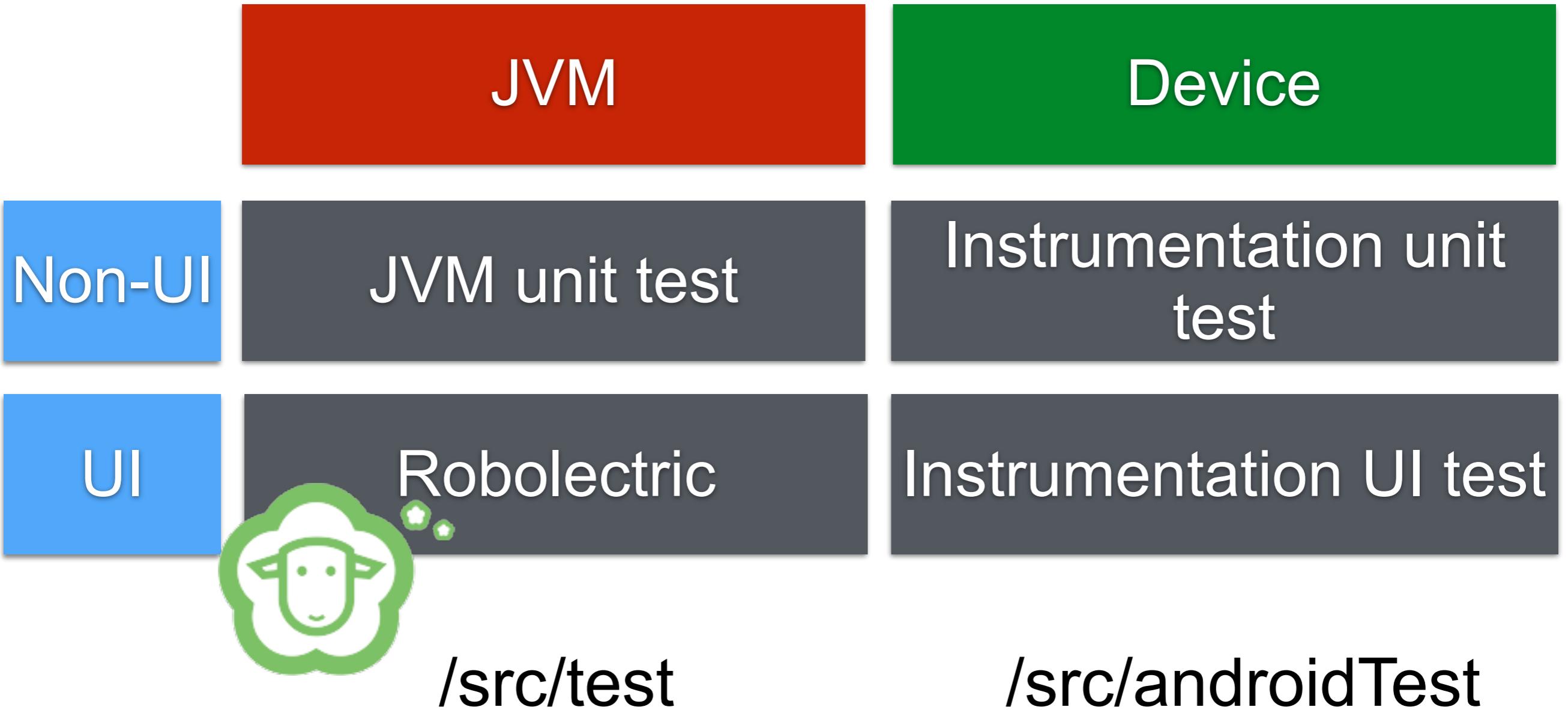
Android Testing



UI test on JVM ?



Android Testing



Resources

<https://developer.android.com/studio/test/index.html>

<https://developer.android.com/topic/libraries/testing-support-library/index.html#Espresso>

<http://robolectric.org/>



Rule of thumb

Instrumentation tests are **slower** than JVM tests.

Try to separate the standard Java code from
Android-dependent code.



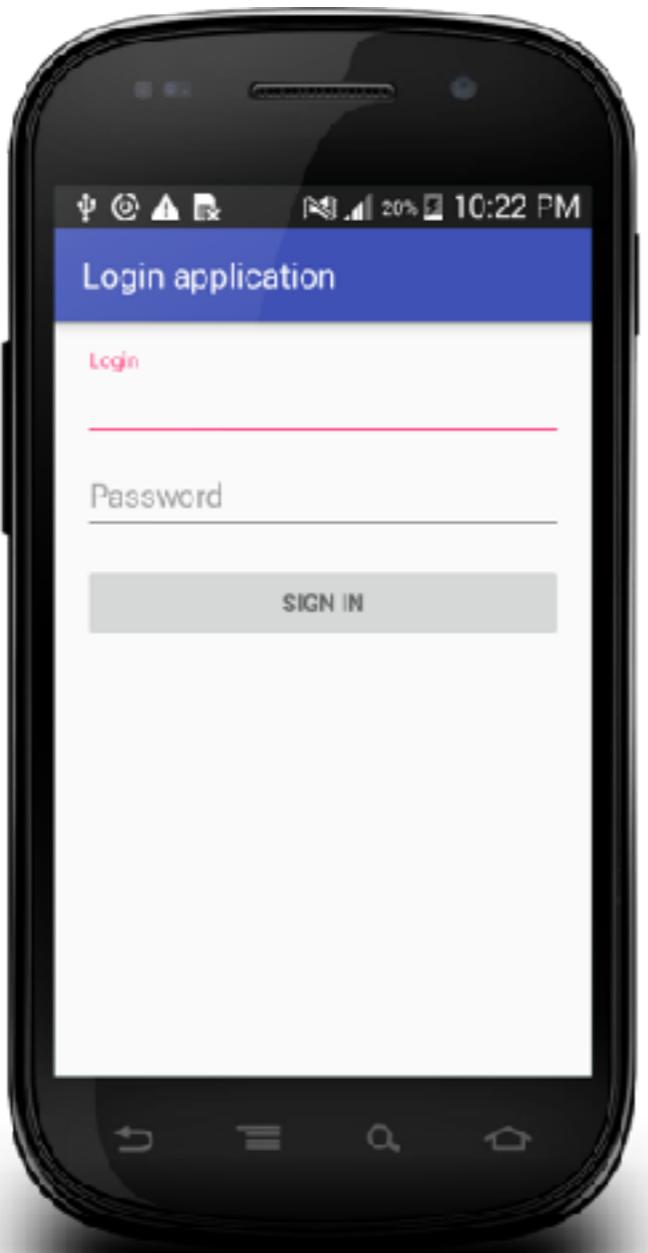
Workshop with Testing

Hello Android Testing



Login application

Login Page



Result

Success

Failure



Test cases

| Test Case Name | Email | Password | Expected Result |
|----------------------|-----------------|----------|-----------------------------|
| Login success | somkiat@xxx.com | | Show success message |
| Login fail | | | Show fail message |



Let's start to write tests with Kotlin

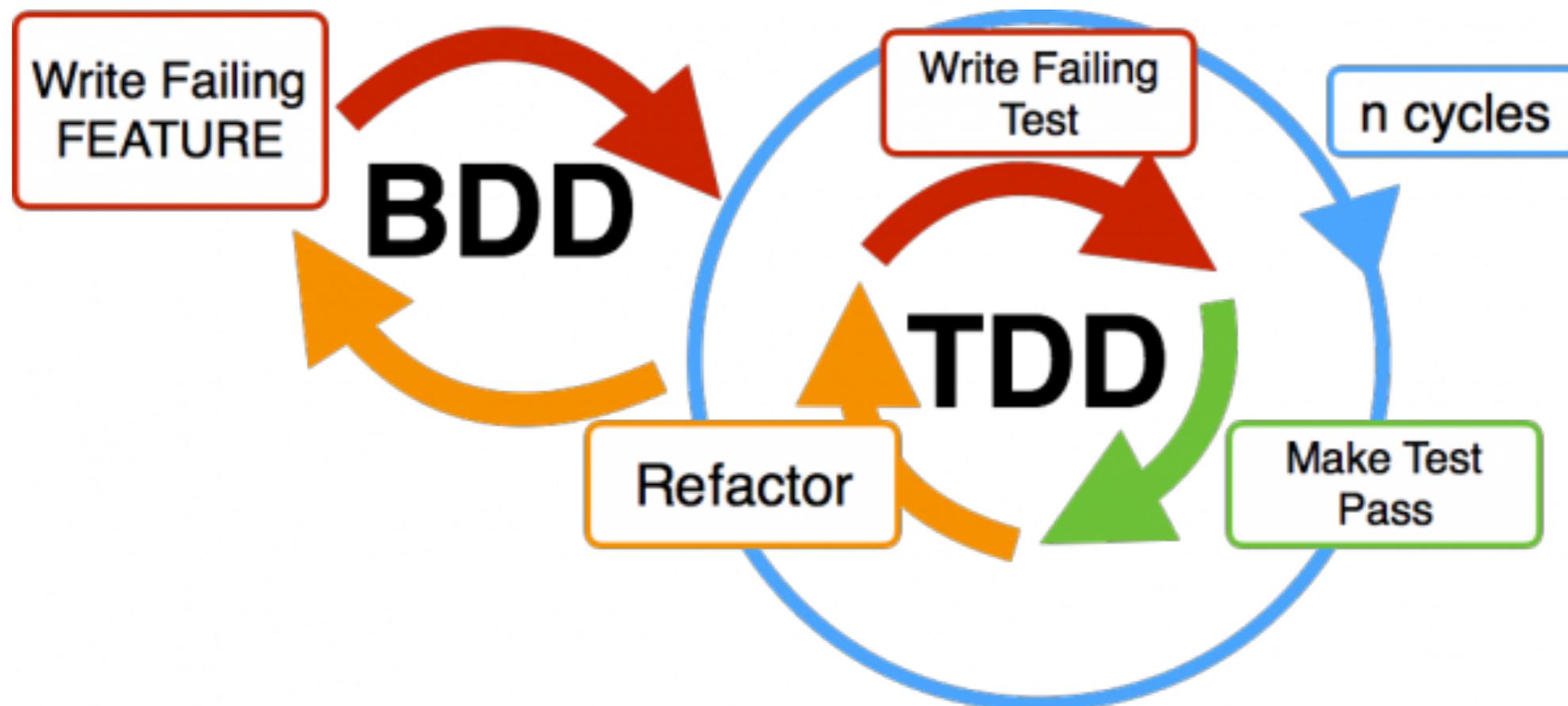


Write test case ?



Start to write UI testing

Using espresso



What is Espresso ?

A simple API for writing reliable UI tests

Develop by Google team

Compatibility with API 8 (Froyo) or higher

Add to Android Testing Support Library in 2015



<https://developer.android.com/training/testing/espresso/>



Why use Espresso ?

