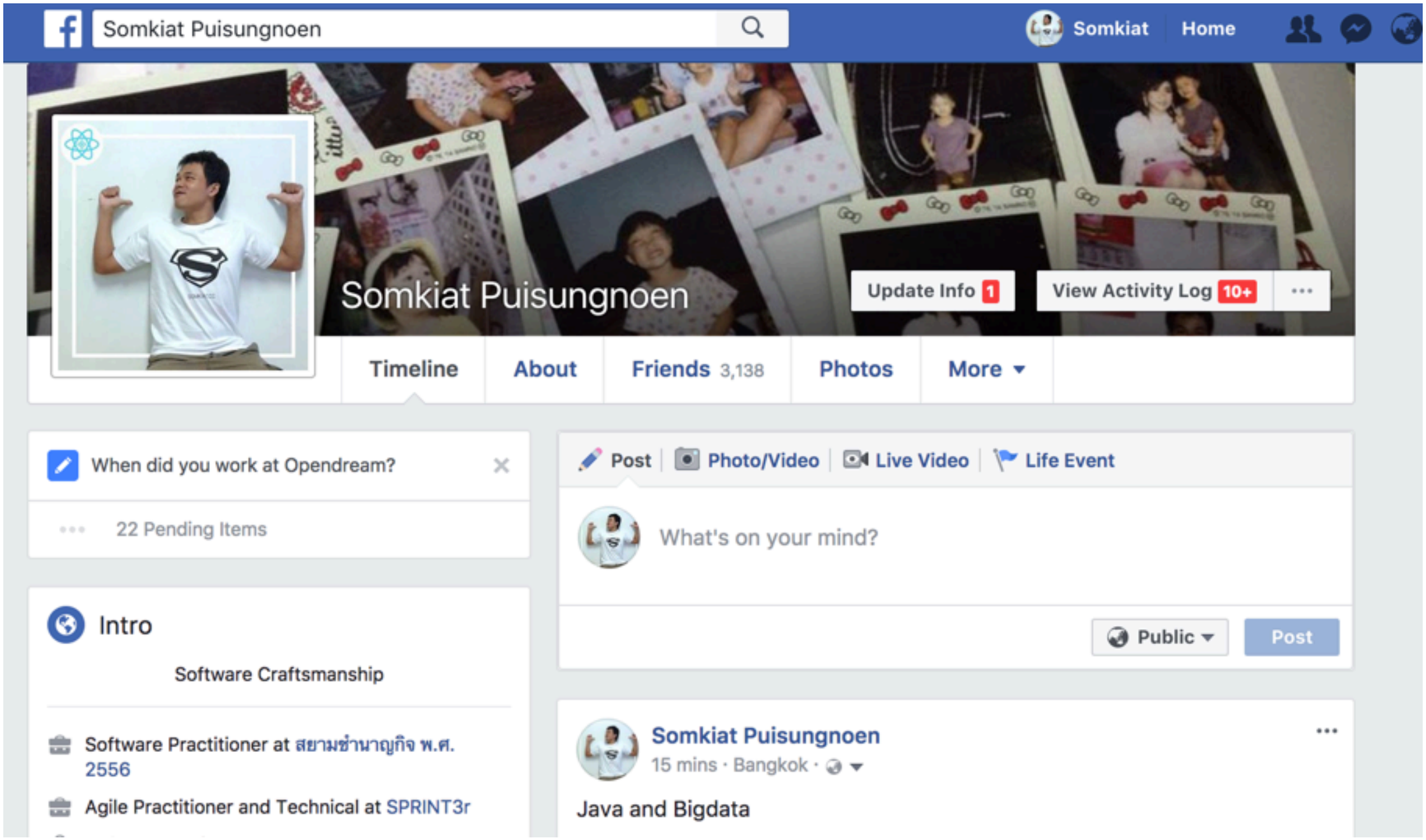
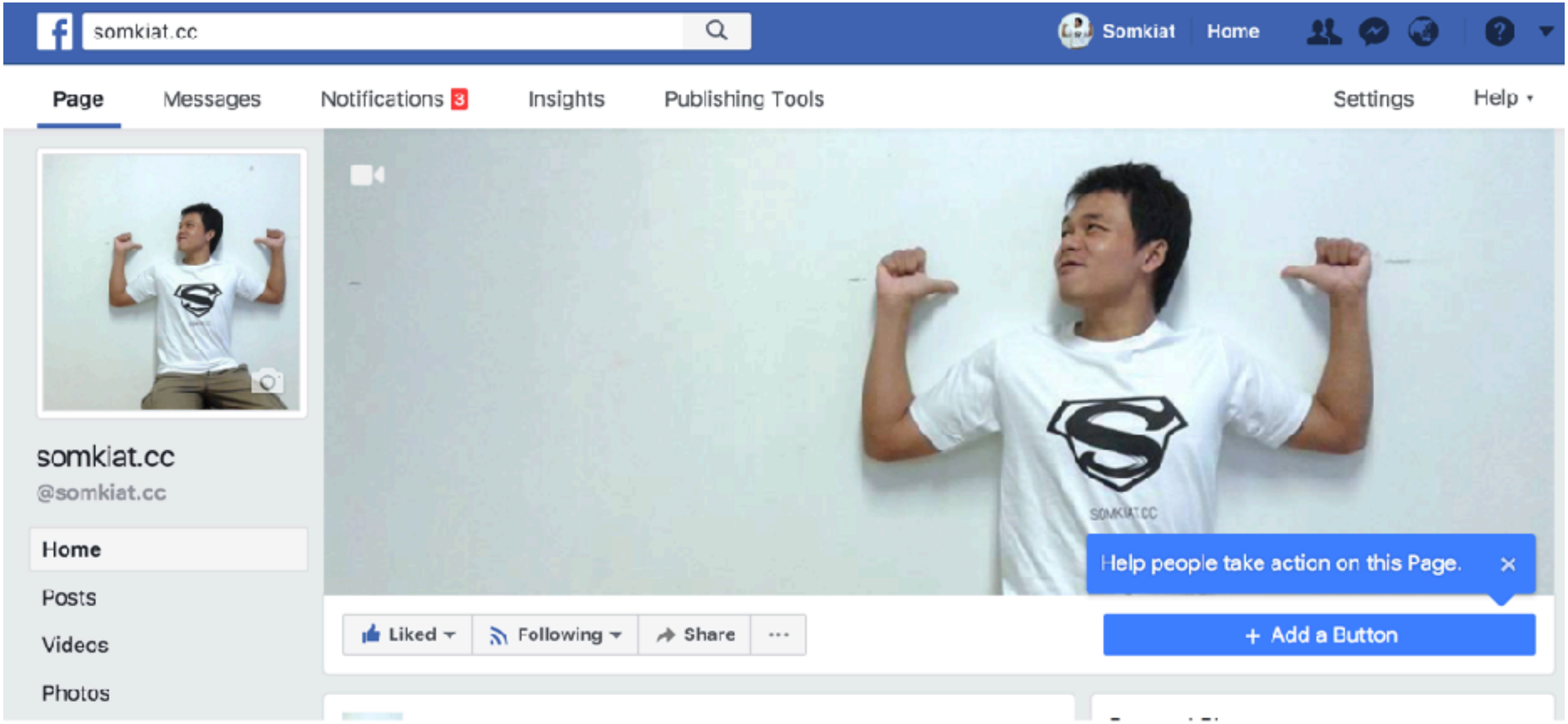
สยามชำนาญกิจ
SIAM CHAMNAN KIT

Effective Android Testing





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
- Expresso 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 ?