Small API and predictable
Easy to learn and understand
Extensible
Fast - UI thread synchronization
No wait, No Sleep
Support JUnit4



Android Studio support by default

Try to create a new project
Open file /app/build.gradle



Cheat sheet



```
onView(ViewMatcher)  
    .perform(ViewAction)  
    .check(ViewAssertion);
```

```
onData(ObjectMatcher)  
    .DataOptions  
    .perform(ViewAction)  
    .check(ViewAssertion);
```

View Matchers

USER PROPERTIES

```
withId(...)  
withText(...)  
withTagKey(...)  
withTagValue(...)  
hasContentDescription(...)  
withContentDescription(...)  
withHint(...)  
withSpinnerText(...)  
hasLinks()  
hasEllipsizedText()  
hasMultilineText()
```

HIERARCHY

```
withParent(Matcher)  
withChild(Matcher)  
hasDescendant(Matcher)  
isDescendantOfA(Matcher)  
hasSibling(Matcher)  
isRoot()
```

INPUT

```
supportsInputMethods(...)  
hasIMEAction(...)
```

UI PROPERTIES

```
isDisplayed()
```

CLASS

```
isAssignableFrom(...)
```

Data Options

```
inAdapterView(Matcher)
```

```
atPosition(Integer)
```

```
onChildView(Matcher)
```

CLICK/PRESS

```
click()  
doubleClick()  
longClick()  
pressBack()  
pressIMEActionButton()
```

GESTURES

```
scrollTo()  
swipeLeft()  
swipeRight()  
swipeUp()  
swipeDown()
```

<https://developer.android.com/training/testing/espresso/cheat-sheet>



Let's start to write tests



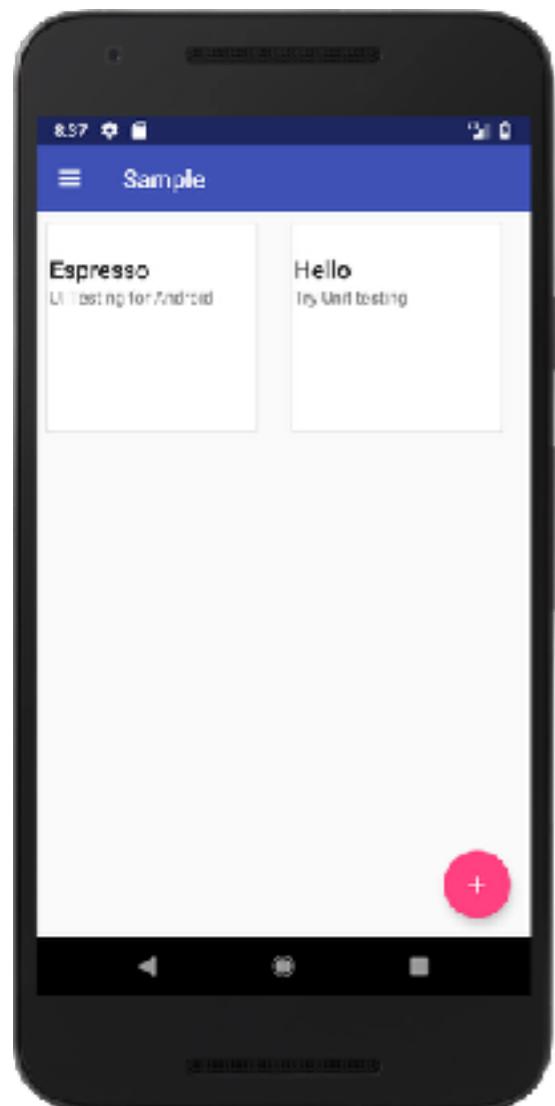
Workshop with Testing

Hello Testing with Kotlin

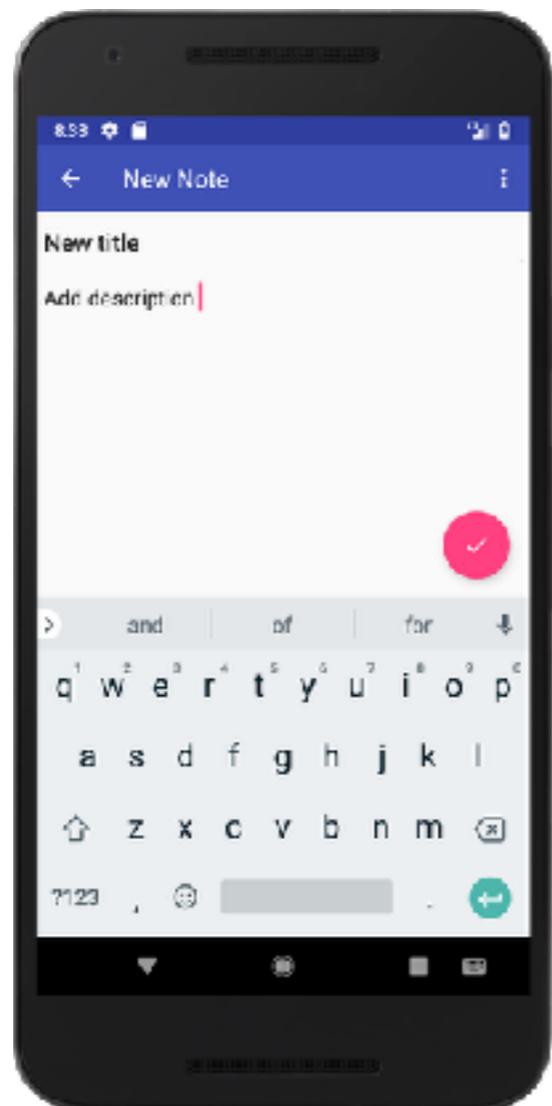
<https://github.com/up1/android-testing-workshop/archive/step01.zip>