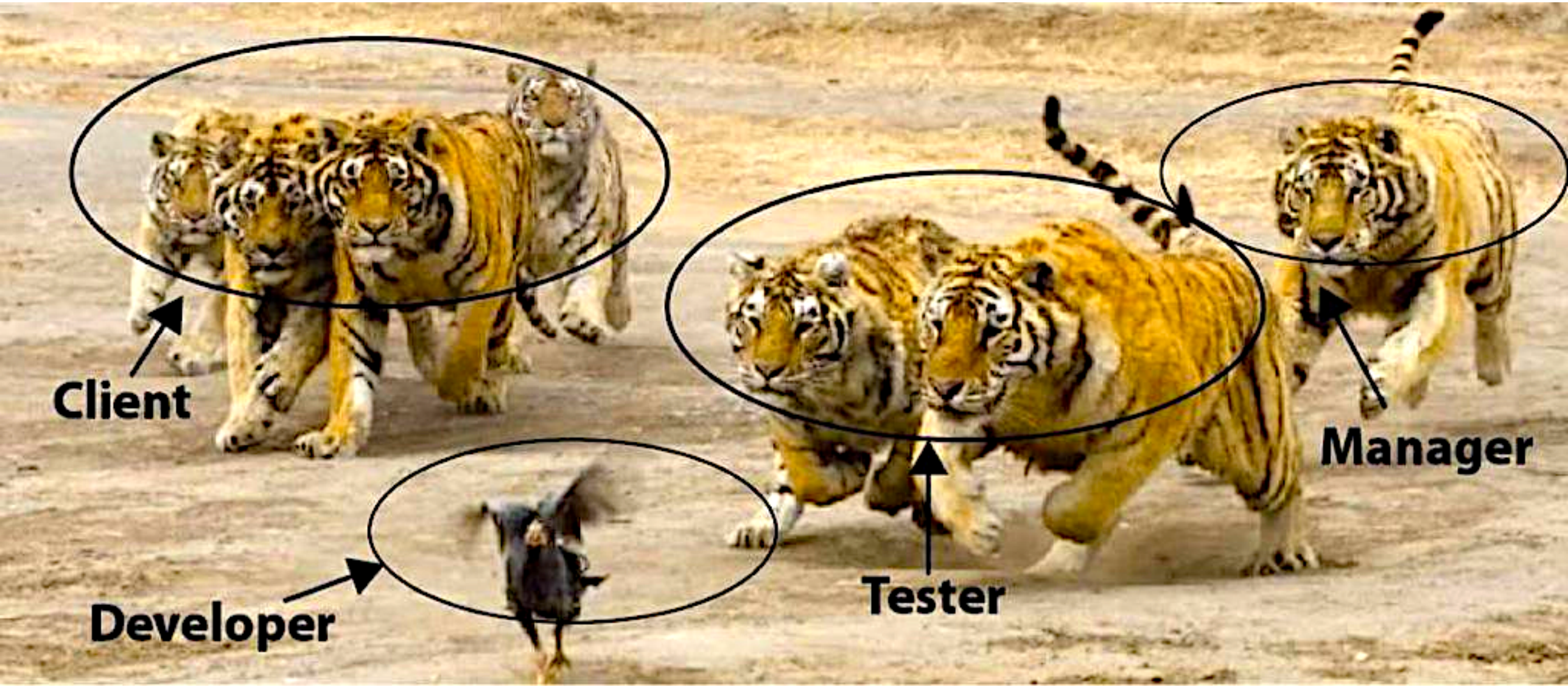
Java/Android/Kotlin

Android Studio

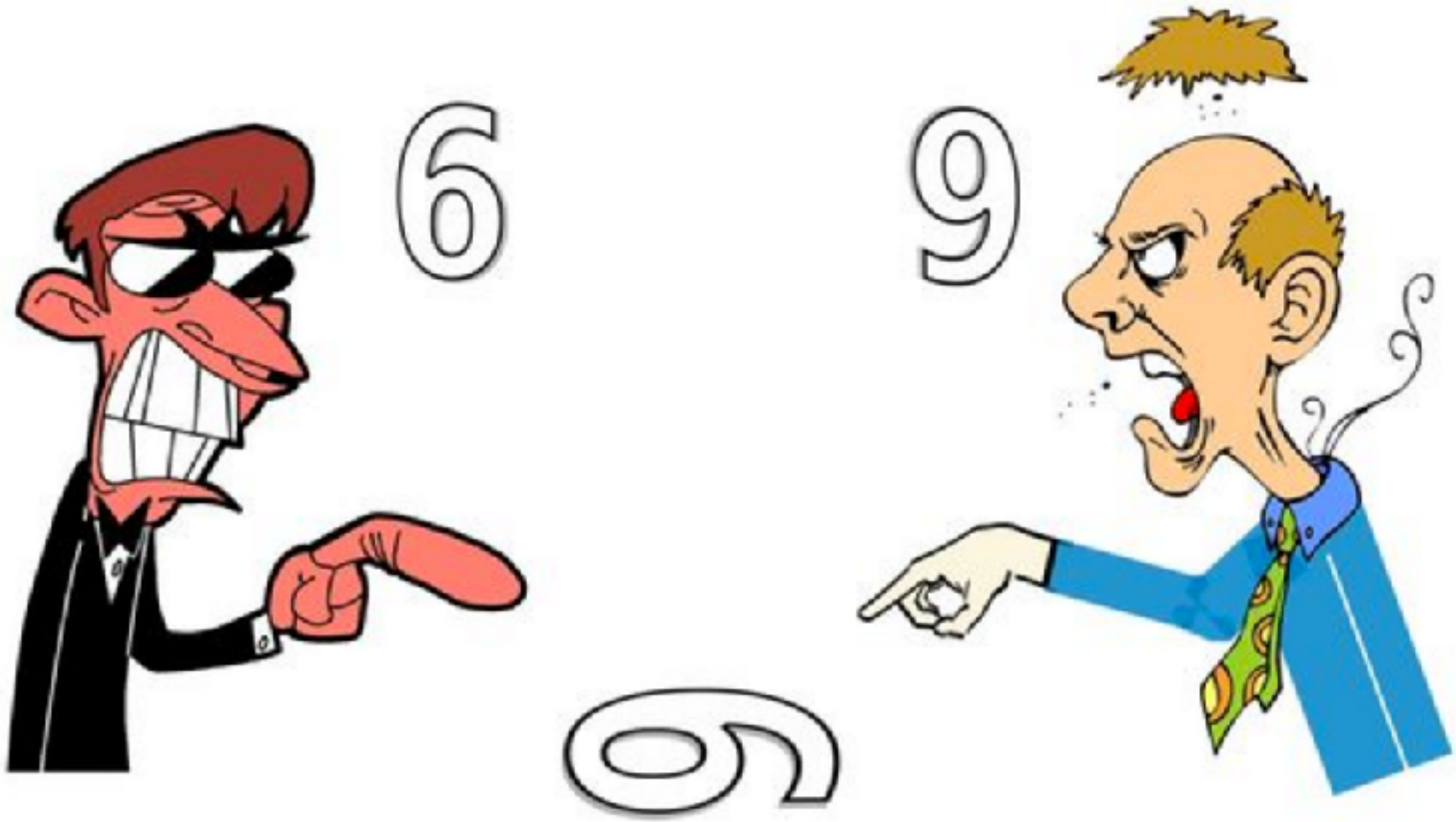
JUnit 4

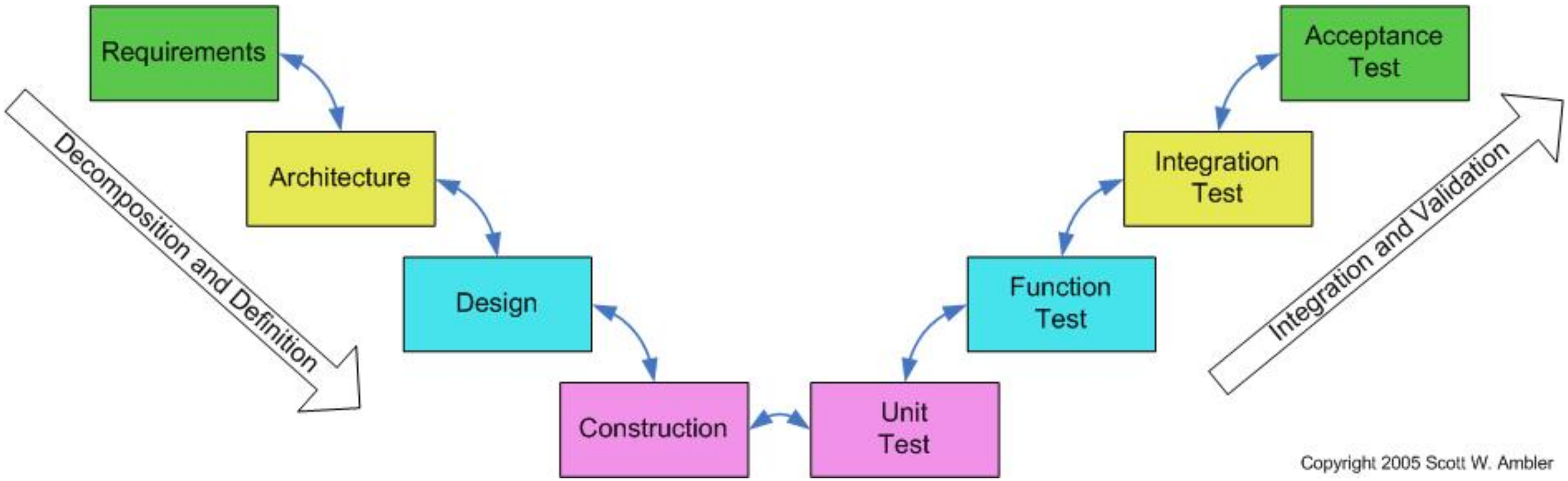
Espresso/Kakao

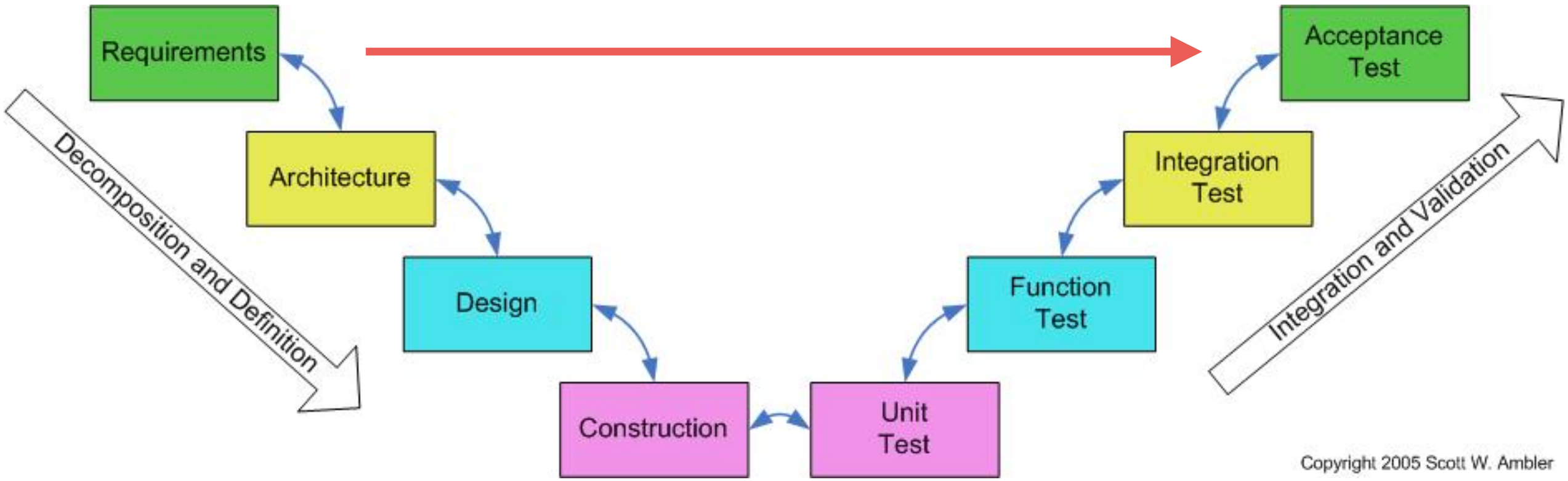
All about Testing

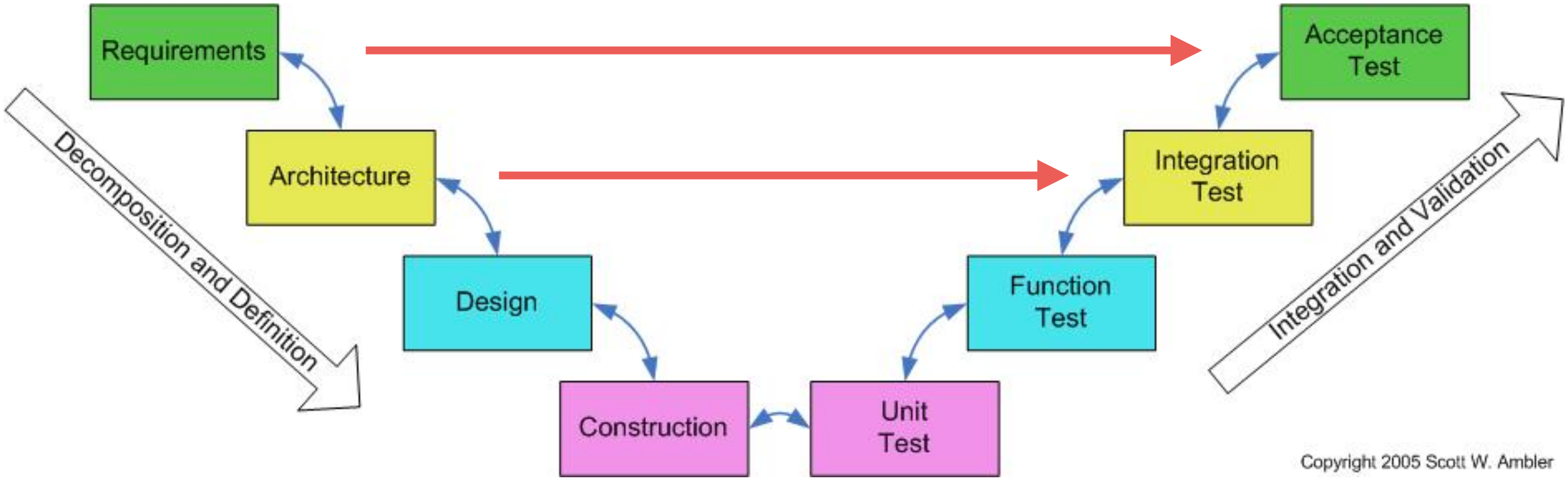


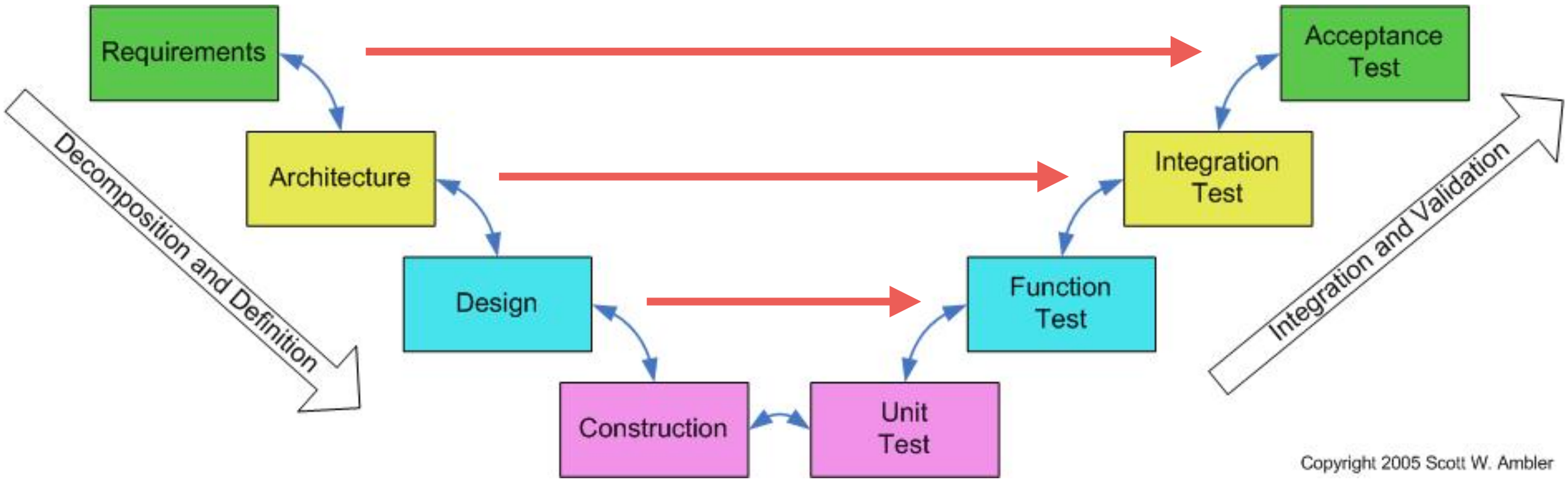
Developer vs Tester

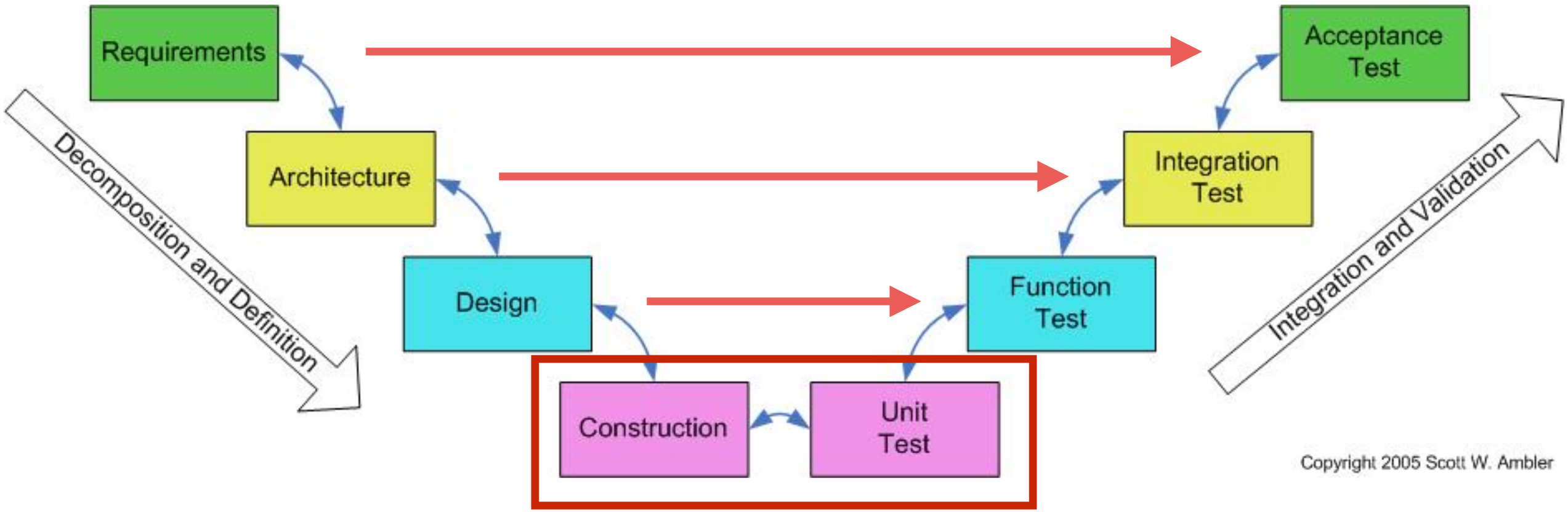






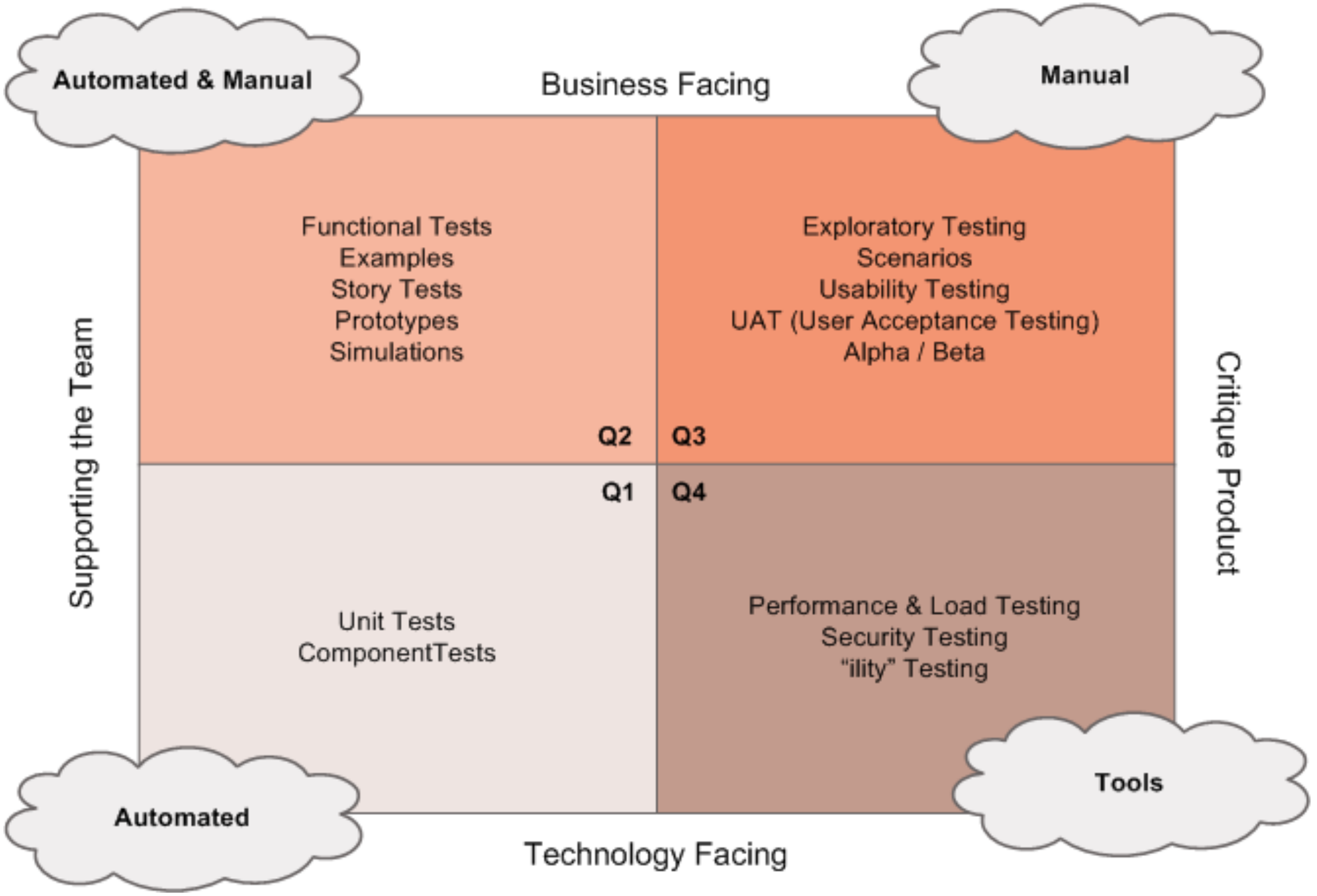