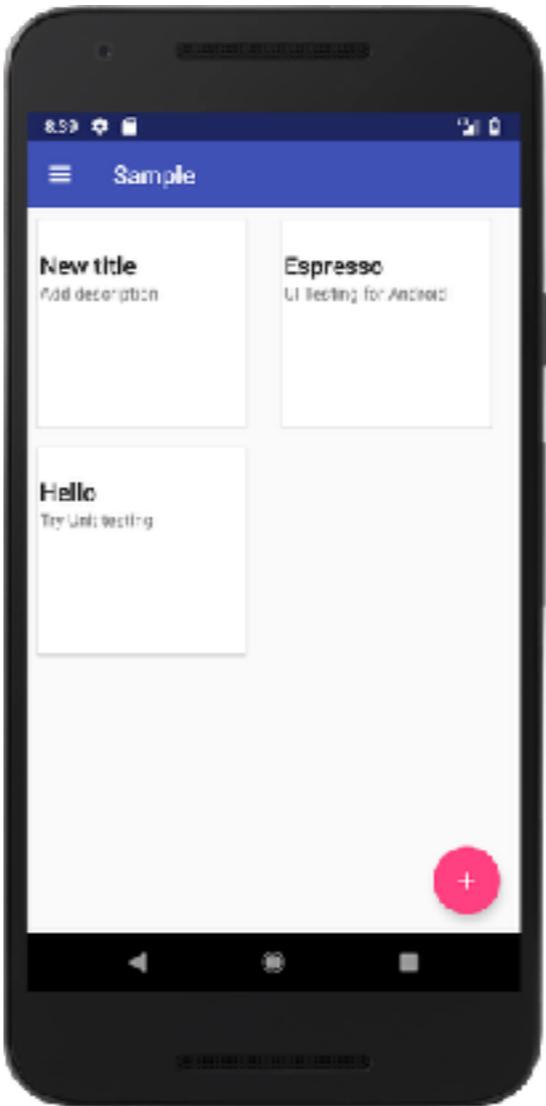
Main



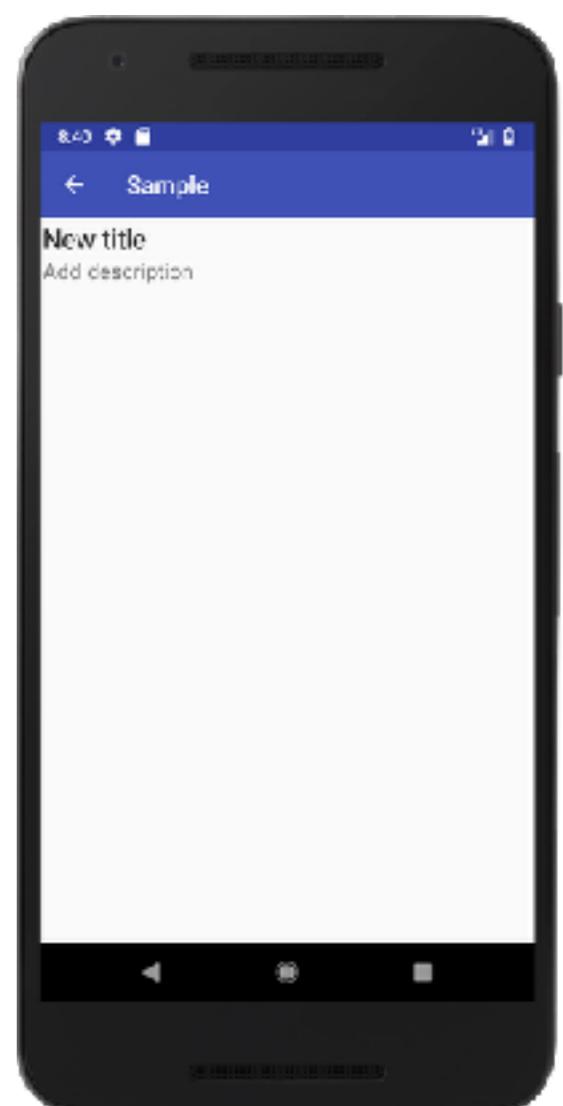
Add



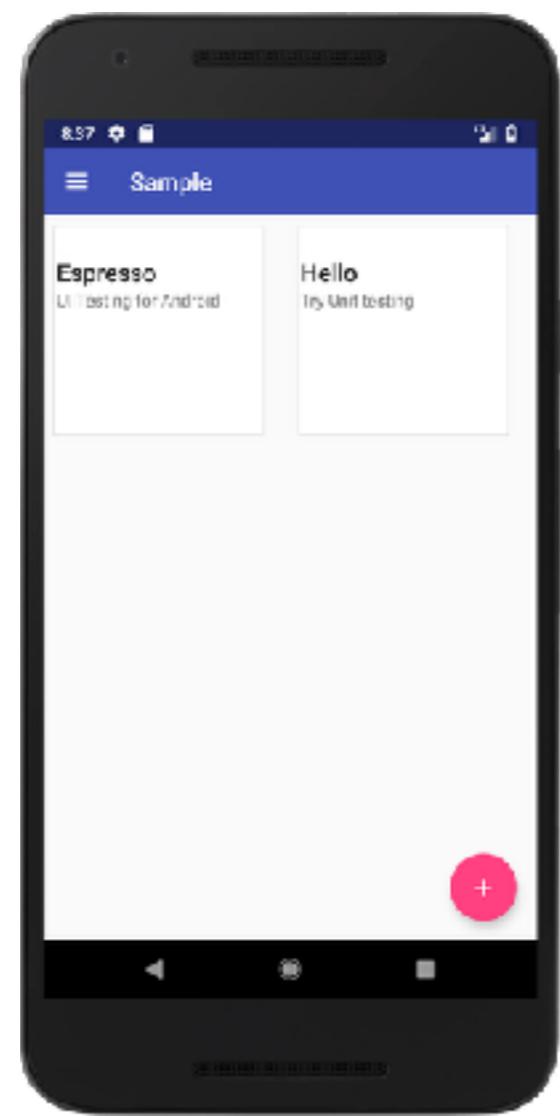
Main



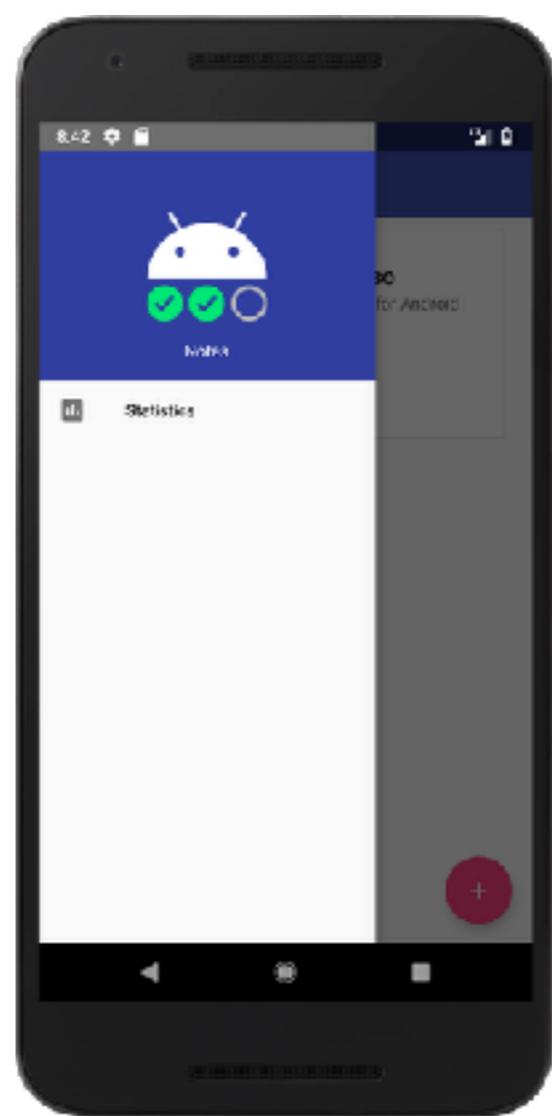
Detail



Main



Menu



Test cases ?

| Test Case Name | | | Expected Result |
|----------------|--|--|-----------------|
| | | | |
| | | | |



How to write test case ?



Add new note

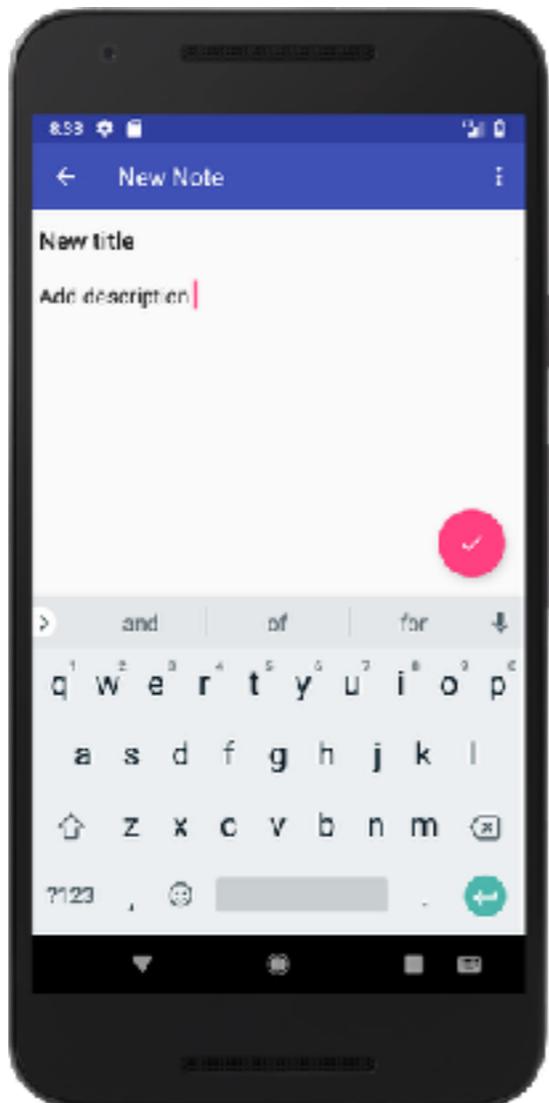


Add new note

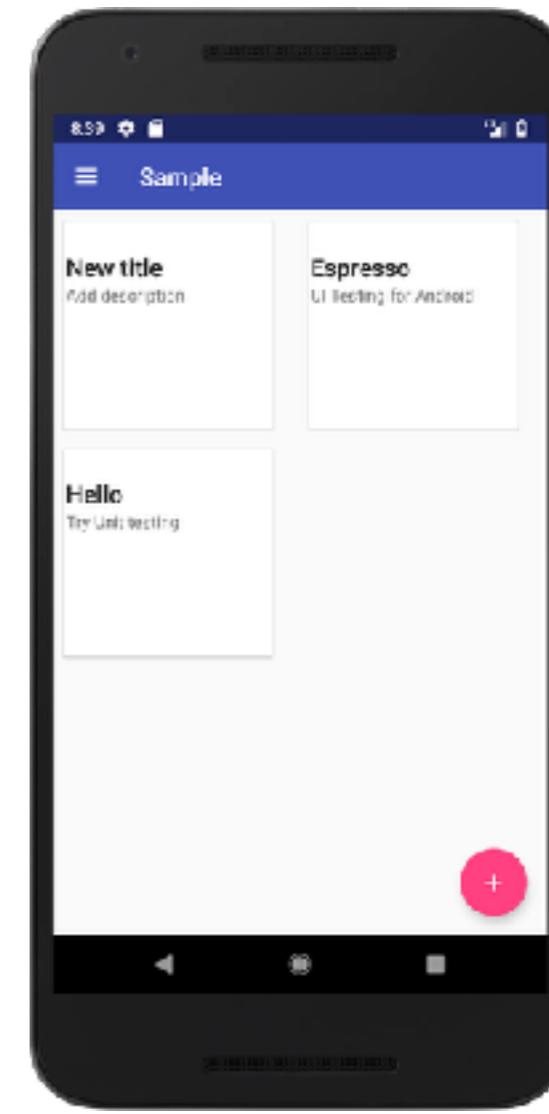
Main



Add

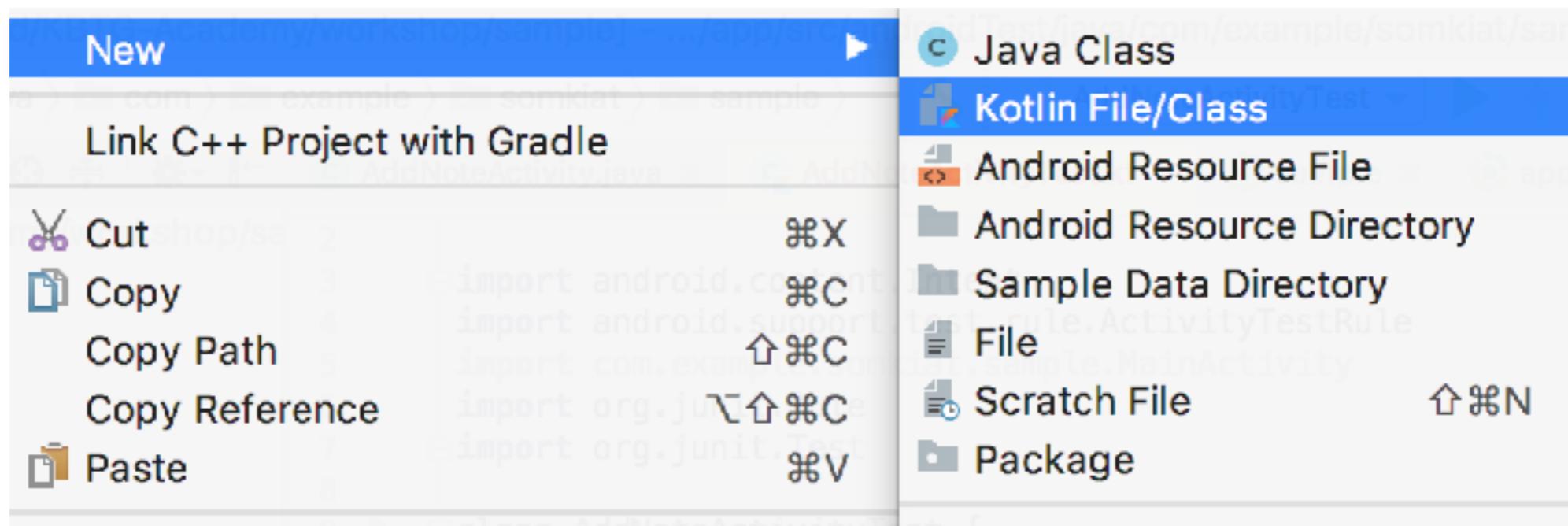


Main



Add new test class (1)

In folder /app/src/androidTest/java



Try to config Kotlin in project (1)

check result in file /app/build.gradle

```
apply plugin: 'com.android.application'  
apply plugin: 'kotlin-android'
```



Try to config Kotlin in project (2)

check result in file /app/build.gradle

```
dependencies {  
    // Unit test  
    testImplementation 'junit:junit:4.12'  
  
    // UI test with Espresso  
    androidTestImplementation 'com.android.support.test:runner:1.0.2'  
    androidTestImplementation 'com.android.support.test:rules:1.0.2'  
    androidTestImplementation  
        'com.android.support.test.espresso:espresso-core:3.0.2'  
  
    // Kotlin  
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"  
}
```



Android Studio will error !!

Try to delete folder `.idea/` and
restart
Android Studio



Android Studio will error !!