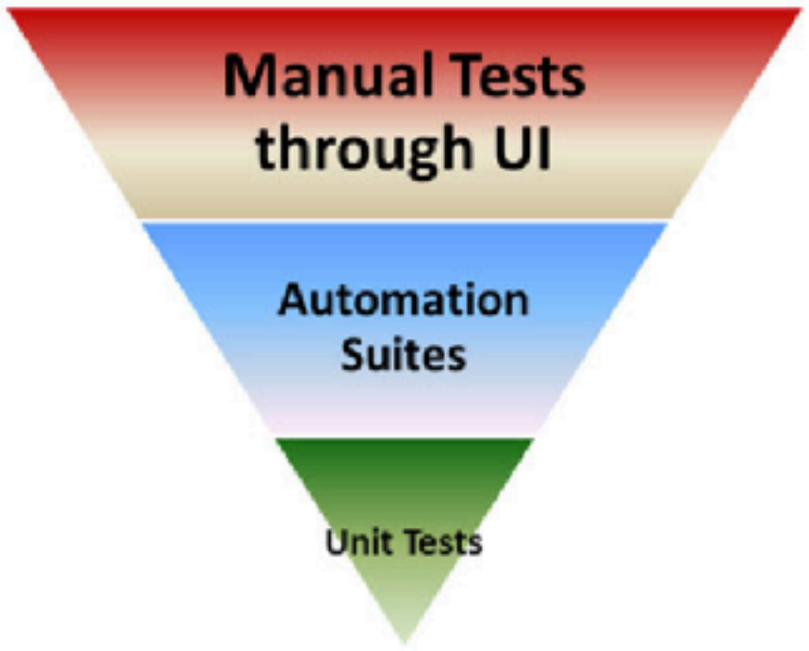
Type of testing

Agile Testing Quadrants

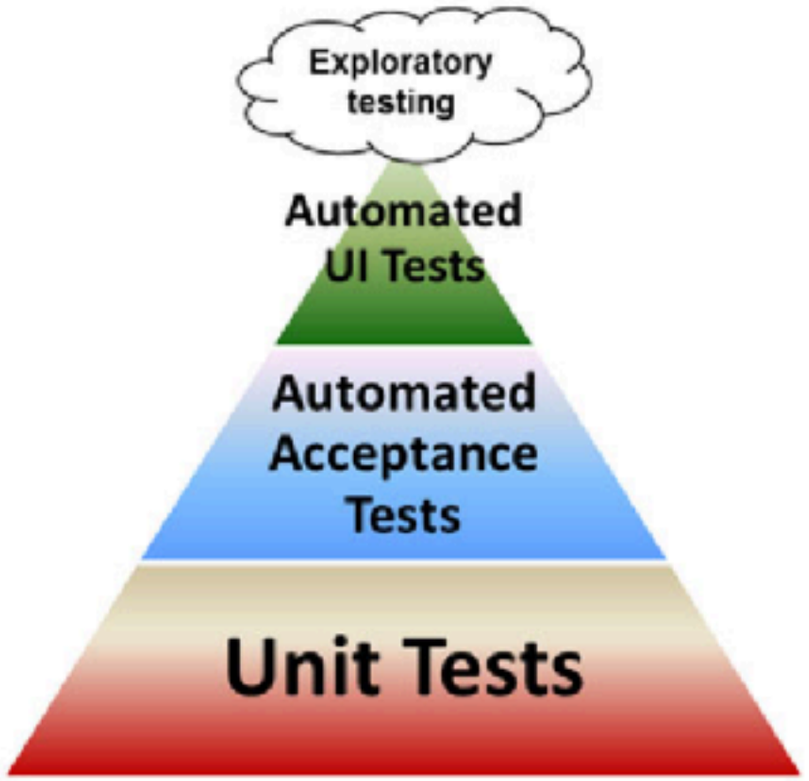


Testing pyramid

Testing Pyramid

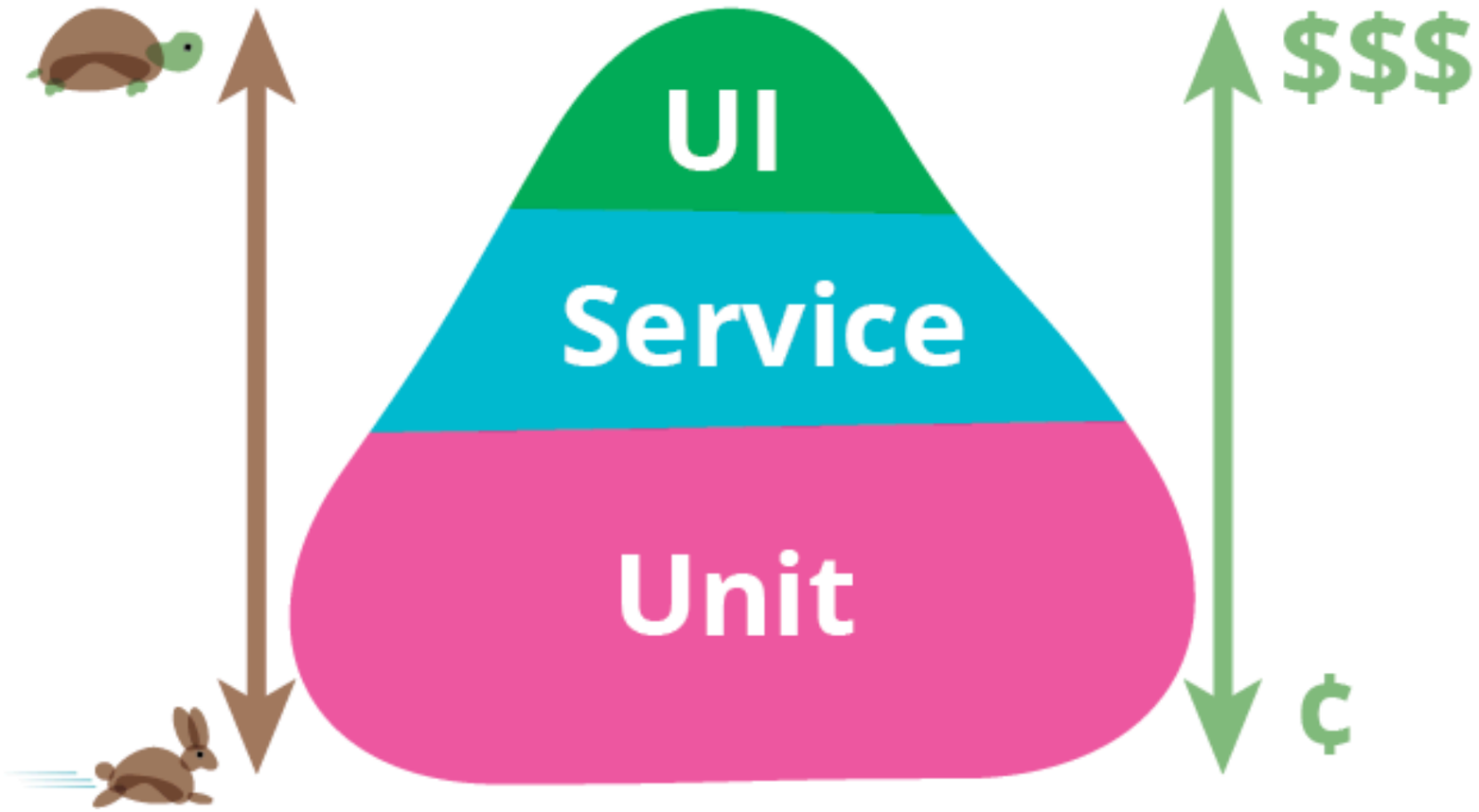


Traditional
(find bugs)



Agile
(prevent bugs)

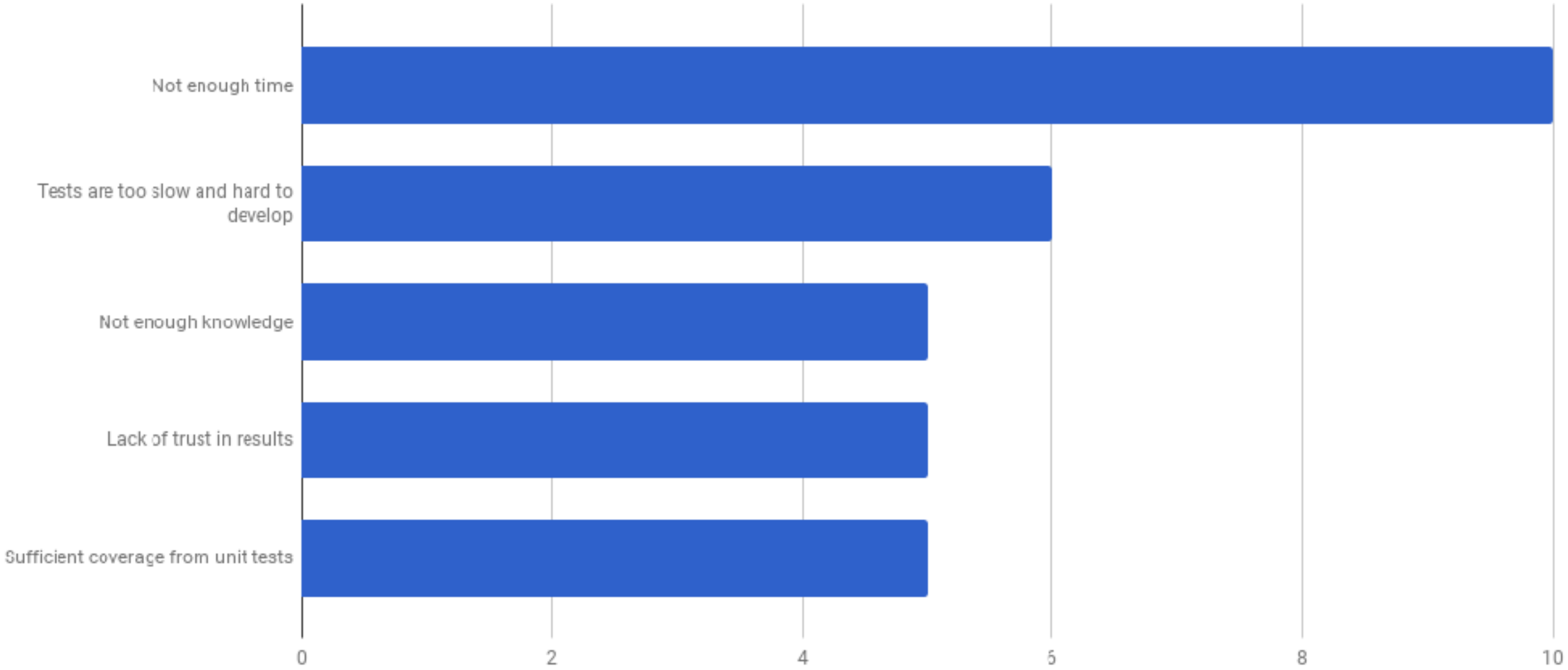
Testing Pyramid



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

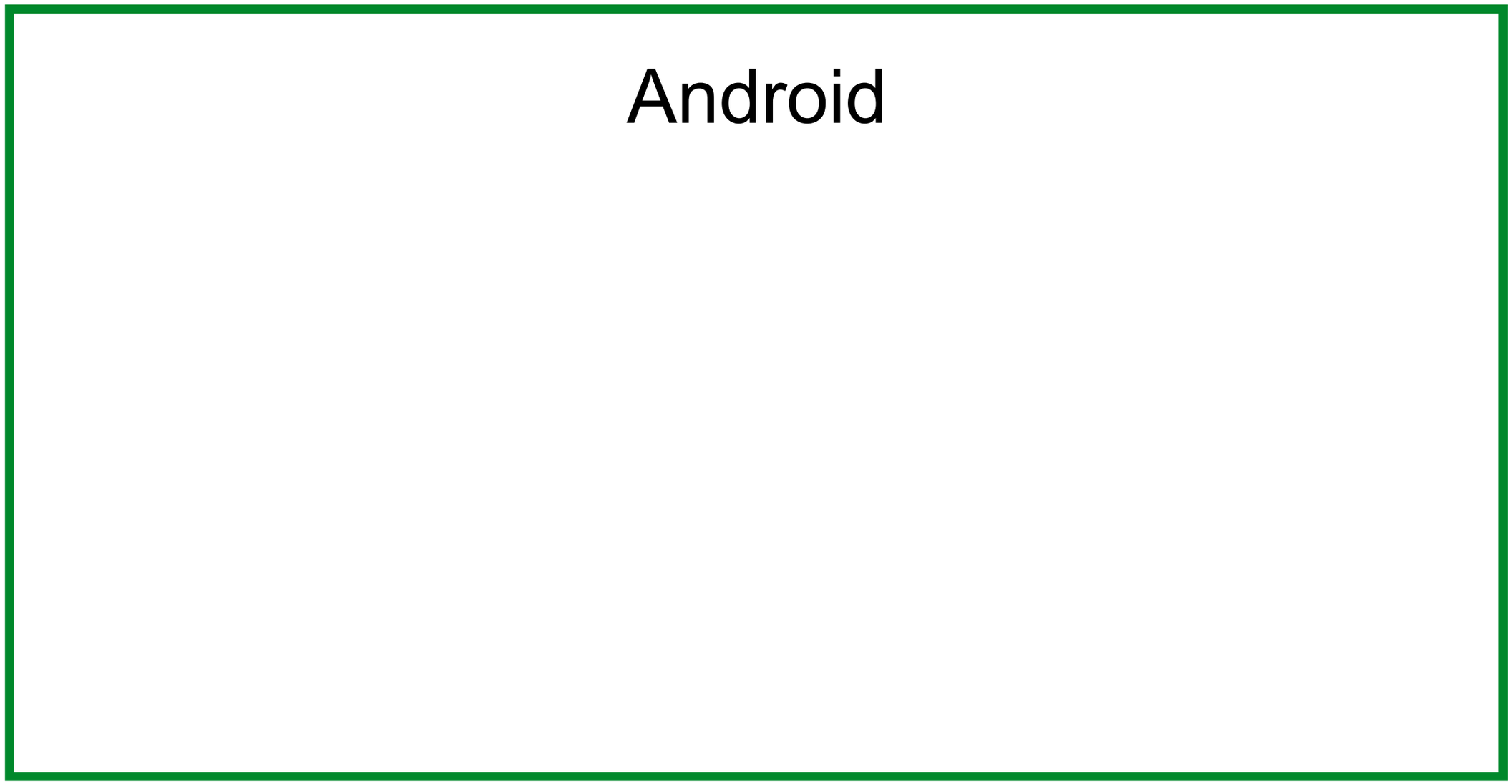
Testing not a phase
Testing is activity

Why not write automated tests ?

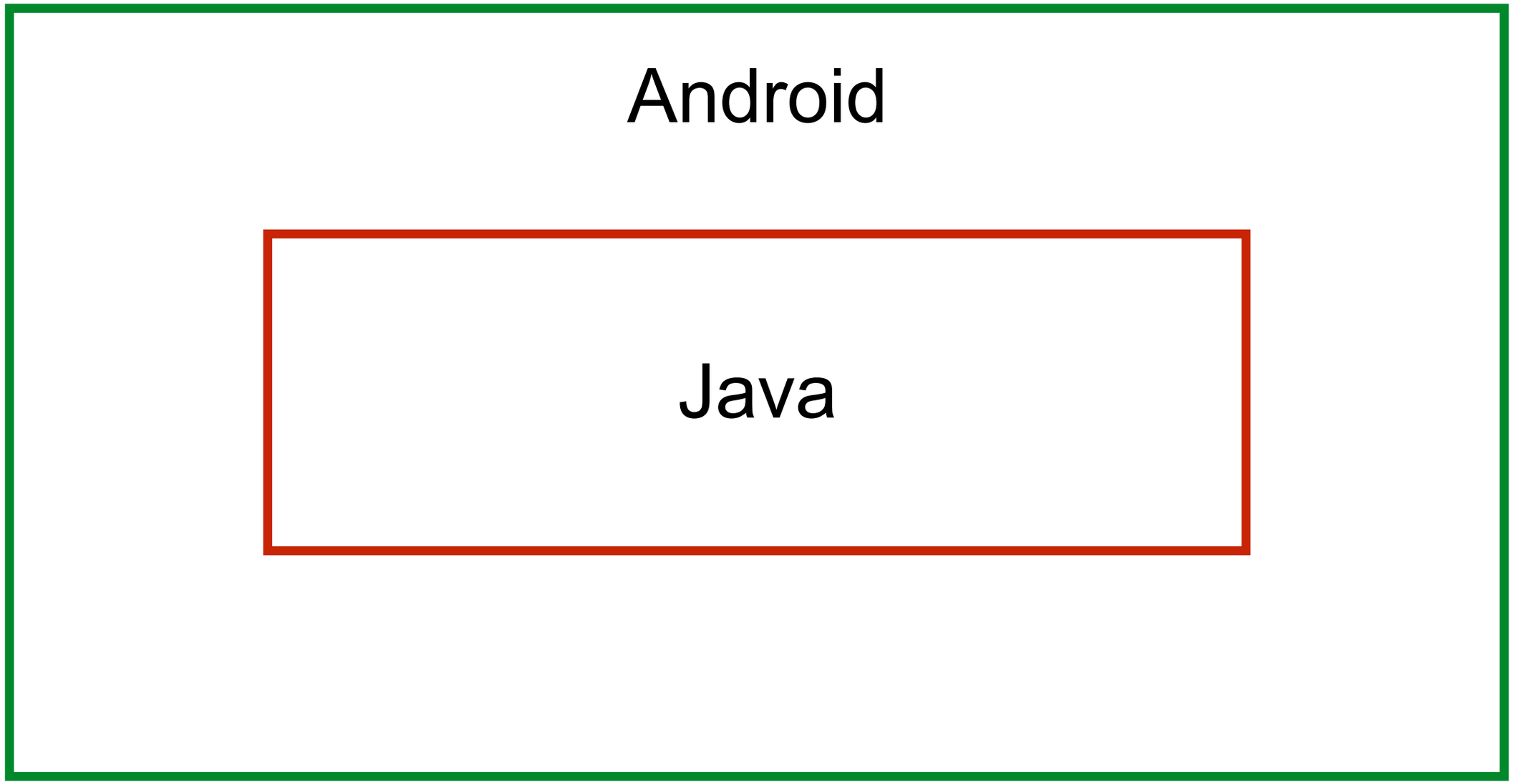


https://slack.engineering/android-ui-automation-part-1-building-trust-de3deb1c5995

Android Testing



Android Testing



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

	JVM	Device
Non-UI	JVM unit test	Instrumentation unit test
UI		

/src/test /src/androidTest

Instrumentation UI test

	JVM	Device
Non-UI	JVM unit test	Instrumentation unit test
UI		Instrumentation UI test

/src/test

/src/androidTest

Android Testing

	JVM	Device
Non-UI	JVM unit test	Instrumentation unit test
UI		Instrumentation UI test

/src/test /src/androidTest

UI test on JVM ?

	JVM	Device
Non-UI	JVM unit test	Instrumentation unit test
UI	? ? ?	Instrumentation UI test

/src/test

/src/androidTest

Android Testing

	JVM	Device
Non-UI	JVM unit test	Instrumentation unit test
UI	Robolectric	Instrumentation UI test

/src/test

/src/androidTest

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.

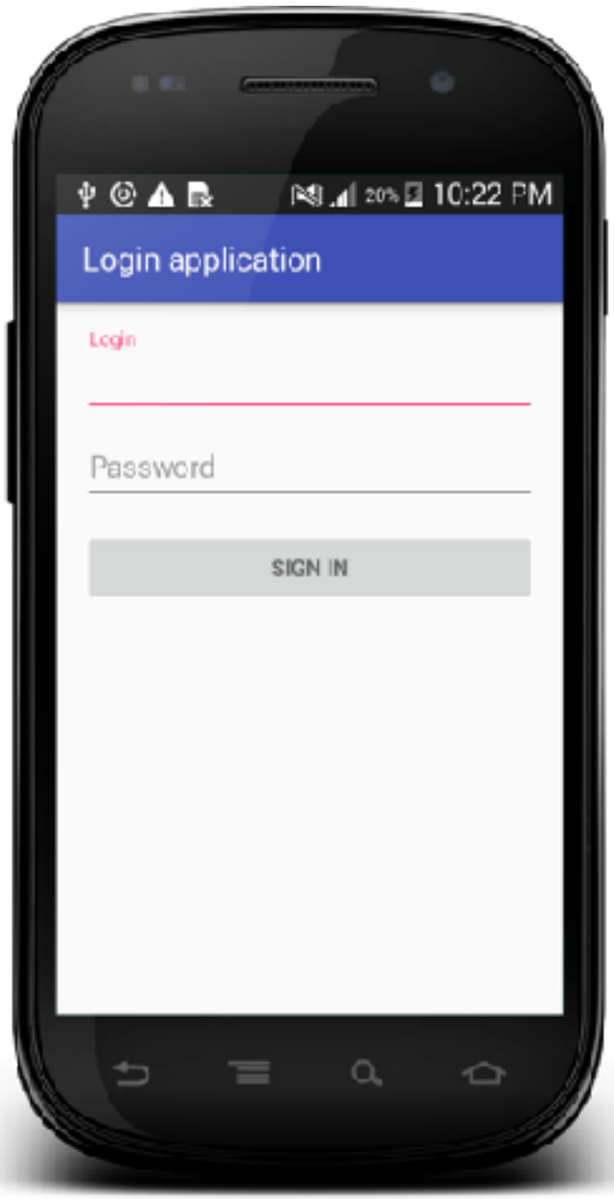
Try to separate the standard Java code from Android-dependent code.

Workshop with Testing

Hello Android Testing

Login application

Login Page



Result

Success

Failure

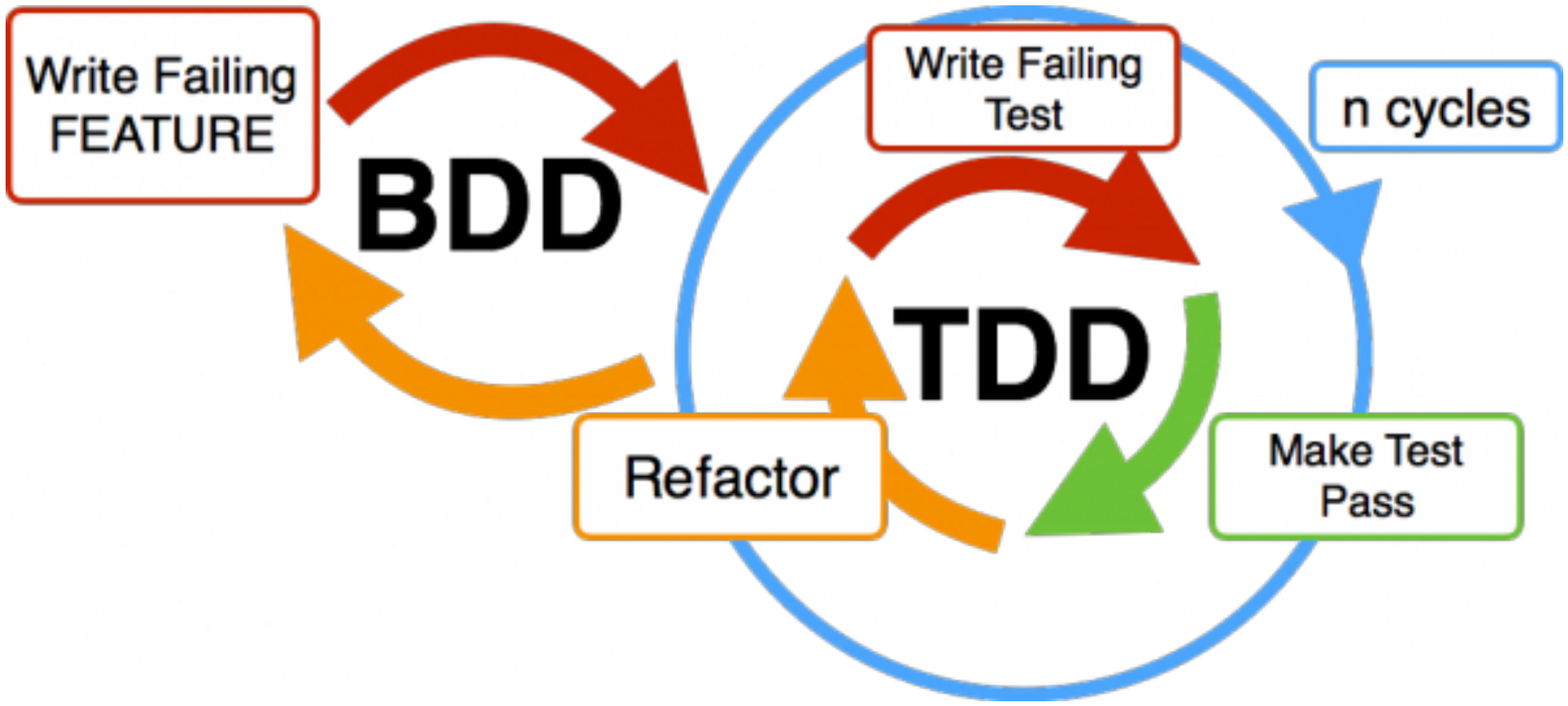
Write test case ?

Test cases

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

Start to write UI testing

Using espresso



Let’ start to write UI tests

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/

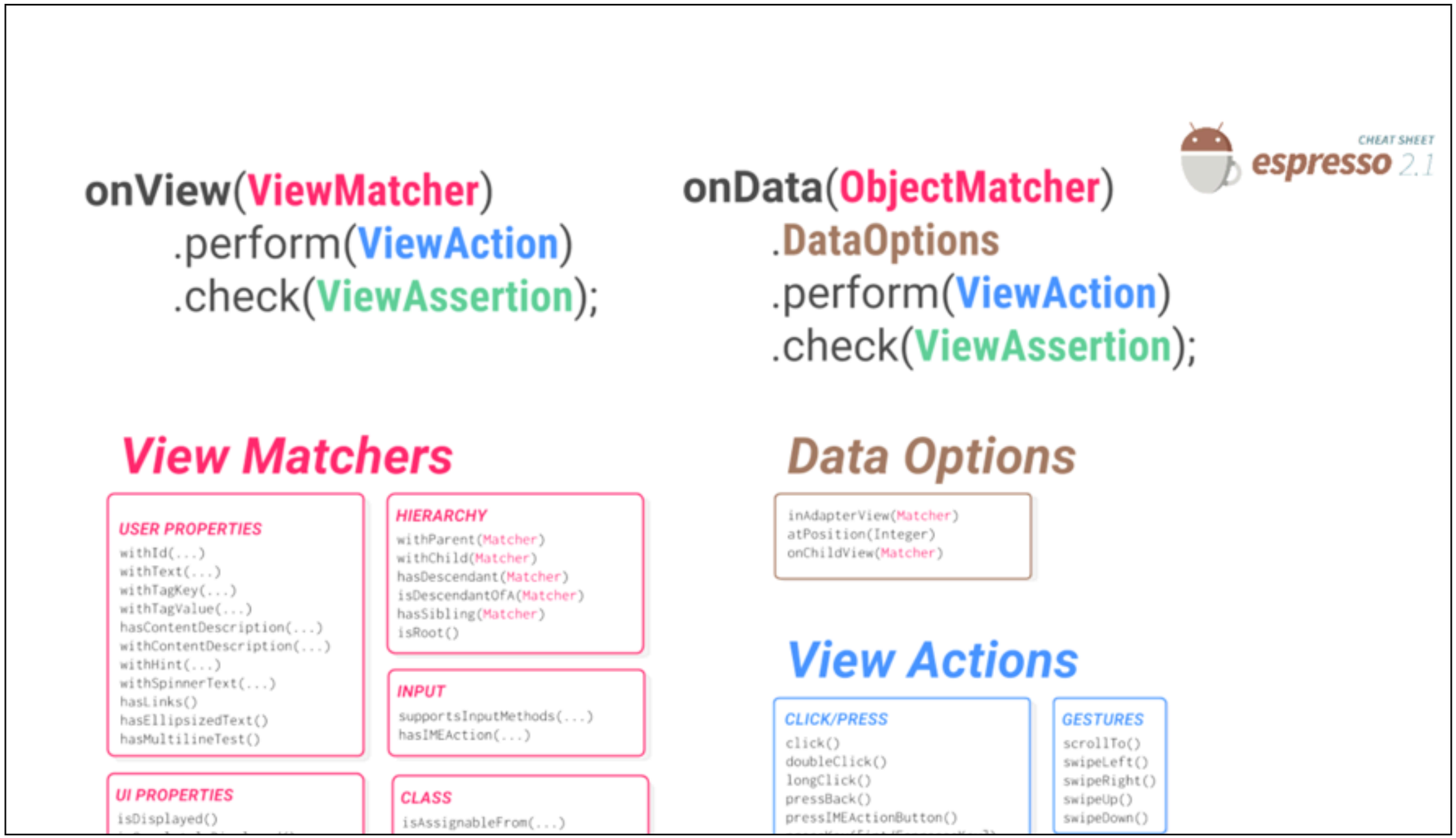
Why use Espresso ?