1. File -> Invalidate Caches -> Invalidate
2. File -> Close Project
3. Remove the project from Android Studio
4. Quit from Android Studio
5. Open from existing project



Write first test with Kotlin

Class name :: AddNoteFlowTest.kt

```
class AddNoteFlowTest {  
    @get:Rule  
    val rule: ActivityTestRule<MainActivity> =  
        ActivityTestRule(MainActivity::class.java)  
  
    @Test  
    fun click_add_not_button_should_open_add_note_screen() {  
        onView(withId(R.id.fab_add_notes))  
            .perform(click());  
        onView(withId(R.id.add_note_title))  
            .check(matches(isDisplayed()));  
    }  
}
```



Good test structure

```
@Test  
fun click_add_not_button_should_open_add_note_screen() {  
    // Act  
    onView(withId(R.id.fab_add_notes))  
        .perform(click())  
  
    // Assert  
    onView(withId(R.id.add_note_title))  
        .check(matches(isDisplayed()))  
    onView(withId(R.id.add_note_description))  
        .check(matches(isDisplayed()))  
}
```



Add a new note (1)

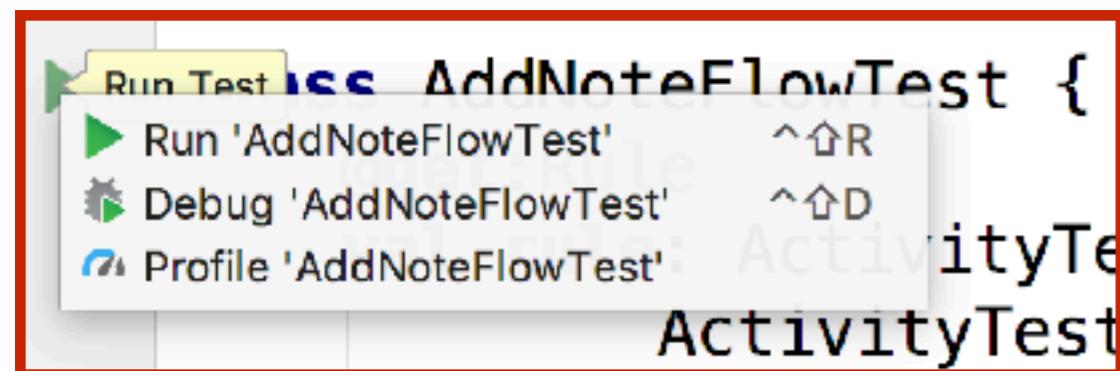
Create new test case to add a new note

```
@Test  
fun add_a_new_note_to_note_list() {  
    val newTitle = "New title"  
    val newDescription = "New description"  
  
    // Click Add new note  
    onView(withId(R.id.fab_add_notes))  
        .perform(click())  
  
    // Fill in data  
    onView(withId(R.id.add_note_title))  
        .perform(typeText(newTitle),  
                closeSoftKeyboard());  
    onView(withId(R.id.add_note_description))  
        .perform(typeText(newDescription),  
                closeSoftKeyboard());
```



Run test in Android Studio (1)

Run all test cases in class



```
Run Test > AddNoteFlowTest {
    ▶ Run 'AddNoteFlowTest'      ^↑R
    ▶ Debug 'AddNoteFlowTest'   ^↑D
    ▶ Profile 'AddNoteFlowTest'
ActivityTestRule<MainActivity> =
    ActivityTestRule(MainActivity::class.java)

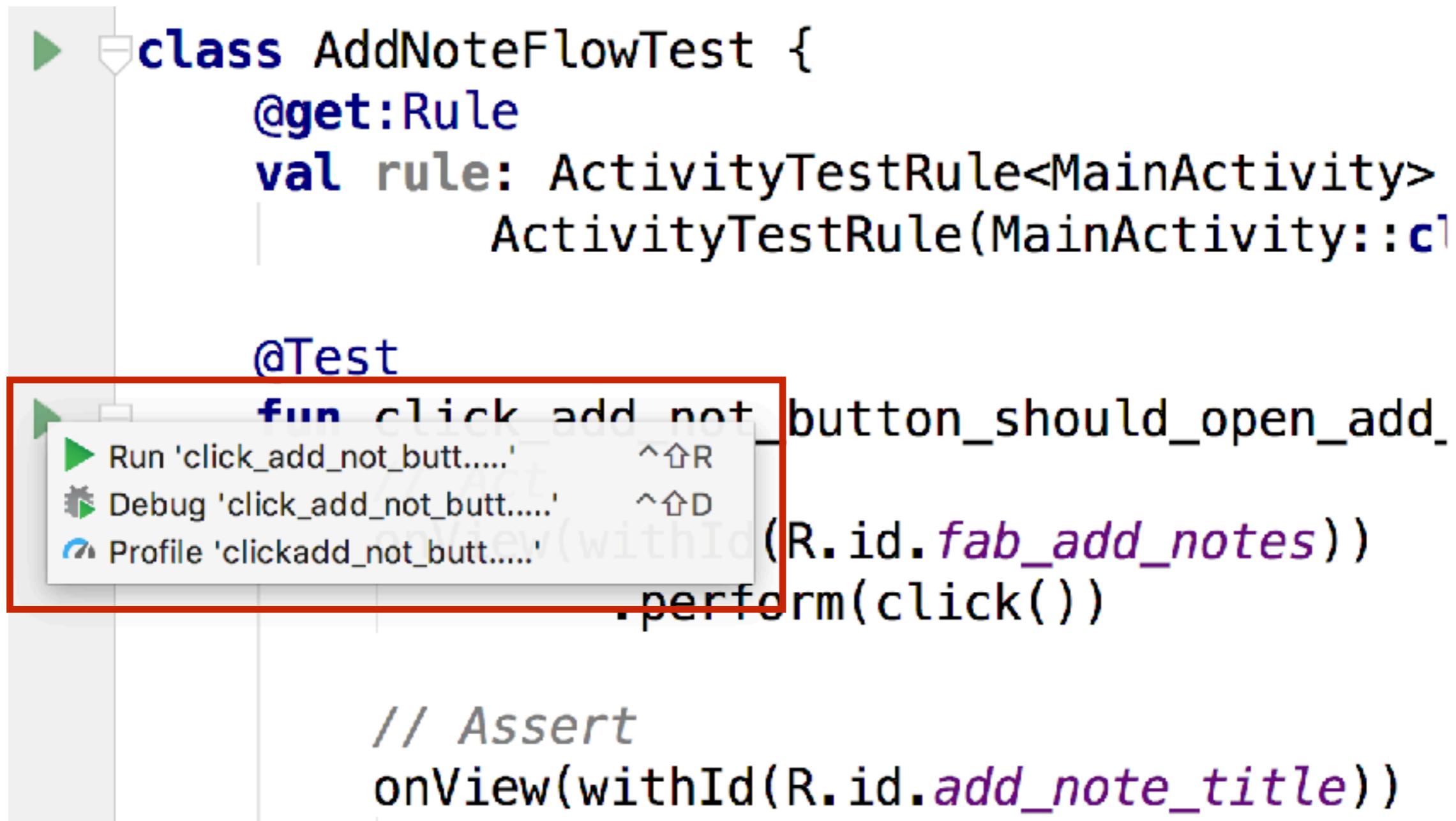
    @Test
    fun click_add_not_button_should_open_add_note_screen() {
        // Act
        onView(withId(R.id.fab_add_notes))
            .perform(click())

        // Assert
        onView(withId(R.id.add_note_title))
            .check(matches(isDisplayed())))
    }
}
```



Run test in Android Studio (2)

Run by test case



```
▶ class AddNoteFlowTest {
    @get:Rule
    val rule: ActivityTestRule<MainActivity>
        ActivityTestRule(MainActivity::cl

    @Test
    fun click_add_not_button_should_open_add_
        Run 'click_add_not_but....'      ^↑R
        Debug 'click_add_not_but....'   ^↑D
        Profile 'clickadd_not_but....'  ↴
            (R.id.fab_add_notes))
                .perform(click())

        // Assert
        onView(withId(R.id.add_note_title))
```



Run test in command line

```
./gradlew clean cAT
```



See test result

In /app/build/reports/androidTests/connected

Test Summary

| | | | |
|---------|------------|------------------|-----------------|
| 4 tests | 0 failures | 11.453s duration | 100% successful |
|---------|------------|------------------|-----------------|

Packages **Classes**

| Package | Tests | Failures | Duration | Success rate |
|--|-------|----------|----------|--------------|
| com.example.somkiat.sample | 3 | 0 | 11.205s | 100% |
| com.example.somkiat.sample.addnote | 1 | 0 | 0.248s | 100% |

Generated by [Gradle 4.4](#) at Jul 30, 2018 10:59:25 PM



See test result of AddNoteFlowTest

Class com.example.somkiat.sample.AddNoteFlowTest

[all > com.example.somkiat.sample > AddNoteFlowTest](#)

| | | | |
|-------------------|----------------------|----------------------------|---------------------------|
| 2 tests | 0 failures | 11.175s duration | 100% successful |
|-------------------|----------------------|----------------------------|---------------------------|

Tests

| Test | SM-G360HU - 4.4.4 |
|--|-------------------|
| add_a_new_note_to_note_list | passed (9.726s) |
| click_add_not_button_should_open_add_note_screen | passed (1.449s) |



Add a new note (2)

Save and check result

```
// Save note  
onView(withId(R.id.fab_add_notes)).perform(click());  
  
// Scroll to new note, by finding description  
onView(withId(R.id.notes_list))  
    .perform(scrollTo<RecyclerView.ViewHolder>(  
        hasDescendant(withText(newDescription))));  
  
// Check description displayed on screen  
onView(itemText(newDescription)).check(matches(isDisplayed()));
```



Add a new note (3)

Try to create a custom matcher **withItemText()**

```
// Save note  
onView(withId(R.id.fab_add_notes)).perform(click());  
  
// Scroll to new note, by finding description  
onView(withId(R.id.notes_list))  
    .perform(scrollTo<RecyclerView.ViewHolder>(  
        hasDescendant(withText(newDescription))));  
  
// Check description displayed on screen  
onView(withItemText(newDescription)).check(matches(isDisplayed()));
```



Custom matcher

```
private fun withItemText(itemText: String): Matcher<View> {
    checkArgument(!TextUtils.isEmpty(itemText), "itemText cannot be null or empty")
    return object : TypeSafeMatcher<View>() {
        public override fun matchesSafely(item: View): Boolean {
            return allOf(
                isDescendantOfA(isAssignableFrom(RecyclerView::class.java)),
                withText(itemText)).matches(item)
        }

        override fun describeTo(description: Description) {
            description.appendText("is isDescendantOfA RV with text $itemText")
        }
    }
}
```

<https://github.com/up1/android-testing-workshop/wiki>



Try to add more note !!