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

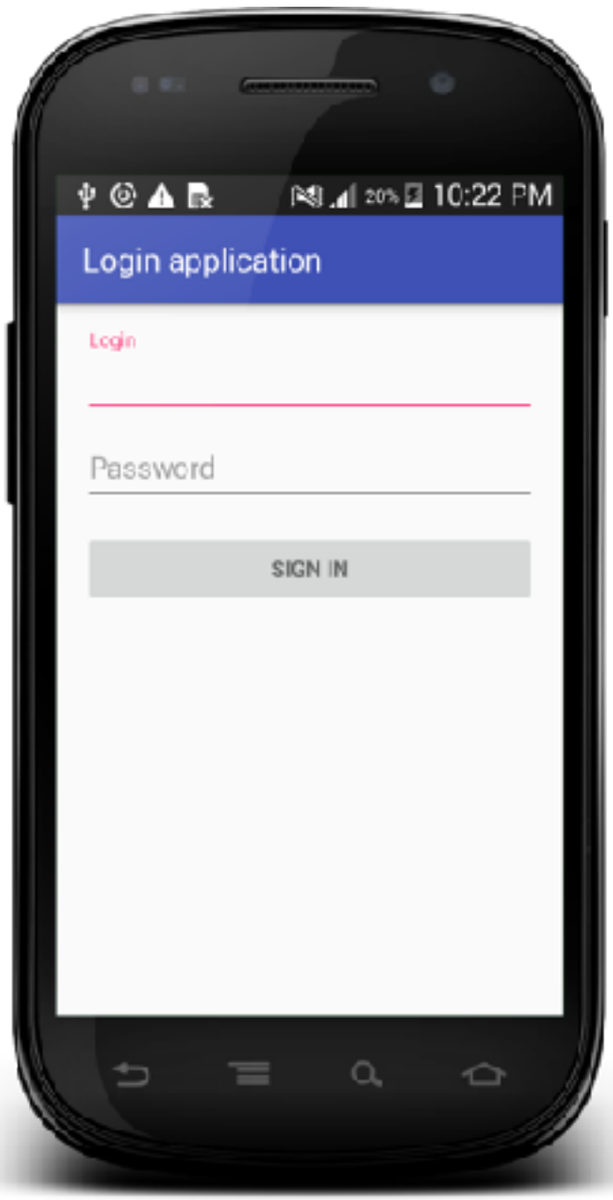


https://developer.android.com/training/testing/espresso/cheat-sheet

Let’ start to write tests

Flow 1 :: Login Success

Login Page

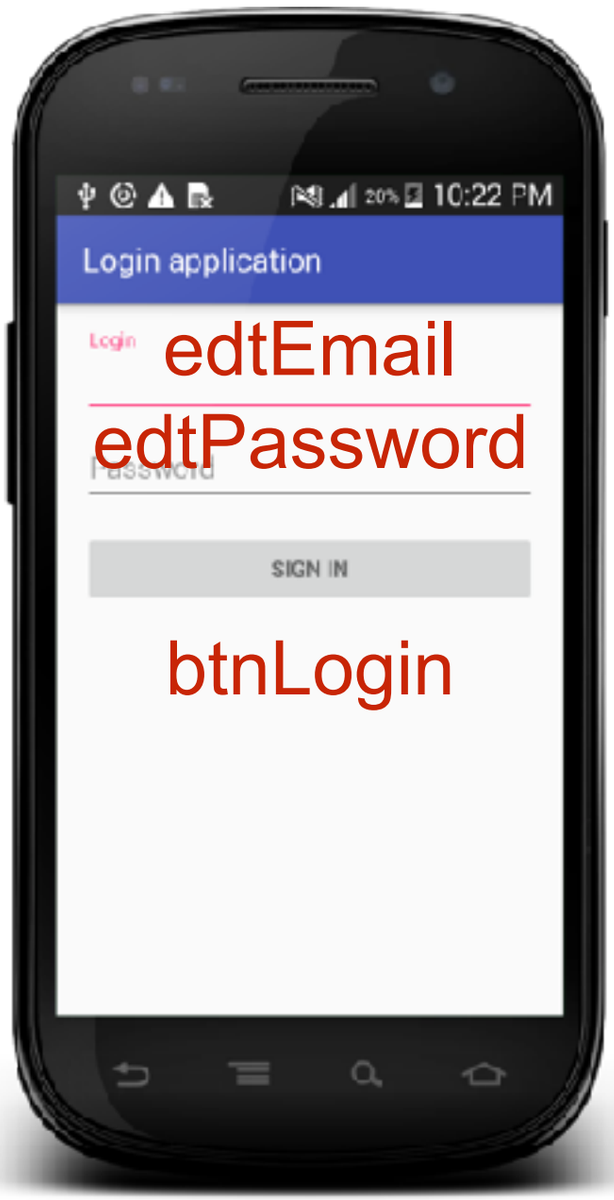


Result Page

Success

Define ID in each element

Login Page



Result Page

txtResult

Success

Create test case of Android Test

/app/src/androidTest/java
Filename = MainActivityTest.java

How to start/open MainActivity ?

Using ActivityTestRule class from Espresso

```
public class MainActivityTest {

    @Rule
    public ActivityTestRule<MainActivity> mainPage =
        new ActivityTestRule<>(MainActivity.class);
```

Issue

ActivityTestRule class not found

Solution :: Add new dependency in file /app/build.gradle

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0-rc01'
    implementation 'com.android.support.constraint:constraint-layout:1.1.2'
    testImplementation 'junit:junit:4.12'
    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'
```

Issue

Conflict version of support-annotation

Solution :: Exclude in /app/build.gradle

```
dependencies {
    androidTestImplementation('com.android.support.test.espresso:espresso-core:3.0.2') {
        exclude group: 'com.android.support', module: 'support-annotations'
    }
    androidTestImplementation('com.android.support.test:runner:1.0.2') {
        exclude group: 'com.android.support', module: 'support-annotations'
    }
```

Write first test case

Login success with correct data

```
@Test
public void login_สำเร็จ() {
    onView(withId(R.id.edtEmail))
            .perform(
                    typeText("somkiat@gmail.com"),
                    closeSoftKeyboard());
    onView(withId(R.id.edtPassword))
            .perform(
                    typeText("12345"),
                    closeSoftKeyboard()
            );
    onView(withId(R.id.btnLogin))
            .perform(click());

    onView(withId(R.id.txtResult))
            .check(matches(withText("Success")));
}
```

Run test in Android Studio

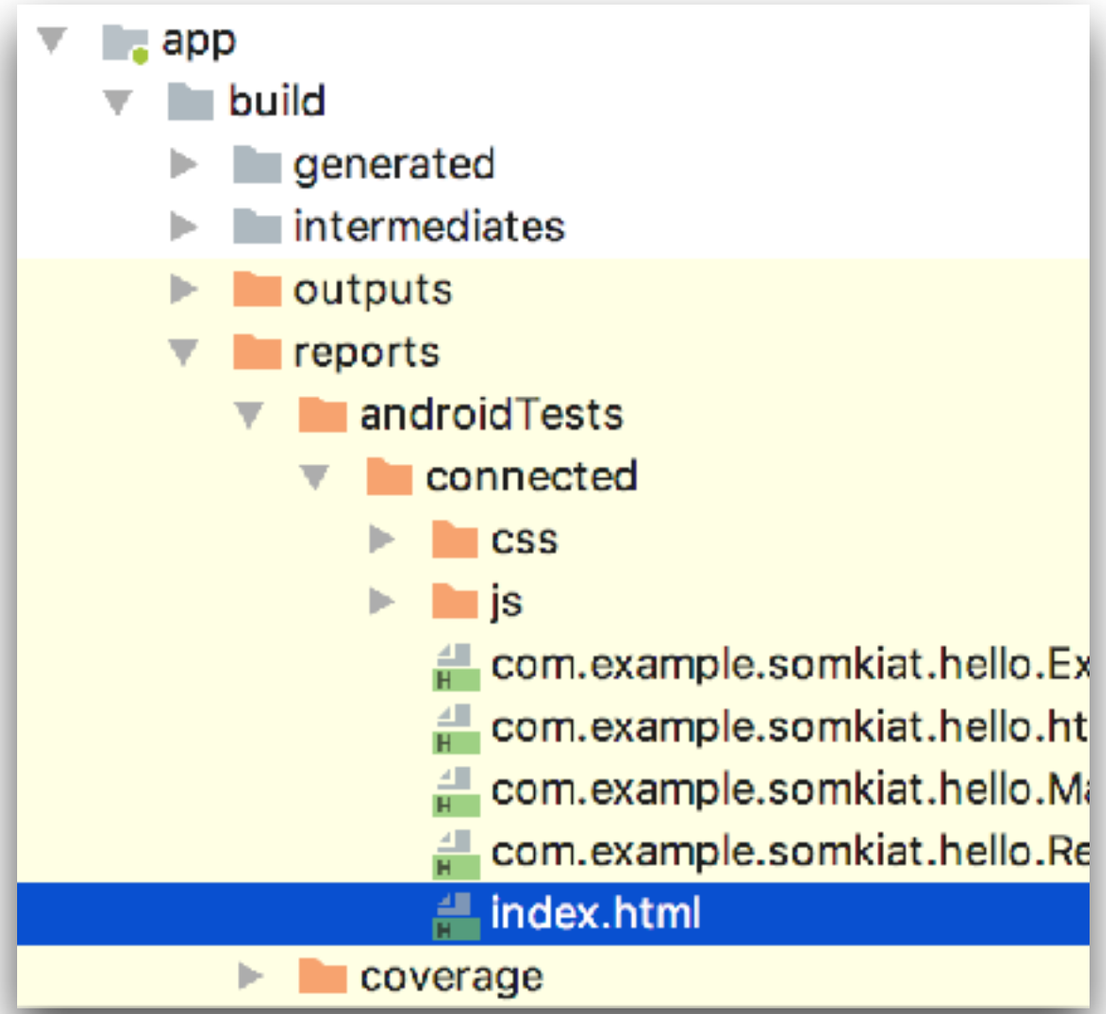
Run test in command line

$gradlew connectedAndroidTest
$gradlew cAT

```
> Task :app:connectedDebugAndroidTest
Starting 5 tests on Nexus_5X_API_28(AVD) - 9


BUILD SUCCESSFUL in 1m 6s
53 actionable tasks: 8 executed, 45 up-to-date
```

Test result of Android Testing

/app/build/reports/androidTests/connected



Test result of Android Testing

/app/build/reports/androidTests/connected

Test Summary

5	0	22.055s
tests	failures	duration

100%
successful

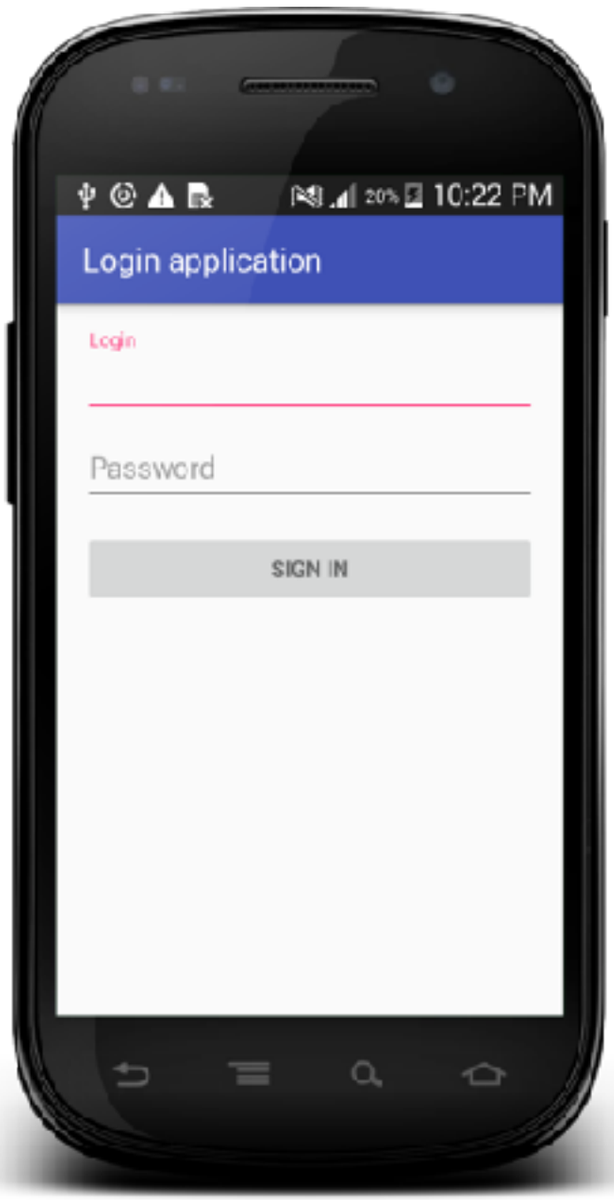
Packages Classes

Package	Tests	Failures	Duration	Success rate
com.example.somkiat.hello	5	0	22.055s	100%

Flow 2 :: Login Fail

Login Page

Result Page



Failure

Write second test case

Login file with wrong email

```
@Test
public void login_ไม่สำเร็จในกรณี_email_ไม่พบในระบบ() {
    onView(withId(R.id.edtEmail))
            .perform(
                    typeText("somkiat1@gmail.com"),
                    closeSoftKeyboard());
    onView(withId(R.id.edtPassword))
            .perform(
                    typeText("12345"),
                    closeSoftKeyboard()
            );
    onView(withId(R.id.btnLogin))
            .perform(click());

    onView(withId(R.id.txtResult))
            .check(matches(withText("BAD")));
}
```

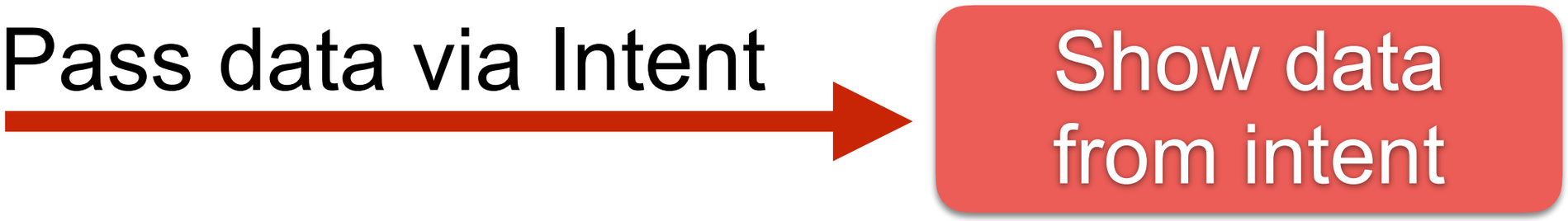
Write third test case

Login file with wrong password

```
@Test
public void login_ไม่สำเร็จในกรณี_password_ไม่พบในระบบ() {
    onView(withId(R.id.edtEmail))
            .perform(
                    typeText("somkiat@gmail.com"),
                    closeSoftKeyboard());
    onView(withId(R.id.edtPassword))
            .perform(
                    typeText("123456"),
                    closeSoftKeyboard()
            );
    onView(withId(R.id.btnLogin))
            .perform(click());

    onView(withId(R.id.txtResult))
            .check(matches(withText("BAD")));
}
```

Write test cases of ResultActivity

ResultActivity



How to start ResultActivity ?

Start activity without open !!