Working with Data Parameterized

<https://github.com/junit-team/junit4/wiki/parameterized-tests>



Working with Parameterized

Class name ::
AddNoteFlowWithParameterizedTest.kt

```
@RunWith(Parameterized::class)
class AddNoteFlowWithParameterizedTest
    (private val newTitle: String,
     private val newDescription: String) {

    @get:Rule
    val rule: ActivityTestRule<MainActivity> =
        ActivityTestRule(MainActivity::class.java)
```



1. Run with Parameterized

```
@RunWith(Parameterized::class)
class AddNoteFlowWithParameterizedTest
    (private val newTitle: String,
     private val newDescription: String) {

    @get:Rule
    val rule: ActivityTestRule<MainActivity> =
        ActivityTestRule(MainActivity::class.java)
```



2. Define constructor of class

```
@RunWith(Parameterized::class)
class AddNoteFlowWithParameterizedTest
    (private val newTitle: String,
     private val newDescription: String) {

    @get:Rule
    val rule: ActivityTestRule<MainActivity> =
        ActivityTestRule(MainActivity::class.java)
```



3. Define your data for test

```
companion object {  
    @JvmStatic  
    @Parameterized.Parameters  
    fun data(): List<Array<String>> {  
        return listOf(  
            arrayOf("Title 1", "Description 1"),  
            arrayOf("Title 2", "Description 2")  
        )  
    }  
}
```

<https://kotlinlang.org/docs/reference/object-declarations.html#companion-objects>



4. Change your test case

```
@Test  
fun add_a_new_note_to_note_list() {  
    // Click Add new note  
    onView(withId(R.id.fab_add_notes))  
        .perform(click())  
  
    // Fill in data  
    onView(withId(R.id.add_note_title))  
        .perform(typeText(newTitle),  
                closeSoftKeyboard());  
    onView(withId(R.id.add_note_description))  
        .perform(typeText(newDescription),  
                closeSoftKeyboard());  
  
    // Save note  
    onView(withId(R.id.fab_add_notes)).perform(click());  
}
```



Run and see test result

Class com.example.somkiat.sample.AddNoteFlowWithParameterizedTest

[all](#) > [com.example.somkiat.sample](#) > AddNoteFlowWithParameterizedTest



| Tests | |
|--------------------------------|--------------------------|
| Test | |
| add_a_new_note_to_note_list[0] | SM-G360HU - 4.4.4 |
| add_a_new_note_to_note_list[1] | passed (8.871s) |
| | passed (8.931s) |

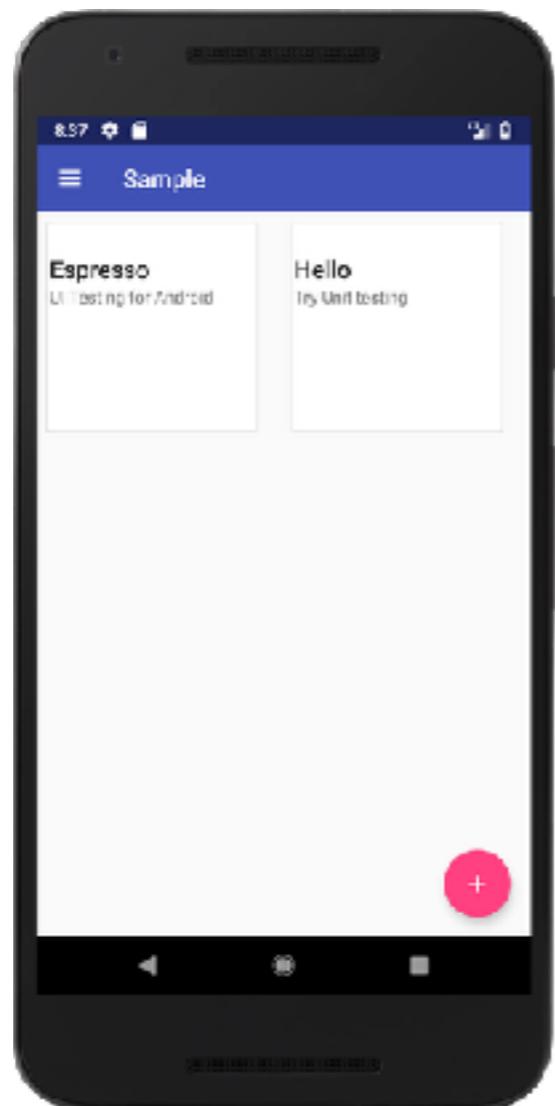


Show detail of note



Show detail of note

Main



Detail



Show detail of note

Create new test class :: NoteDetailScreenTest.kt

```
class NoteDetailScreenTest {  
    @get:Rule  
    private val rule: ActivityTestRule<NoteDetailActivity> =  
        ActivityTestRule(NoteDetailActivity::class.java,  
                         true, // Initial touch mode  
                         false) // Lazily launch activity
```



Show detail of note

Create a test case

```
@Test  
fun show_detail_of_note_in_screen() {  
    // Arrange and Act  
    val startIntent = Intent()  
    // ID of note ?  
    startIntent.putExtra(NoteDetailActivity.EXTRA_NOTE_ID, "0")  
    rule.launchActivity(startIntent)  
  
    // Assert  
    onView(withId(R.id.note_detail_title))  
        .check(matches(withText(""))) // Expected value ?  
    onView(withId(R.id.note_detail_description))  
        .check(matches(withText(""))); // Expected value ?  
}
```



Problem ?

Note ID ?

```
@Test
fun show_detail_of_note_in_screen() {
    // Arrange and Act
    val startIntent = Intent()
    // ID of note ?
    startIntent.putExtra(NoteDetailActivity.EXTRA_NOTE_ID, "0")
    rule.launchActivity(startIntent)

    // Assert
    onView(withId(R.id.note_detail_title))
        .check(matches(withText("")))
    onView(withId(R.id.note_detail_description))
        .check(matches(withText("")));
}
```



Problem ?

Expected result/value ?

```
@Test  
fun show_detail_of_note_in_screen() {  
    // Arrange and Act  
    val startIntent = Intent()  
    // ID of note ?  
    startIntent.putExtra(NoteDetailActivity.EXTRA_NOTE_ID, "0")  
    rule.launchActivity(startIntent)  
  
    // Assert  
    onView(withId(R.id.note_detail_title))  
        .check(matches(withText(""))) // Expected value ?  
    onView(withId(R.id.note_detail_description))  
        .check(matches(withText(""))); // Expected value ?  
}
```



Your project can test or not ?

Testable application ?



Look inside your project



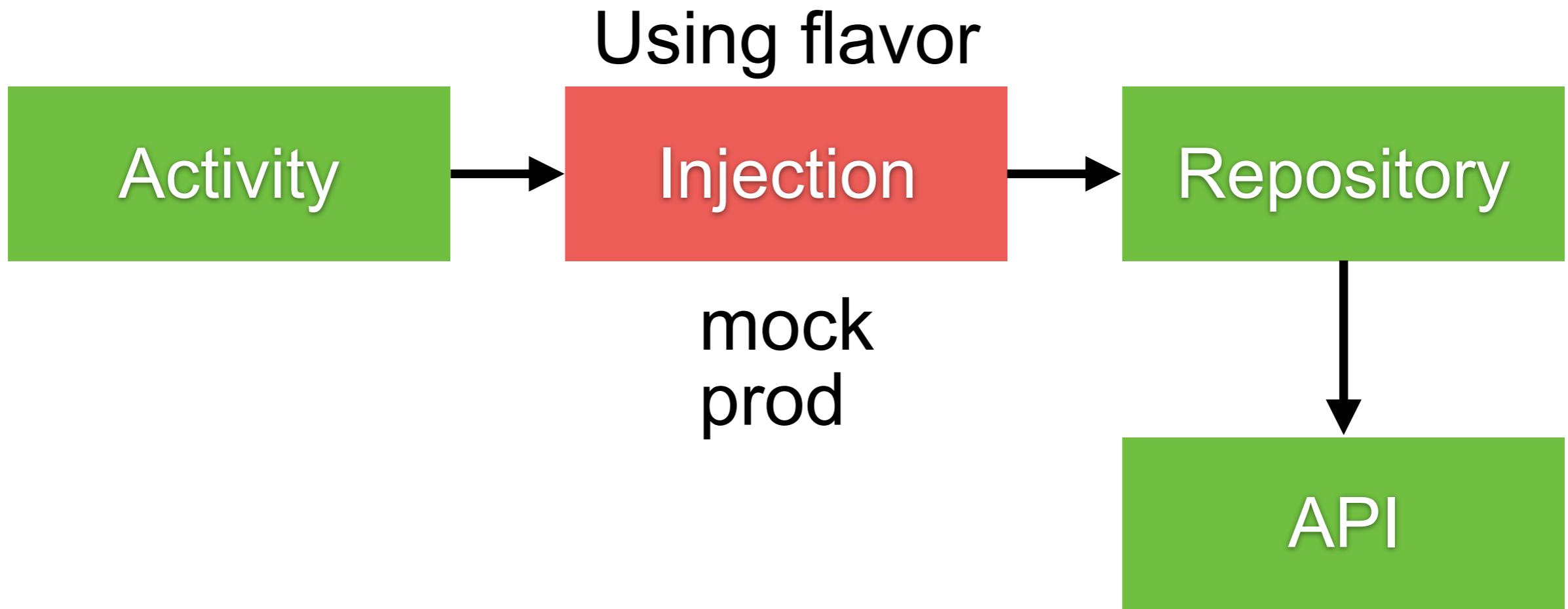
Injection.java

Use to create Repository and API

```
public class Injection {  
    public static ImageStorage provideImageFile() {  
        return new ImageFile();  
    }  
  
    public static NoteRepository provideNotesRepository() {  
        return new InMemoryNotesRepository(new MockNoteServiceApi());  
    }  
}
```



Concept :: More testable



Add product flavour

To file /app/build.gradle

```
flavorDimensions "mock"
```

```
// If you need to add more flavors, consider using flavor dimensions.
```

```
productFlavors {  
    mock {  
        applicationIdSuffix = ".mock"  
        dimension "mock"  
    }  
    prod {  
    }  
}
```

```
// Remove mockRelease as it's not needed.
```

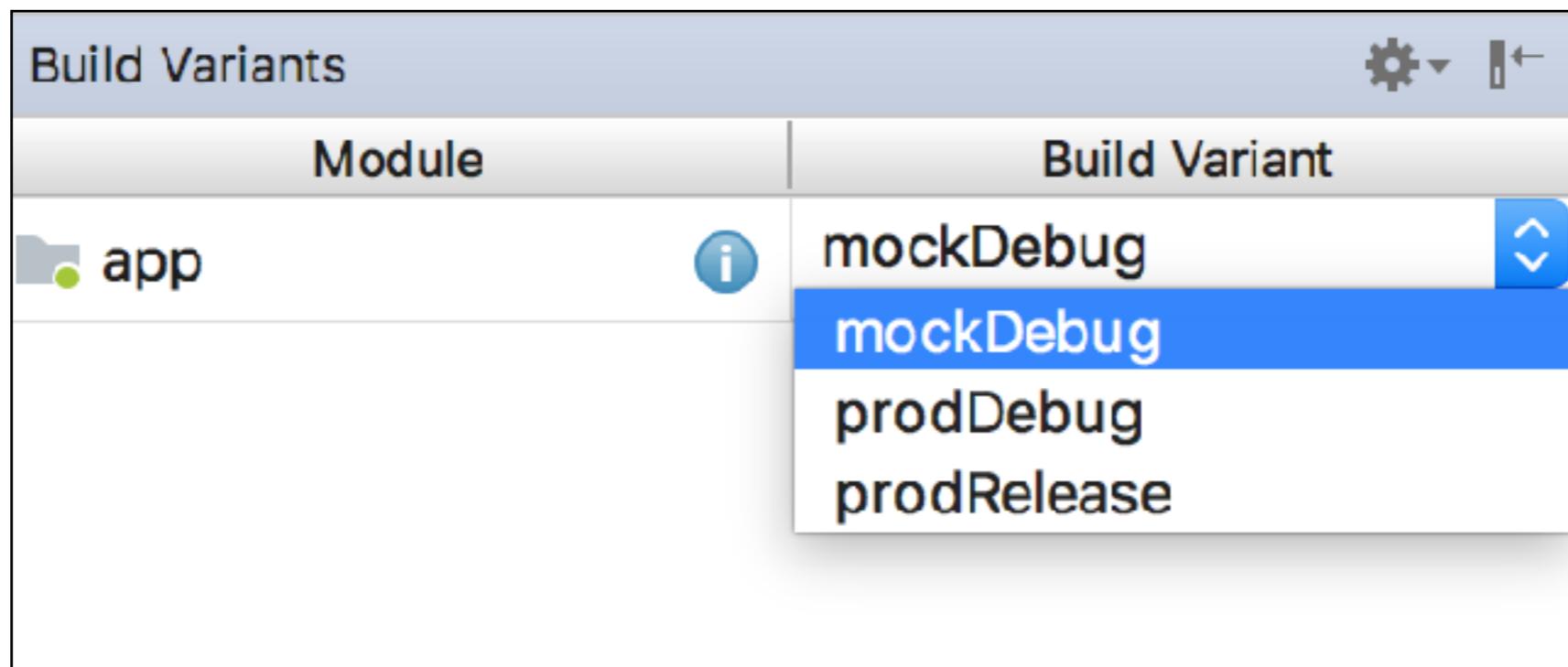
```
android.variantFilter { variant ->  
    if(variant.buildType.name.equals('release')  
        && variant.getFlavors().get(0).name.equals('mock')) {  
        variant.setIgnore(true);  
    }  
}
```