```
public class ResultActivityTest {

    @Rule
    public ActivityTestRule<ResultActivity> resultPage =
            new ActivityTestRule<>(
                    ResultActivity.class, true, false);
```

flag to launch activity

Setup data for intent

```
@Test
public void แสดงค่า_Success() {
    Intent intent = new Intent();
    intent.putExtra("result", "Success");

    resultPage.launchActivity(intent);

    onView(withId(R.id.txtResult))
            .check(matches(withText("Success")));
}
```

Launch activity with intent

```
@Test
public void แสดงค่า_Success() {
    Intent intent = new Intent();
    intent.putExtra("result", "Success");

    resultPage.launchActivity(intent);

    onView(withId(R.id.txtResult))
            .check(matches(withText("Success")));
}
```

Check result in current page

```
@Test
public void แสดงค่า_Success() {
    Intent intent = new Intent();
    intent.putExtra("result", "Success");

    resultPage.launchActivity(intent);

    onView(withId(R.id.txtResult))
            .check(matches(withText("Success")));
}
```

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

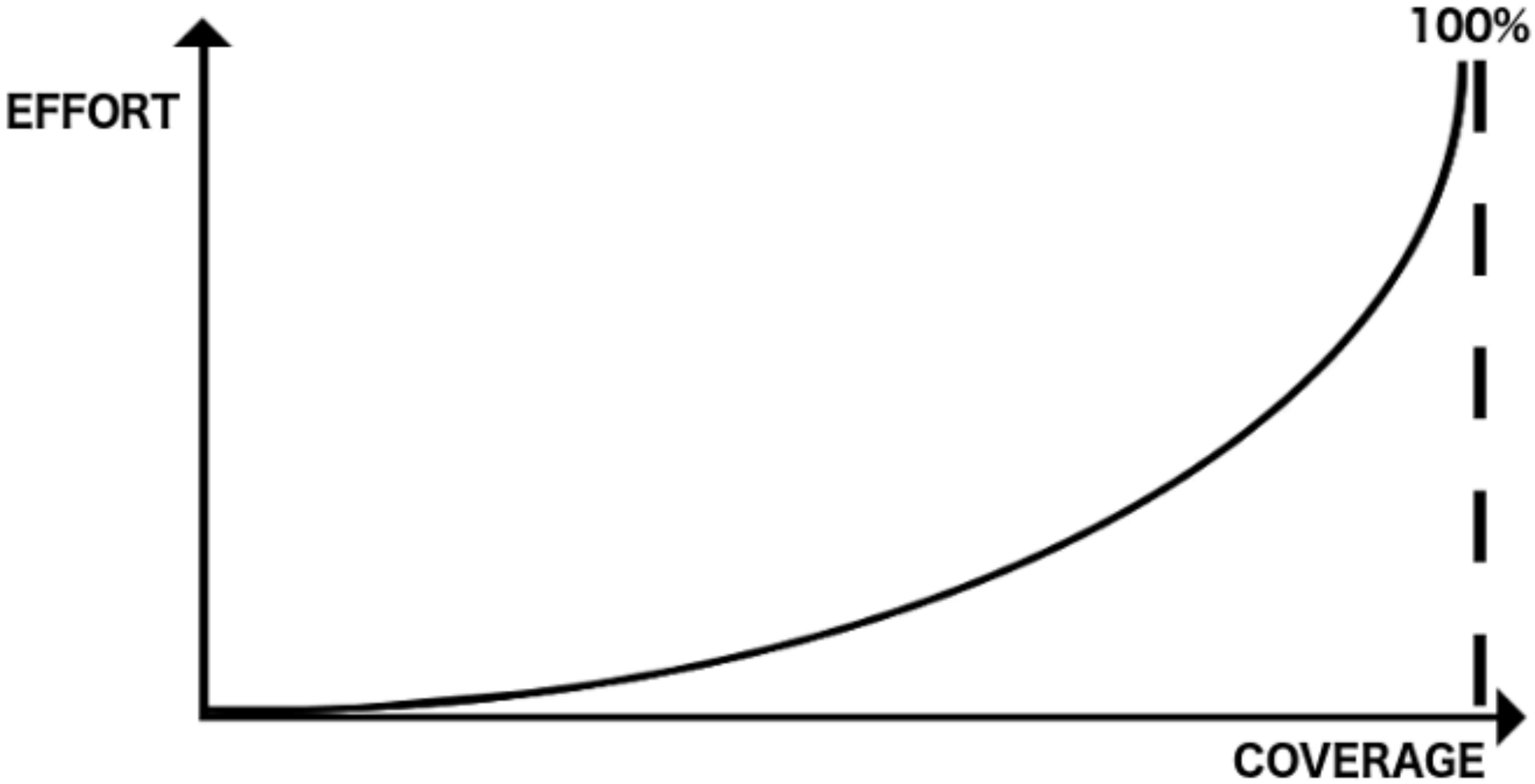
But 100% of code coverage **does not mean** that your code is 100% correct

Code coverage

Powerful tool to improve the quality of your code

Code coverage != quality of tests

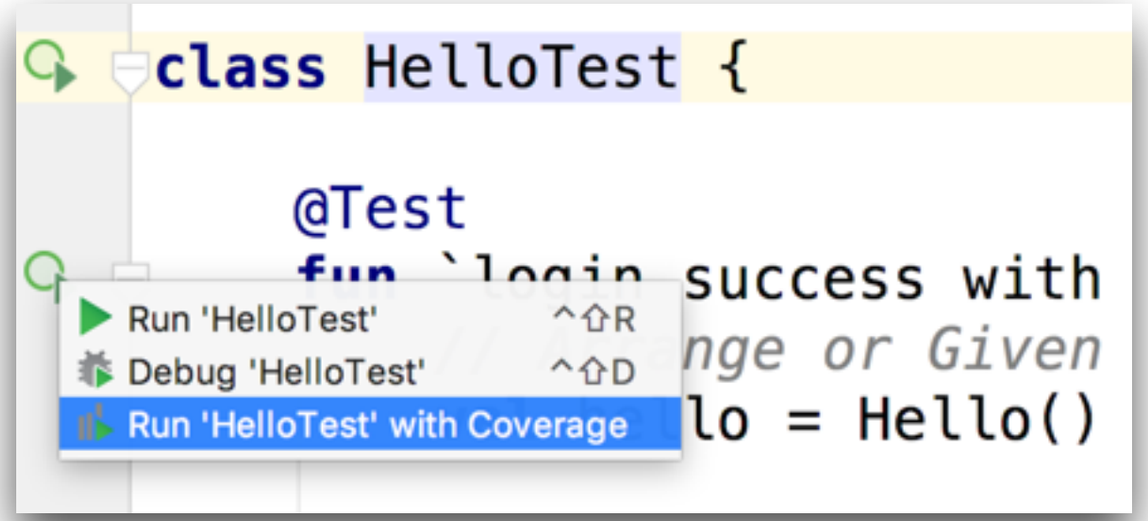
Code coverage 100% ?



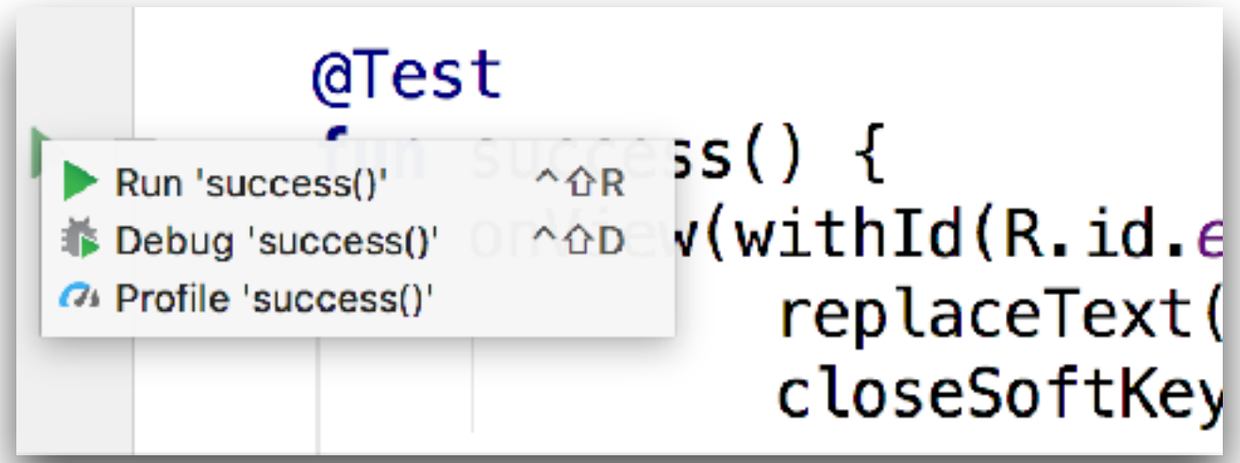
Code coverage for android

Disabled by default !!

Unit test



androidTest



Enable code coverage

1. Enable coverage in debug

In file /app/build.gradle

```
buildTypes {
    debug {
        testCoverageEnabled true
    }
    release {
        minifyEnabled false
        proguardFiles getDefaultProg
    }
}
```

2. Use jacobo library

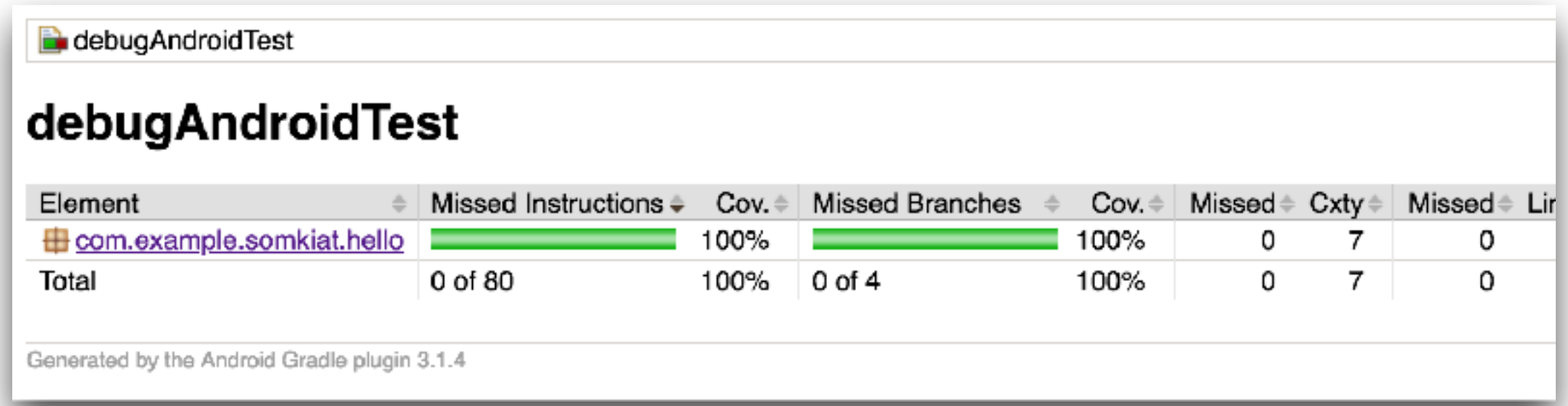
In file /app/build.gradle

```
apply plugin: 'jacoco'