<https://github.com/up1/android-testing-workshop/wiki>



Add product flavour

See result in Build Variants



Create new folder in /app/src

**mock/java/com/example/somkiat/sample
prod/java/com/example/somkiat/sample**



Move class Injection into prod

prod/java/com/example/somkiat/sample



Run with prodDebug

| Module | Build Variant |
|--------|---------------|
| app | prodDebug |



Run with prodDebug

```
./gradlew clean connectedProdDebugAndroidTest
```



Create class Injection in mock

mock/java/com/example/somkiat/sample

```
public class Injection {  
    public static ImageStorage provideImageFile() {  
        return new FakeImageFile();  
    }  
  
    public static NoteRepository provideNotesRepository() {  
        return new InMemoryNotesRepository(new FakeNoteServiceApi());  
    }  
}
```

Create file **FakeNoteServiceApi**



Create class FakeNoteServiceApi

mock/java/com/example/somkiat/sample

```
class FakeNoteServiceApi implements NoteServiceApi {
    private static final Map<String, Note> NOTES_SERVICE_DATA = new HashMap<>();

    @Override
    public void getAllNotes(NotesServiceCallback<List<Note>> callback) {
        callback.onLoaded(Lists.newArrayList(NOTES_SERVICE_DATA.values()));
    }

    @Override
    public void getNote(String noteId, NotesServiceCallback<Note> callback) {
        Note note = NOTES_SERVICE_DATA.get(noteId);
        callback.onLoaded(note);
    }

    @Override
    public void saveNote(Note note) {
        NOTES_SERVICE_DATA.put(note.getId(), note);
    }

    @VisibleForTesting
    public static void addNotes(Note... notes) {
        for (Note note : notes) {
            NOTES_SERVICE_DATA.put(note.getId(), note);
        }
    }
}
```

<https://github.com/up1/android-testing-workshop/wiki>



Create new folder in /app/src

androidTestMock/java/com/example/somkiat/sample

Move your test class to this folder



Back to show detail test



Show detail of note

```
@Test  
fun show_detail_of_note_in_screen() {  
    // Arrange and Act  
    val newNote = Note("Fake title", "Fake description");  
    FakeNoteServiceApi.addNotes(newNote);  
    val startIntent = Intent()  
    startIntent.putExtra(NoteDetailActivity.EXTRA_NOTE_ID, newNote.id)  
    rule.launchActivity(startIntent)  
  
    // Assert  
    onView(withId(R.id.note_detail_title))  
        .check(matches(withText(newNote.title)))  
    onView(withId(R.id.note_detail_description))  
        .check(matches(withText(newNote.description)));  
}
```



Run with mockDebug

```
./gradlew clean connectedMockDebugAndroidTest
```



Test result

Test Summary

4 tests 0 failures 11.064s duration

100%
successful

Packages

Classes

| Package | Tests | Failures | Duration | Success rate |
|---|-------|----------|----------|--------------|
| com.example.somkiat.sample | 2 | 0 | 10.707s | 100% |
| com.example.somkiat.sample.addnote | 1 | 0 | 0.220s | 100% |
| com.example.somkiat.sample.notedetail | 1 | 0 | 0.137s | 100% |



Code coverage



"Code coverage can show the high risk areas in a program, but never the risk-free."

Paul Reilly, 2018, Kotlin TDD with Code Coverage



Code coverage

A tool to measure how much of your code is covered by tests that break down into classes, methods and lines.



Code coverage

But 100% of code coverage **does not mean** that your code is 100% correct



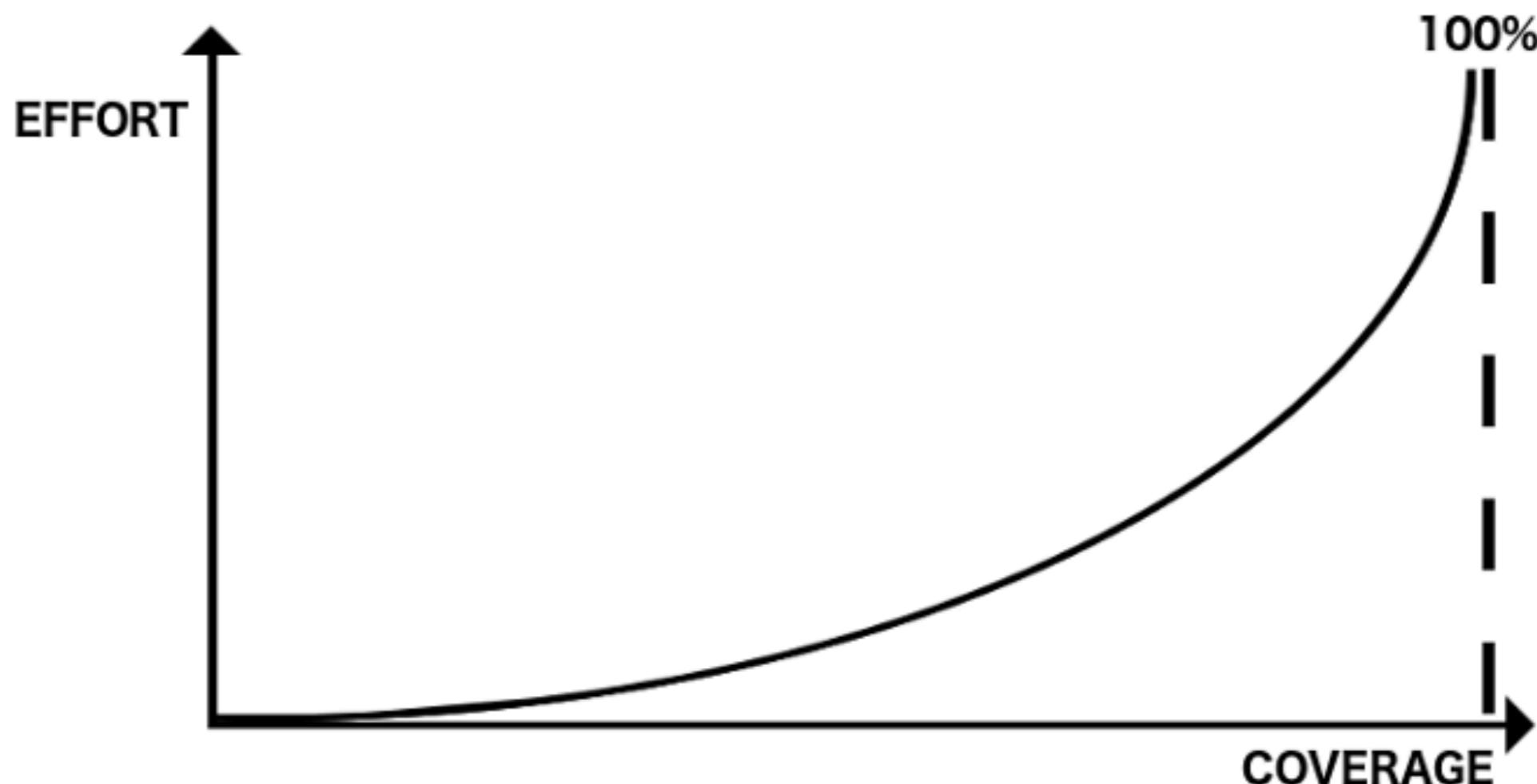
Code coverage

Powerful tool to improve the quality of your code

Code coverage != quality of tests



Code coverage 100% ?



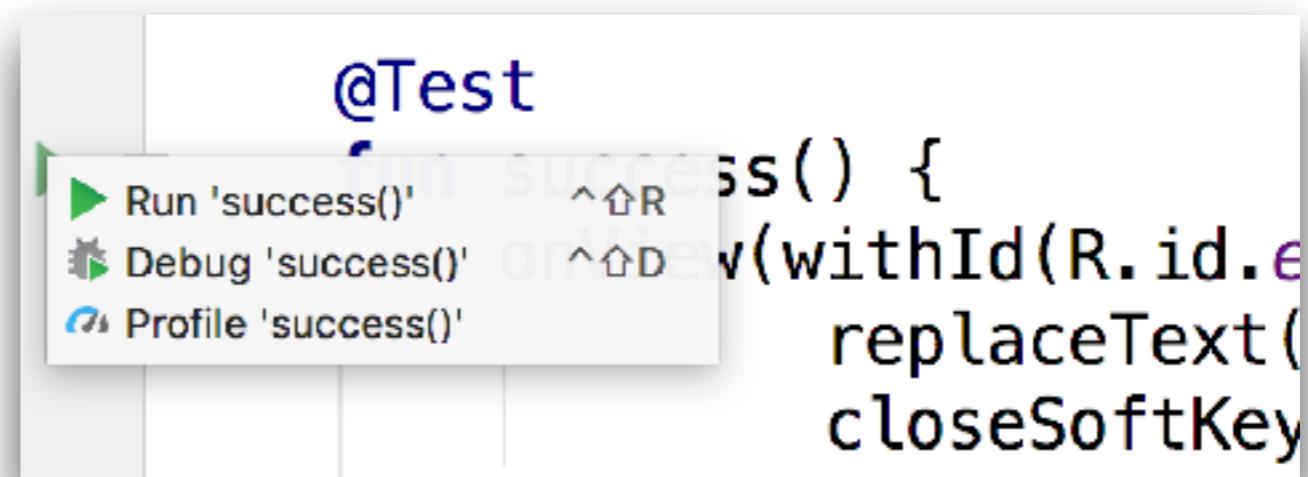
Code coverage for android

Disabled by default !!