jacoco {
    toolVersion = "0.8.1"
}
```

3. Run code coverage of Android Test

$gradlew createDebugCoverageReport

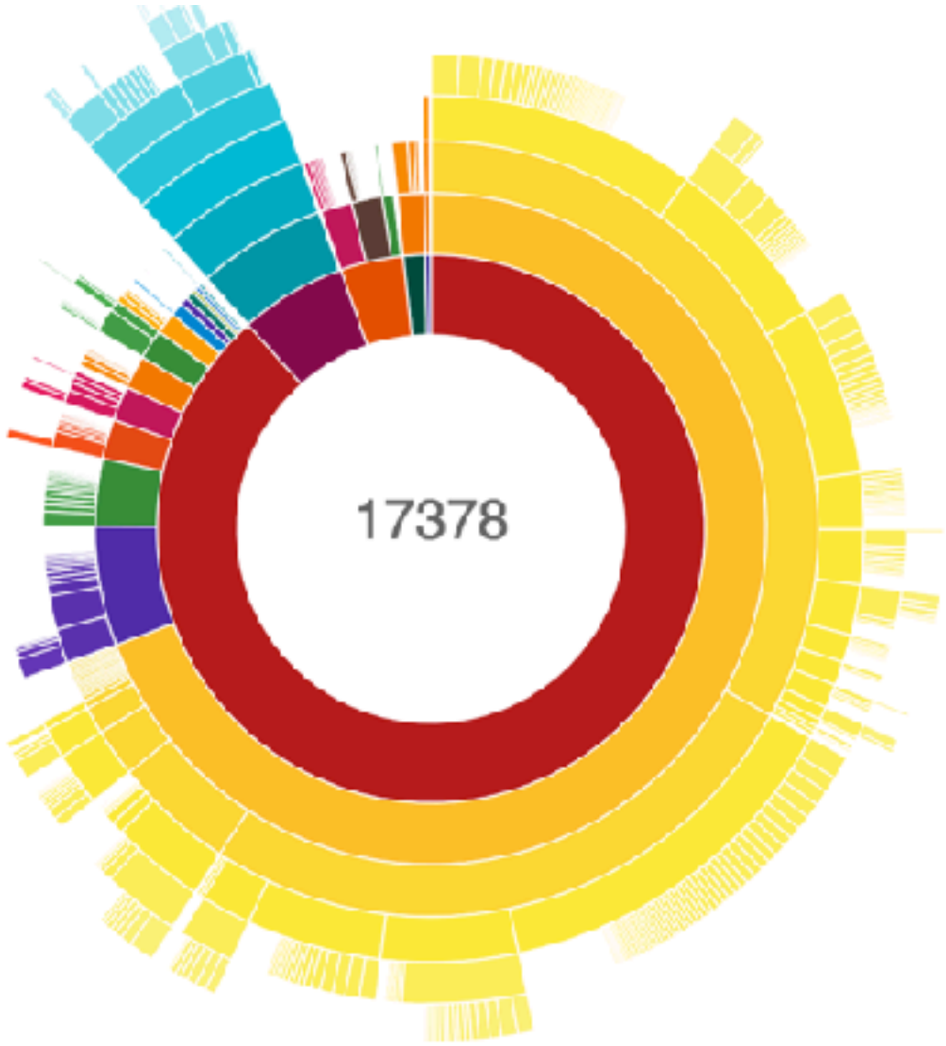
Result in /app/build/reports/coverage/debug/index.html



Tips and Tricks

Count number of methods (65K)

Using dexcount plugin



https://github.com/KeepSafe/dexcount-gradle-plugin

Using monkey test (Stress testing)

$**adb** shell monkey -p your.package.name -v 500

Number of events for testing

https://developer.android.com/studio/test/monkey

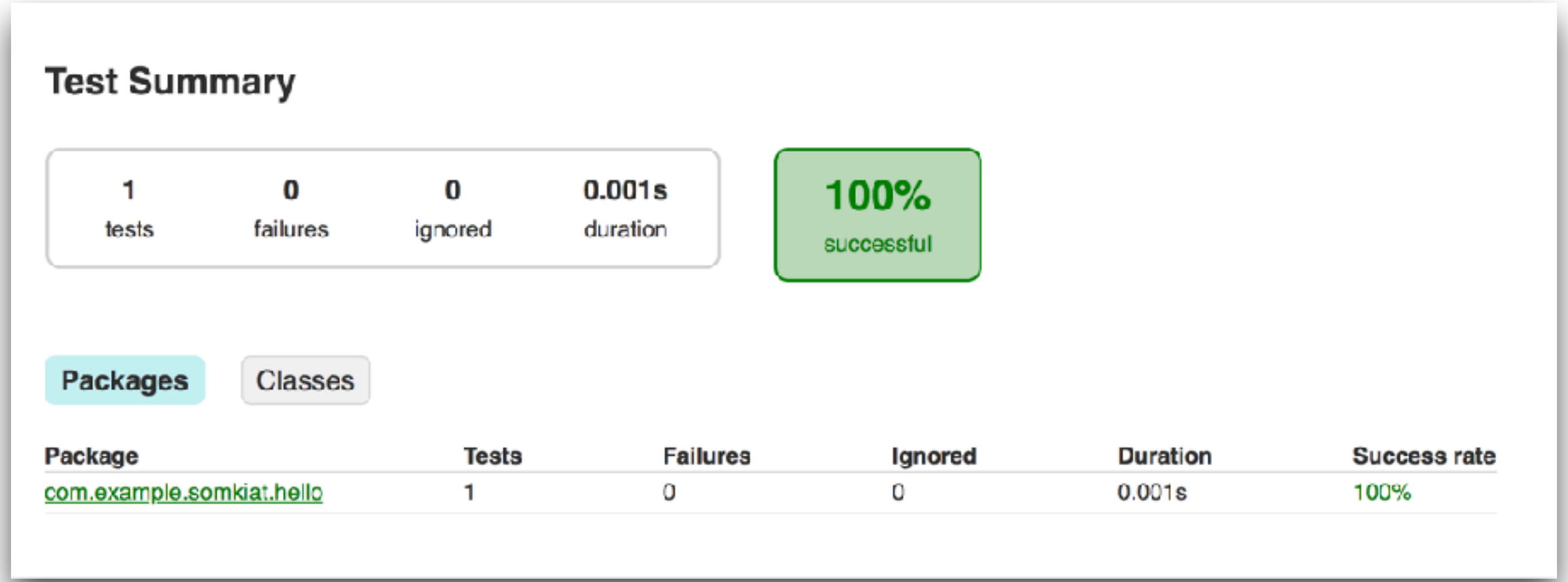
Run unit testing in command line

$gradlew testDebugUnitTest
$gradlew tDUT

https://developer.android.com/studio/test/command-line

Run unit testing in command line

See result in /app/build/reports/tests/testDebugUnitTest

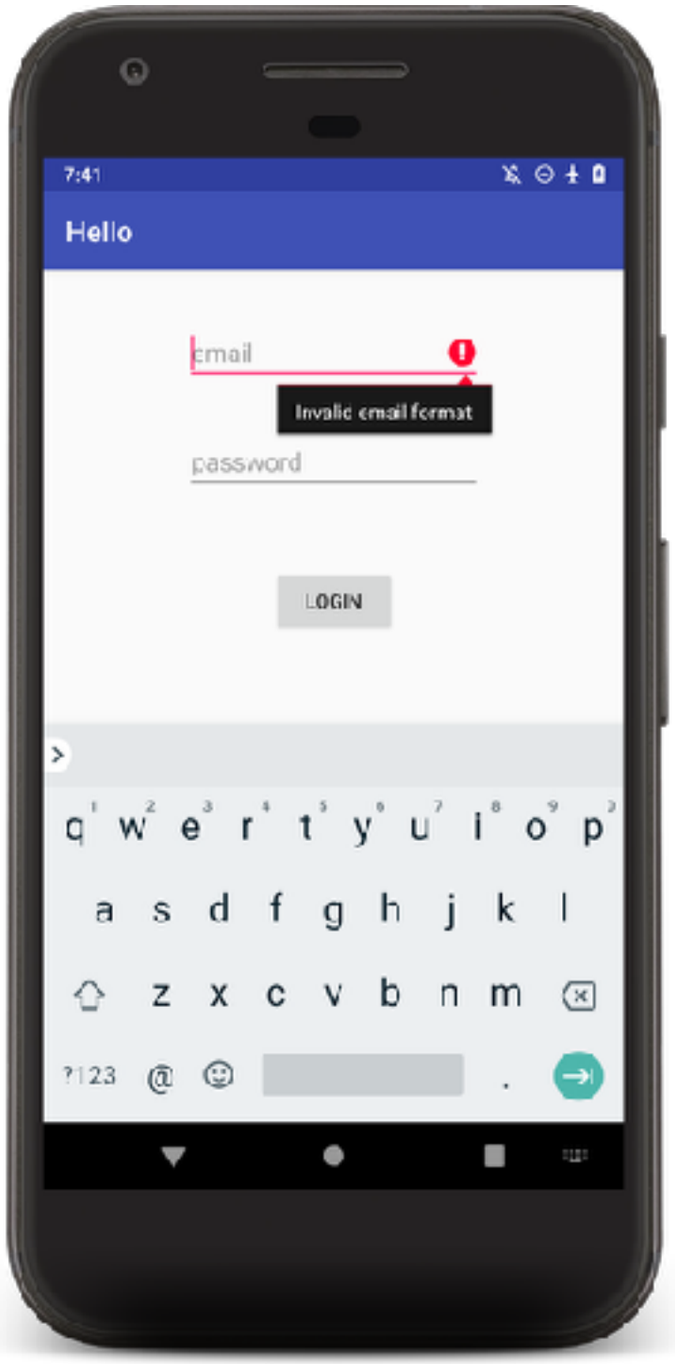


https://developer.android.com/studio/test/command-line

Validation input

https://github.com/up1/course-effective-android-testing/wiki/Validation

How to write test cases ?



Sample code to validate inputs

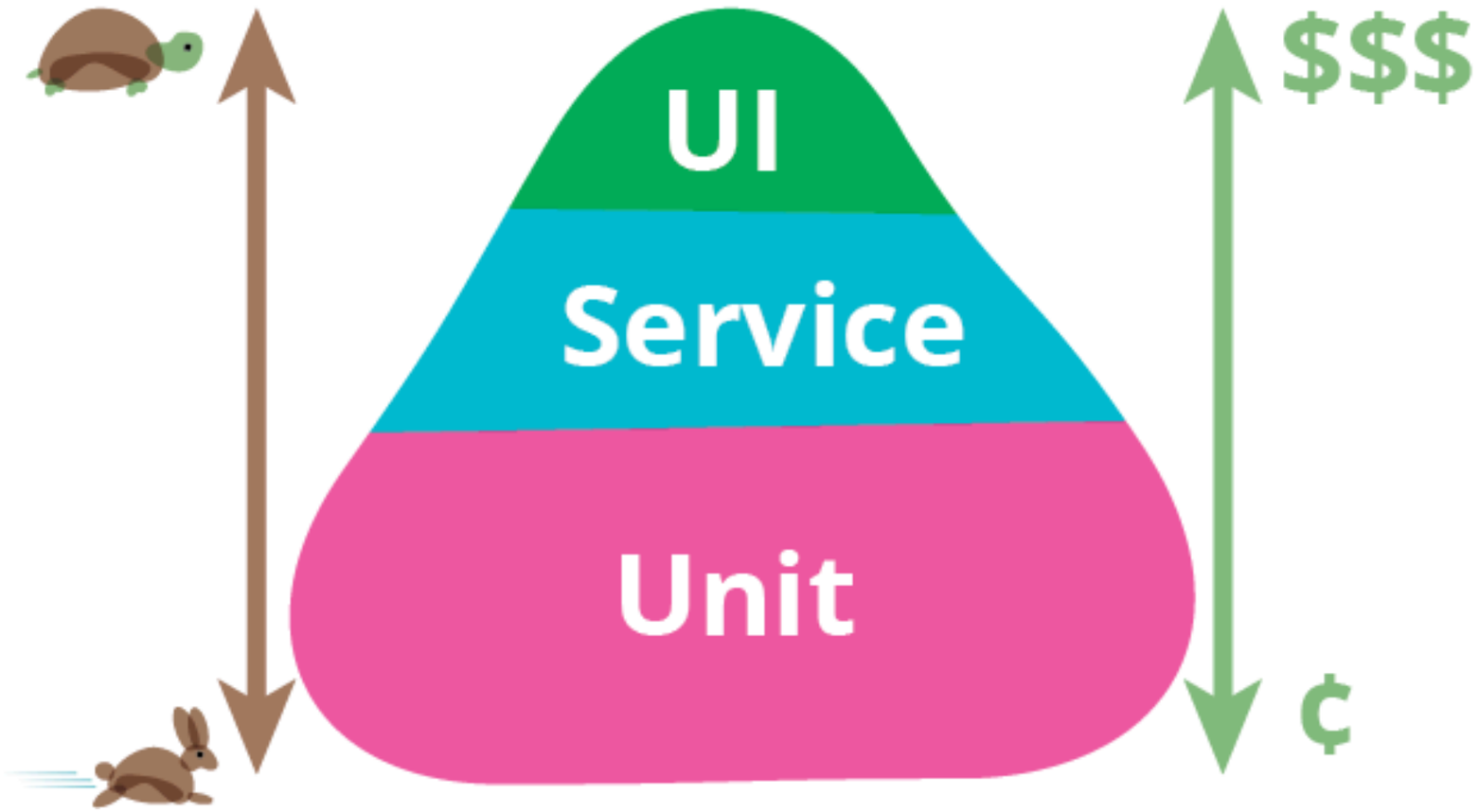
```
// Validation
if(TextUtils.isEmpty(email)) {
    edtEmail.setError("Invalid email");
    return;
}
```

```
if(TextUtils.isEmpty(password)) {
    edtPassword.setError("Invalid password");
    return;
}
```

Test cases

Test Case Name	Email	Password	Expected Result
Empty email			Invalid email
Empty password			Invalid password

Write test cases ?



UI testing

UI testing with validation input

Create class **MainActivityValidateInputTest**
in /app/src/androidTest/java

Case 1 :: Empty email

```
@Test
public void invalid_email_with_empty_email() {
    onView(withId(R.id.edtEmail))
            .perform(
                    typeText(""),
                    closeSoftKeyboard());
    onView(withId(R.id.edtPassword))
            .perform(
                    typeText("12345"),
                    closeSoftKeyboard()
            );
    onView(withId(R.id.btnLogin))
            .perform(click());

    onView(withId(R.id.edtEmail))
            .check(matches(hasErrorText("Invalid email")));
}
```

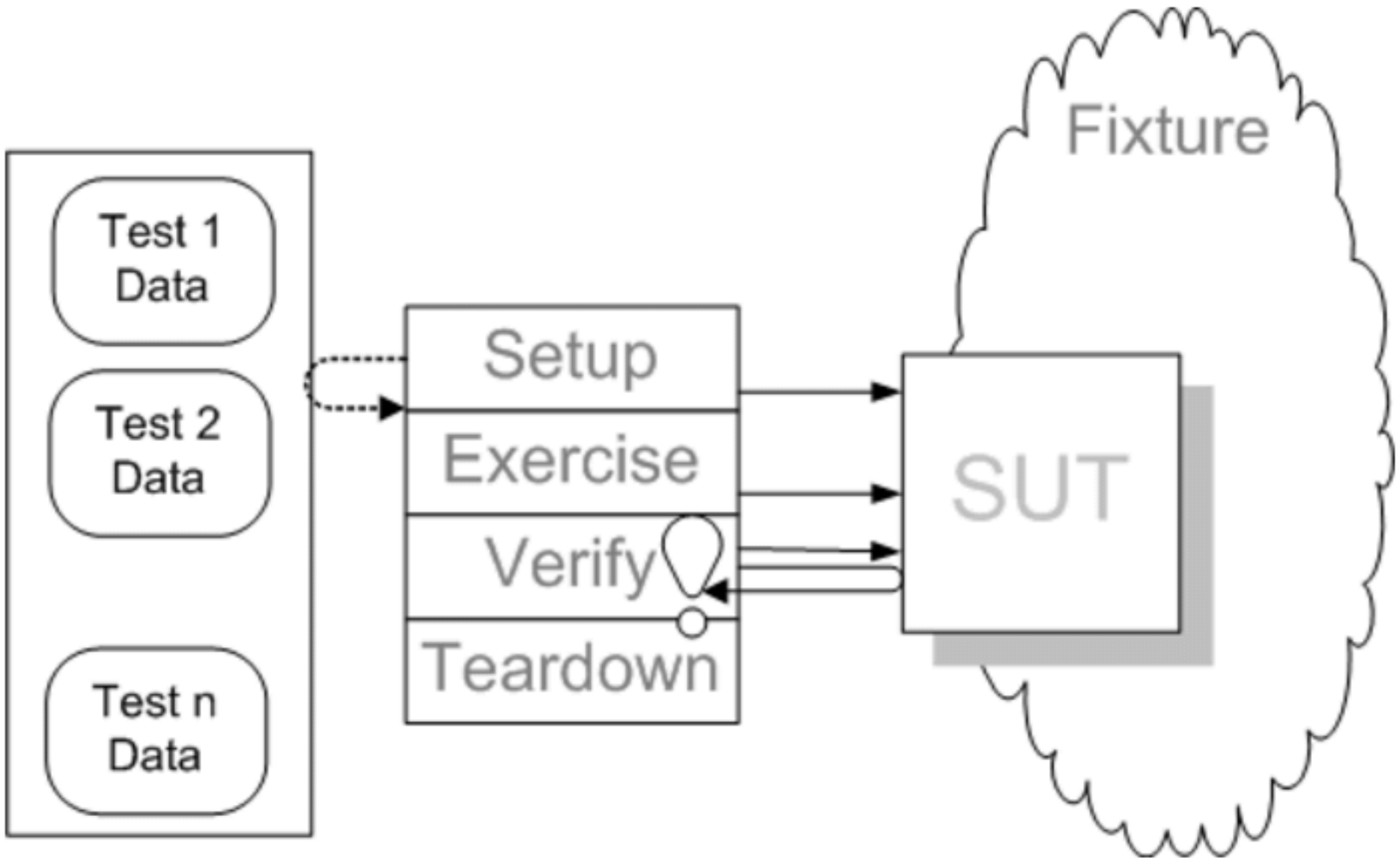
Case 2 :: Empty password

```
@Test
public void invalid_password_with_empty_password() {
    onView(withId(R.id.edtEmail))
            .perform(
                    typeText("somkiat@gmail.com"),
                    closeSoftKeyboard());
    onView(withId(R.id.edtPassword))
            .perform(
                    typeText(""),
                    closeSoftKeyboard()
            );
    onView(withId(R.id.btnLogin))
            .perform(click());

    onView(withId(R.id.edtPassword))
            .check(matches(hasErrorText("Invalid password")));
}
```

Add more tests …

Working with data-driven testing

Data-Driven Testing



Data-Driven Testing

JUnit Parameterized
JUnitParams

1. JUnit Parameterized

JUnit Parameterized
JUnitParams

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

Working with Parameterized

RunWith class Parameterized
Add @Parameters in data method
Create constructor of test to receive data
Write test case with data

Working with Parameterized (1)

1