Unit test



androidTest



Try to enable code coverage

Use Jacoco library

The logo for Jacoco, featuring the word "JACOCO" in large red letters above the words "Java Code Coverage" in smaller blue letters.
<https://github.com/nomisRev/AndroidGradleJacoco>

<https://www.jacoco.org/jacoco/>



1. Enable coverage in debug

In file /app/build.gradle

```
buildTypes {  
    debug {  
        testCoverageEnabled true  
    }  
  
    release {  
        minifyEnabled false  
        proguardFiles getDefaultProguard...  
    }  
}
```



2. Use jacobo library

In file /app/build.gradle

```
apply plugin: 'jacoco'

jacoco {
    toolVersion = "0.8.1"
}
```



3. Merge result of coverage

Create new grade's task

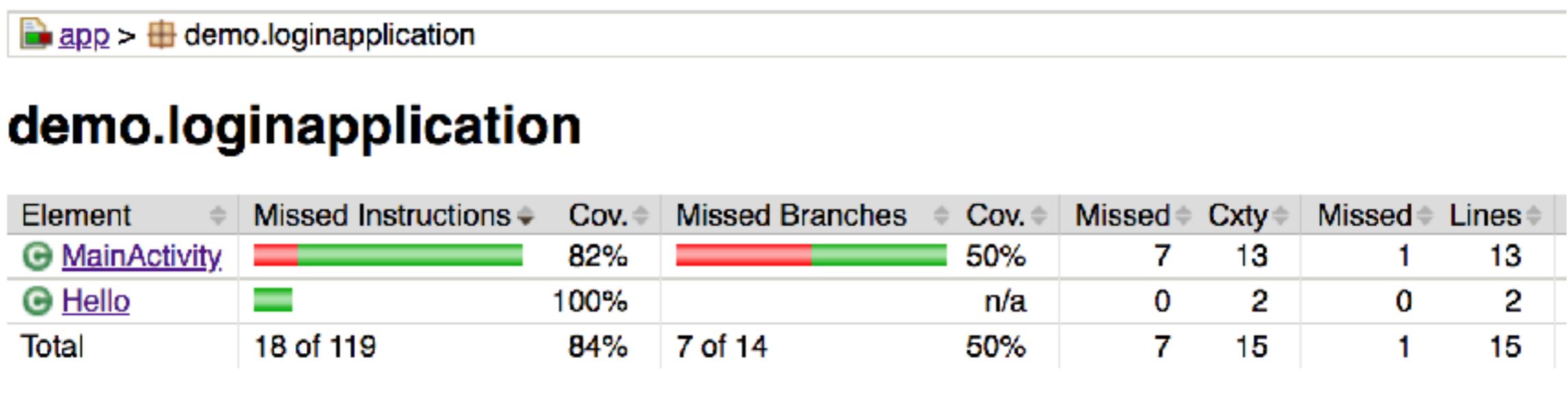
```
task jacocoTestReport(type: JacocoReport, dependsOn: ['testDebugUnitTest',  
    reports {  
        xml.enabled = true  
        html.enabled = true  
    }  
  
    def fileFilter = [ '**/R.class', '**/R*.class', '**/BuildConfig.*', '*  
    def debugTree = fileTree(dir: "$project.buildDir/tmp/kotlin-classes/debug")  
    def mainSrc = "$project.projectDir/src/main/java"  
  
    sourceDirectories = files([mainSrc])  
    classDirectories = files([debugTree])  
    executionData = fileTree(dir: project.buildDir, includes: [  
        'jacoco/testDebugUnitTest.exec', 'outputs/code-coverage/connected'  
    ])  
}
```



4. Run test with code coverage

\$gradlew clean jacocoTestReport

Result in /build/reports/jacoco/jacocoTestReport



More detail of coverage

app > demo.loginapplication > MainActivity.kt

MainActivity.kt

```
1. package demo.loginapplication
2.
3. import android.support.v7.app.AppCompatActivity
4. import android.os.Bundle
5. import android.view.View
6. import android.widget.Toast
7. import kotlinx.android.synthetic.main.activity_main.*
8.
9. class MainActivity : AppCompatActivity(), View.OnClickListener {
10.
11.     override fun onCreate(savedInstanceState: Bundle?) {
12.         super.onCreate(savedInstanceState)
13.         setContentView(R.layout.activity_main)
14.
15.         email_sign_in_button.setOnClickListener(this)
16.     }
17.
18.     override fun onClick(v: View?) {
19.         when(v?.id) {
20.             R.id.email_sign_in_button -> {
21.                 processLogin()
22.             }
23.         }
24.     }
25.
26.     private fun processLogin() {
27.         if(email.text.toString().equals("somkiatxxx.com")) {
28.             Toast.makeText(this@MainActivity, "Success", Toast.LENGTH_SHORT).show()
29.         } else {
30.             Toast.makeText(this@MainActivity, "Fail", Toast.LENGTH_SHORT).show()
31.         }
32.     }
33. }
```

Miss in branch coverage

Miss in line coverage



Problem with Jacoco

version of Jacoco

<https://github.com/nomisRev/AndroidGradleJacoco>

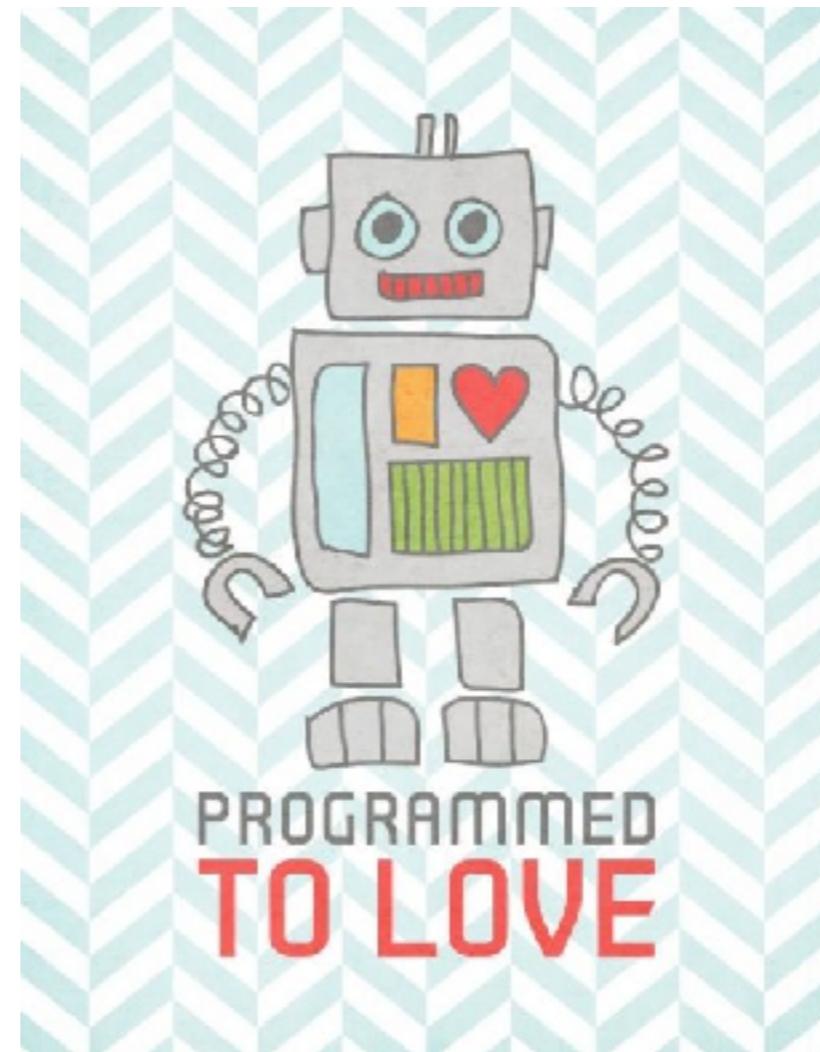


Improve UI Testing with Espresso



1. Improve test structure

Robot or Page Object pattern

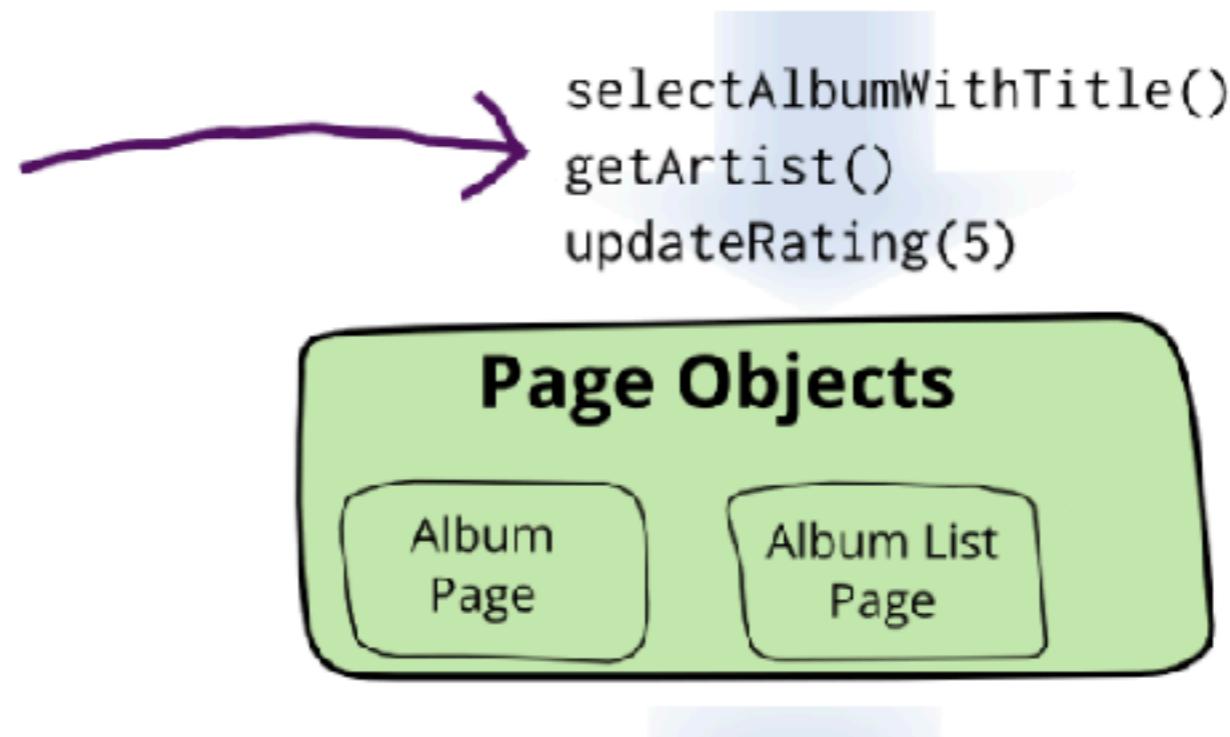


<https://martinfowler.com/bliki/PageObject.html>



Page Object / Robot pattern

this API is about
the application

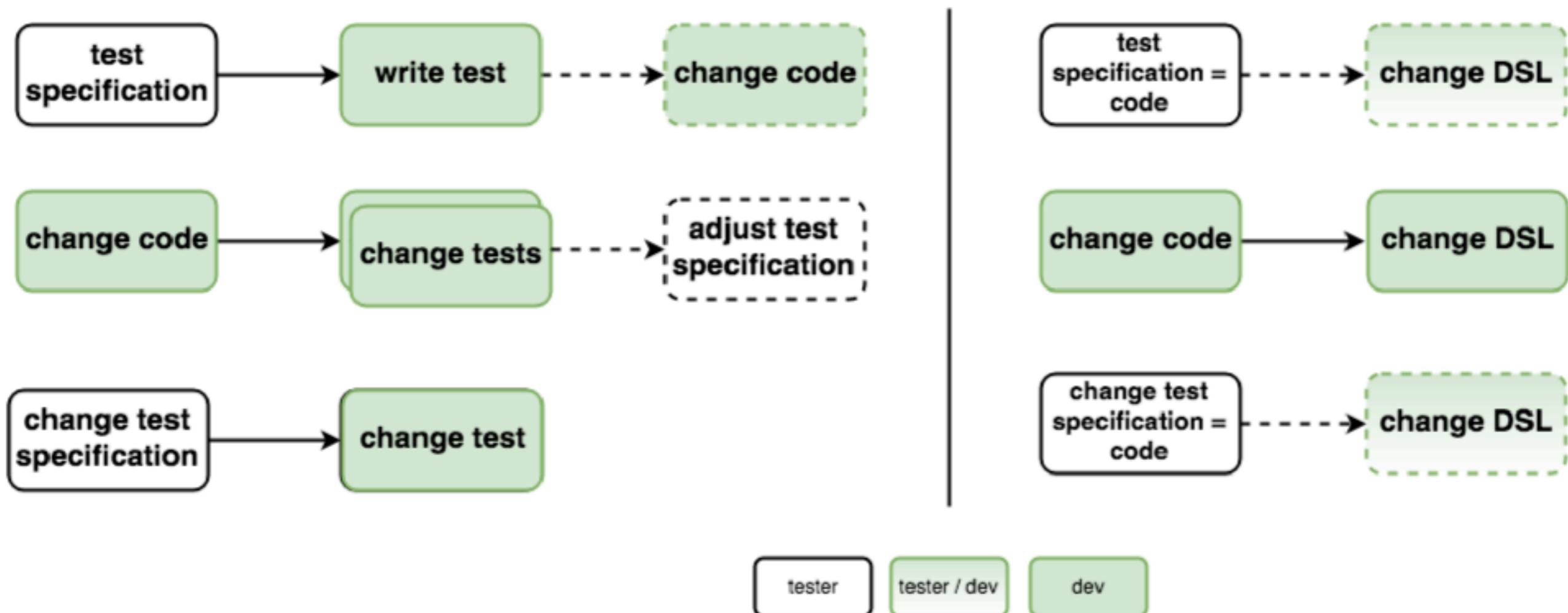


<https://proandroiddev.com/kotlin-using-test-robots-to-make-espresso-8cec2d746973>



Page Object / Robot pattern

Classic Espresso vs Robot DSL



<https://proandroiddev.com/kotlin-using-test-robots-to-make-espresso-8cec2d746973>



Easy to read and understand

Robot or Page Object pattern

```
@Test  
fun success_with_robot() {  
    robot  
        .setEmail("somkiat@xxx.com")  
        .setPassword("")  
        .clickLogin()  
        .mustShow( text: "Success")  
}
```



Create Login page

Create file LoginRobot.kt

```
class LoginRobot(val activity: Activity){  
  
    fun setEmail(email: String)  
        = apply { fillEditText(R.id.email, email) }  
    fun setPassword(password: String)  
        = apply { fillEditText(R.id.password, password) }  
    fun clickLogin() = apply { clickButton(R.id.email_sign_in_button) }  
    fun mustShow(text: String) = apply { toast(text) }  
}
```



Create Login page

Create file LoginRobot.kt

```
private fun fillEditText(resId: Int, text: String): ViewInteraction =  
    onView(withId(resId))  
        .perform(replaceText(text),  
                closeSoftKeyboard())  
  
private fun clickButton(resId: Int): ViewInteraction =  
    onView(withId(resId)).perform(click())  
  
private fun matchText(viewInteraction: ViewInteraction,  
                     text: String): ViewInteraction = viewInteraction  
    .check(ViewAssertions.matches(ViewMatchers.withText(text)))
```



2. Use library/DSL :: Kakao



<https://github.com/agoda-com/Kakao>



Add dependencies

In file /app/build.gradle

```
dependencies {  
    androidTestImplementation 'com.agoda.kakao:kakao:1.3.0'  
}
```



Easy to read and understand

```
fun success_with_kakao() {  
    screen {  
        email {  
            replaceText("somkiat@xxx.com")  
            closeSoftKeyboard()  
        }  
        password {  
            replaceText("")  
        }  
        login {  
            click()  
        }  
    }  
}
```



Login Screen

```
open class TestLoginScreen: Screen<TestLoginScreen>() {  
  
    val email: KEditText = KEditText { withId(R.id.email) }  
    val password: KEditText = KEditText { withId(R.id.password) }  
    val login: KButton = KButton { withId(R.id.email_sign_in_button) }  
  
}
```



More topics ...



Capture screenshot with Screengrab



<https://docs.fastlane.tools/getting-started/android/screenshots/>



1. Install Screengrab

```
$sudo gem install fastlane -NV
```

```
$sudo gem install screengrab
```



2. Add dependency

In file /app/build.gradle

```
dependencies {  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'com.android.support.test:runner:1.  
    androidTestImplementation 'com.android.support.test:rules:1.  
    androidTestImplementation 'com.android.support.test.espresso'  
  
    androidTestImplementation 'com.agoda.kakao:kakao:1.3.0'  
    androidTestImplementation 'tools.fastlane:screengrab:1.0.0'  
}
```



3. Create file AndroidManifest.xml

In folder **/app/src/debug/**

```
<!-- Allows unlocking your device and activating its screen so UI tests can succeed -->
<uses-permission android:name="android.permission.DISABLE_KEYGUARD"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>

<!-- Allows for storing and retrieving screenshots -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

<!-- Allows changing locales -->
<uses-permission android:name="android.permission.CHANGE_CONFIGURATION"
    tools:ignore="ProtectedPermissions" />
```



4. Add screengrab to UI test

In folder **/app/src/debug/**

```
class MainActivityUITest {  
  
    @get:Rule  
    val rule: ActivityTestRule<MainActivity>  
        = ActivityTestRule(MainActivity::class.java)  
  
    @get:Rule  
    val local: LocaleTestRule = LocaleTestRule();
```



4. Add screengrab to UI test

```
@Test  
fun success() {  
    Screengrab.screenshot( screenshotName: "step01" );  
  
    onView(withId(R.id.email)).perform(  
        replaceText( stringToBeSet: "somkiat@xxx.com" ),  
        closeSoftKeyboard() )  
  
    onView(withId(R.id.password)).perform(  
        replaceText( stringToBeSet: "" ),  
        closeSoftKeyboard() )  
  
    Screengrab.screenshot( screenshotName: "step02" );  
  
    onView(withId(R.id.email_sign_in_button)).perform(click())  
  
    Screengrab.screenshot( screenshotName: "step03" );  
  
    onView(withText( text: "Success" ))  
        .inRoot(withDecorView(not(rule.activity.window.decorView)))  
        .check(matches(isDisplayed()))  
}
```



5. Run

```
$gradlew assembleDebug assembleAndroidTest  
$fastlane screengrab
```

```
[✓] 🚀  
[07:19:15]: Get started using a Gemfile for fastlane https://ls/getting-started/ios/setup/#use-a-gemfile  
[07:19:15]: To not be asked about this value, you can specify 'package_name'  
[07:19:15]: The package name of the app under test (e.g. com.yourcompany.yourapp): █
```



6. Try to config screengrabfile

\$screengrab init

```
app_package_name('demo.loginapplication')
app_apk_path('app/build/outputs/apk/debug/app-debug.apk')
tests_apk_path('app/build/outputs/apk/androidTest/debug/app-debug-androidTest.apk')
locales(['en-US', 'fr-FR', 'it-IT'])
# clear all previously generated screenshots in your local output directory before c
clear_previous_screenshots(true)
```

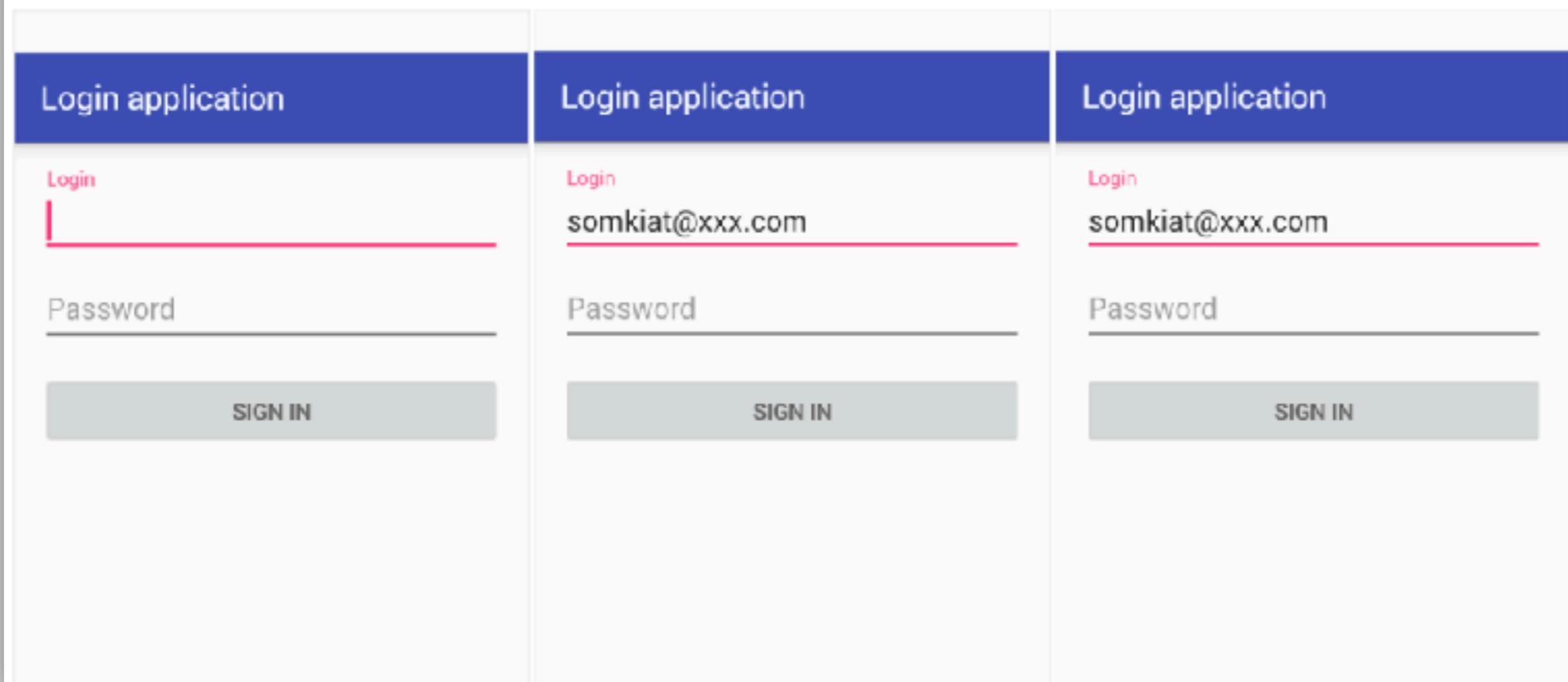


7. See result

In folder **/fastlane**

en-US

phoneScreenshots



Problem with Screengrab

Version of screen grab for Android N+

Solution :: reduce version of screengrab