```
@RunWith(Parameterized.class)
public class MainActivityValidateInputParameterizedTest {

    @Parameters
    public static Collection<Object[]> data() {
        return Arrays.asList(new Object[][] {
                { "", "12345", "Invalid email" }
        });
    }
}
```

2

Working with Parameterized (2)

Create datas and constructor

```
private String email;
private String password;
private String expectedResult;

public MainActivityValidateInputParameterizedTest(
        String email, String password, String expectedResult) {
    this.email = email;
    this.password = password;
    this.expectedResult = expectedResult;
}
```

Working with Parameterized (3)

Create test case and use datas

```
@Test
public void validate() {
    onView(withId(R.id.edtEmail))
            .perform(
                    typeText(this.email),
                    closeSoftKeyboard());
    onView(withId(R.id.edtPassword))
            .perform(
                    typeText(this.password),
                    closeSoftKeyboard()
            );
    onView(withId(R.id.btnLogin))
            .perform(click());

    onView(withId(R.id.edtEmail))
            .check(matches(hasErrorText(this.expectedResult)));
}
```

1

2

3

Run your test

$gradlew cAT

Class com.example.somkiat.hello.MainActivityValidateInputParameterizedTest

all > com.example.somkiat.hello > MainActivityValidateInputParameterizedTest

1	0	2.585s
tests	failures	duration

100%
successful

Tests

Test	Nexus_5X_API_28(AVD) - 9
validate[0]	passed (2.585s)

2. JUnitParams

Add dependency in /app/build.gradle

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0-rc01'
    implementation 'com.android.support.constraint:constraint-layout:
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test:rules:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espre
```

```
androidTestImplementation 'pl.pragmatists:JUnitParams:1.1.1'
```

https://github.com/Pragmatists/JUnitParams

Using JUnitParams

```
@RunWith(JUnitParamsRunner.class)
public class MainActivityValidateInputJUnitParamsTest {
```

1

```
    @Test
    @Parameters({", 12345, Invalid email" })
    public void validate(String email, String password, String expectedResult) {
        onView(withId(R.id.edtEmail))
                .perform(
                        typeText(email),
                        closeSoftKeyboard());
        onView(withId(R.id.edtPassword))
                .perform(
                        typeText(password),
                        closeSoftKeyboard()
                );
```

2

https://github.com/Pragmatists/JUnitParams

Unit testing

Unit testing with validation input

How to testing with unit test ?
How to write test cases ?

Create new class in /app/src/main

Class Validation

```
import android.text.TextUtils;

public class Validation {

    public boolean isEmpty(String input) {
        return TextUtils.isEmpty(input);
    }

}
```

Create unit test of validation

Create file in /app/test

```
public class ValidationTest {

    @Test
    public void isEmpty() {
        Validation validation = new Validation();
        assertTrue(validation.isEmpty(""));
    }
}
```

Run unit test

$gradlew testDebugUnitTest

Found error :: Method isEmpty in android.text.TextUtils not mock

Change code to use pure Java

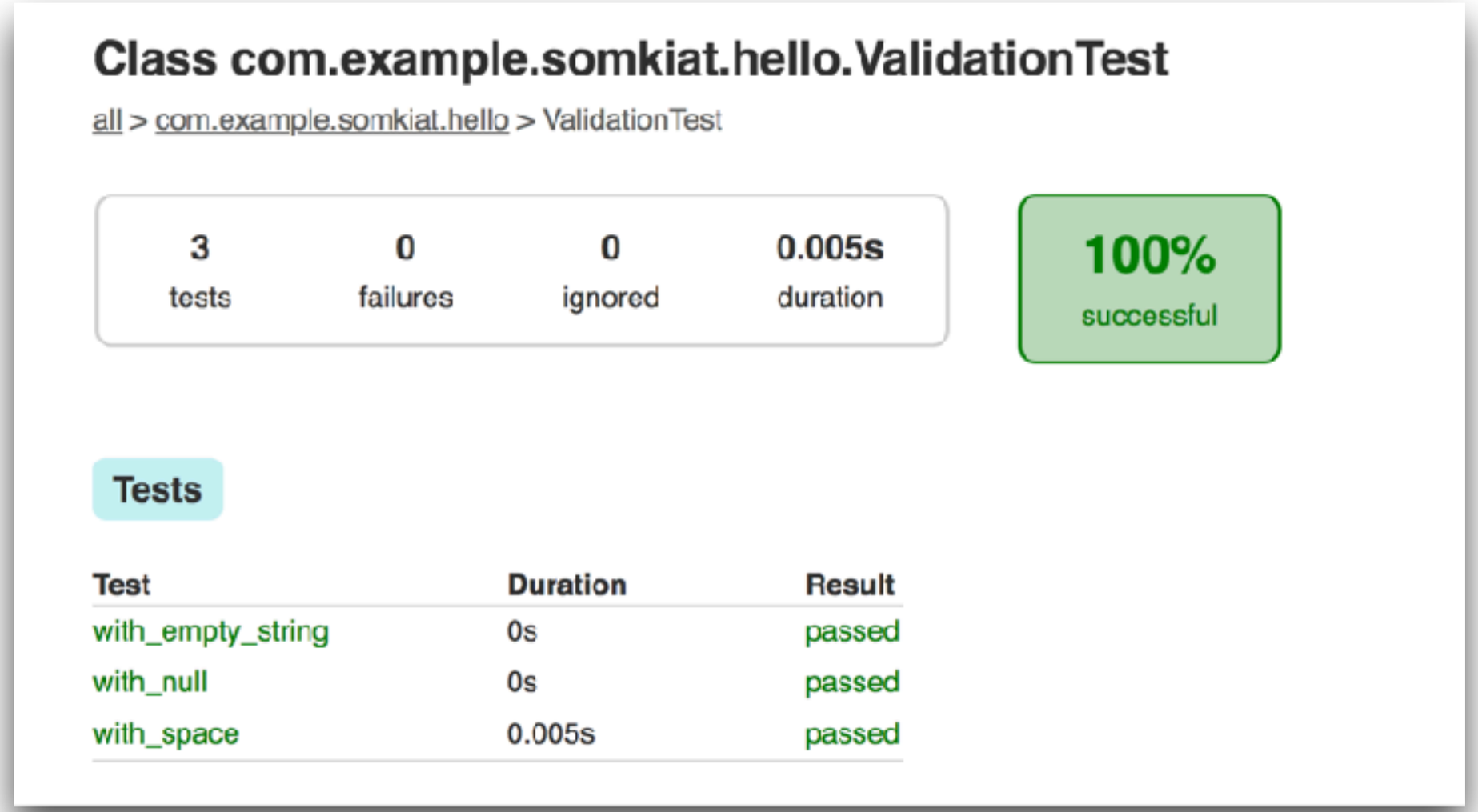
```
public class Validation {

    public boolean isEmpty(String input) {
        return input == null ||
                input.trim().length() == 0;
    }

}
```

Add more test cases

```
@Test
public void with_empty_string() {
    Validation validation = new Validation();
    assertTrue(validation.isEmpty(""));
}

@Test
public void with_null() {
    Validation validation = new Validation();
    assertTrue(validation.isEmpty(null));
}

@Test
public void with_space() {
    Validation validation = new Validation();
    assertTrue(validation.isEmpty(" "));
}
```

Run unit test again

$gradlew testDebugUnitTest



Class com.example.somkiat.hello.ValidationTest

all > com.example.somkiat.hello > ValidationTest

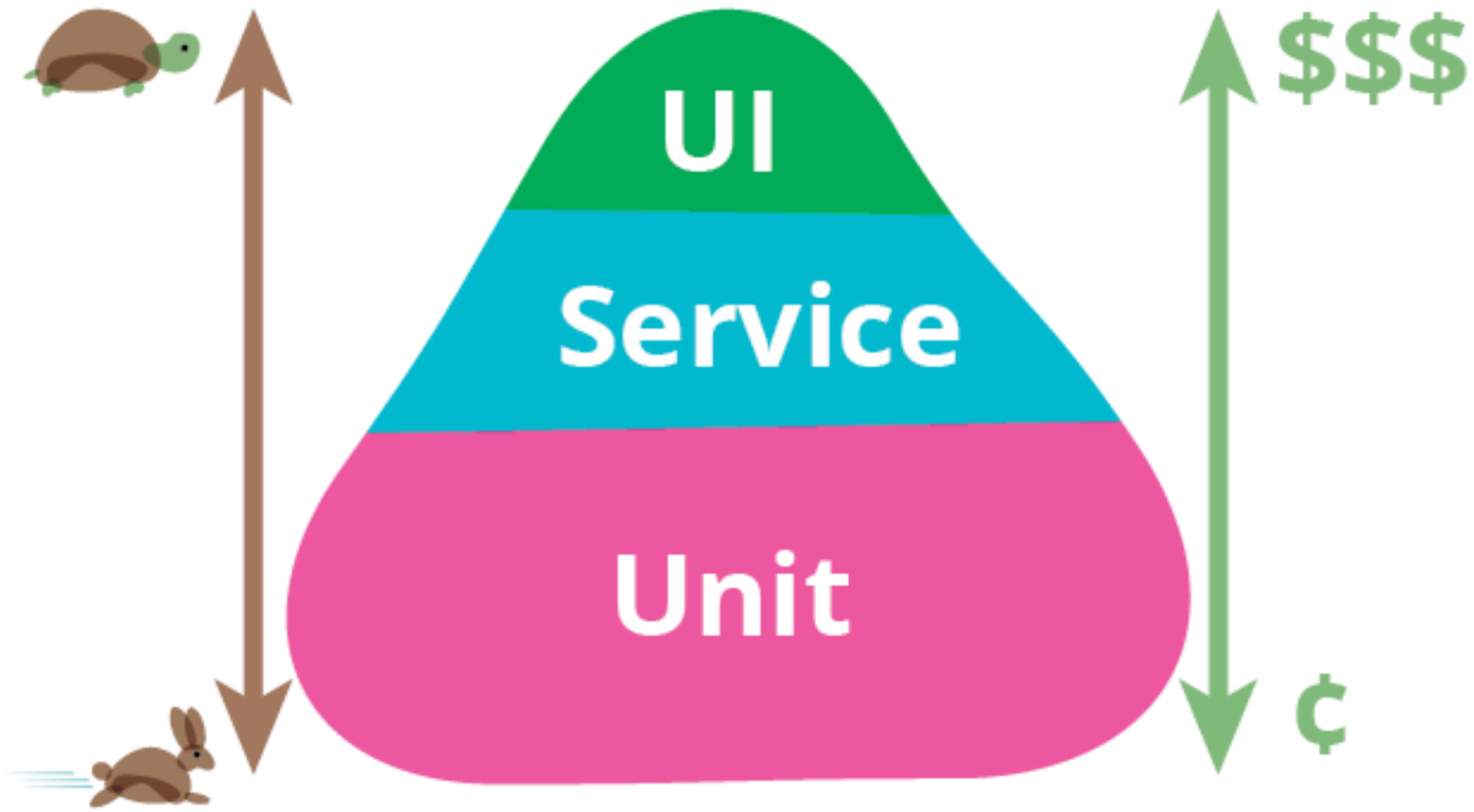
3	0	0	0.005s
tests	failures	ignored	duration

100% successful

Tests

Test	Duration	Result
with_empty_string	0s	passed
with_null	0s	passed
with_space	0.005s	passed

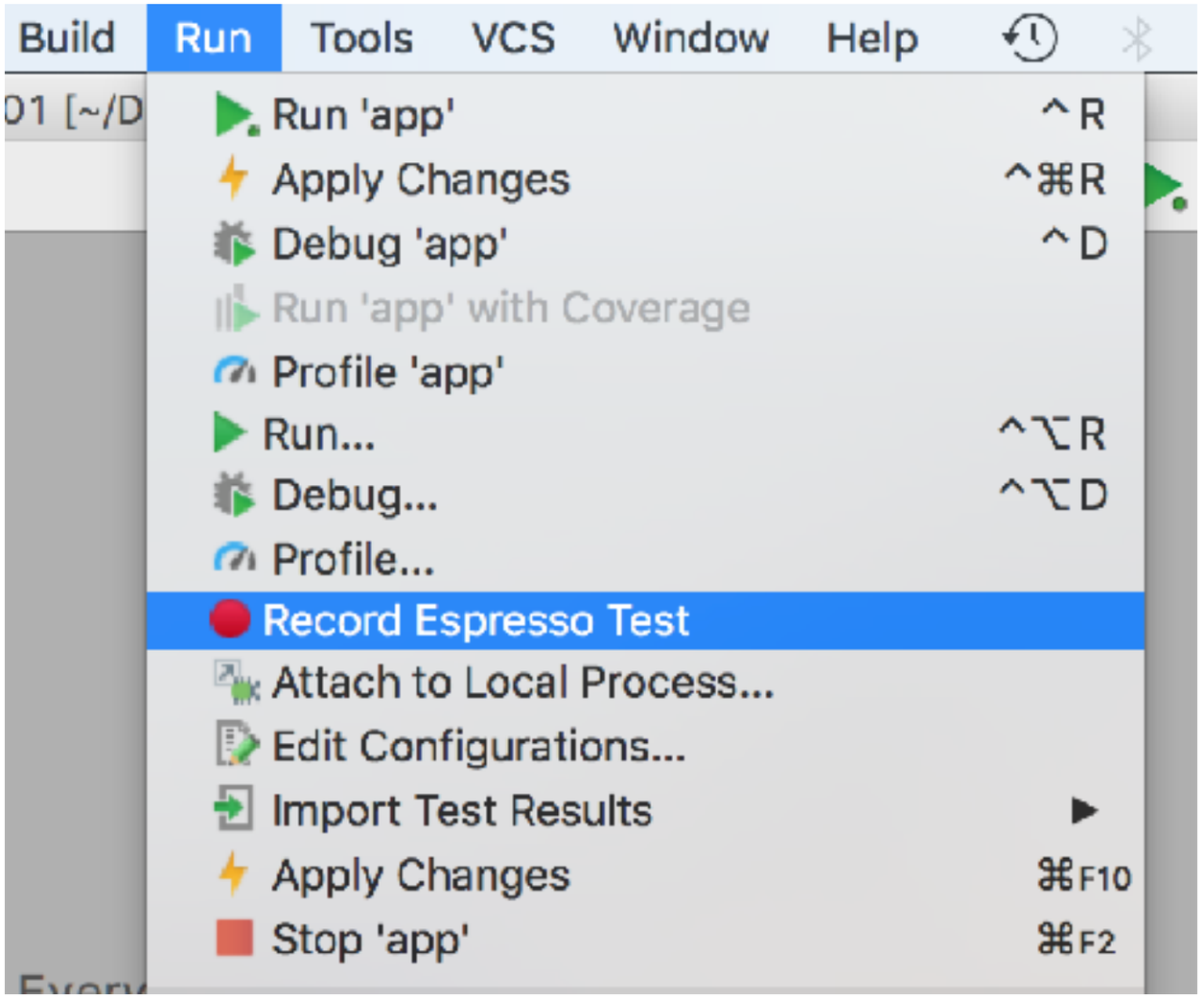
Unit vs UI testing ?



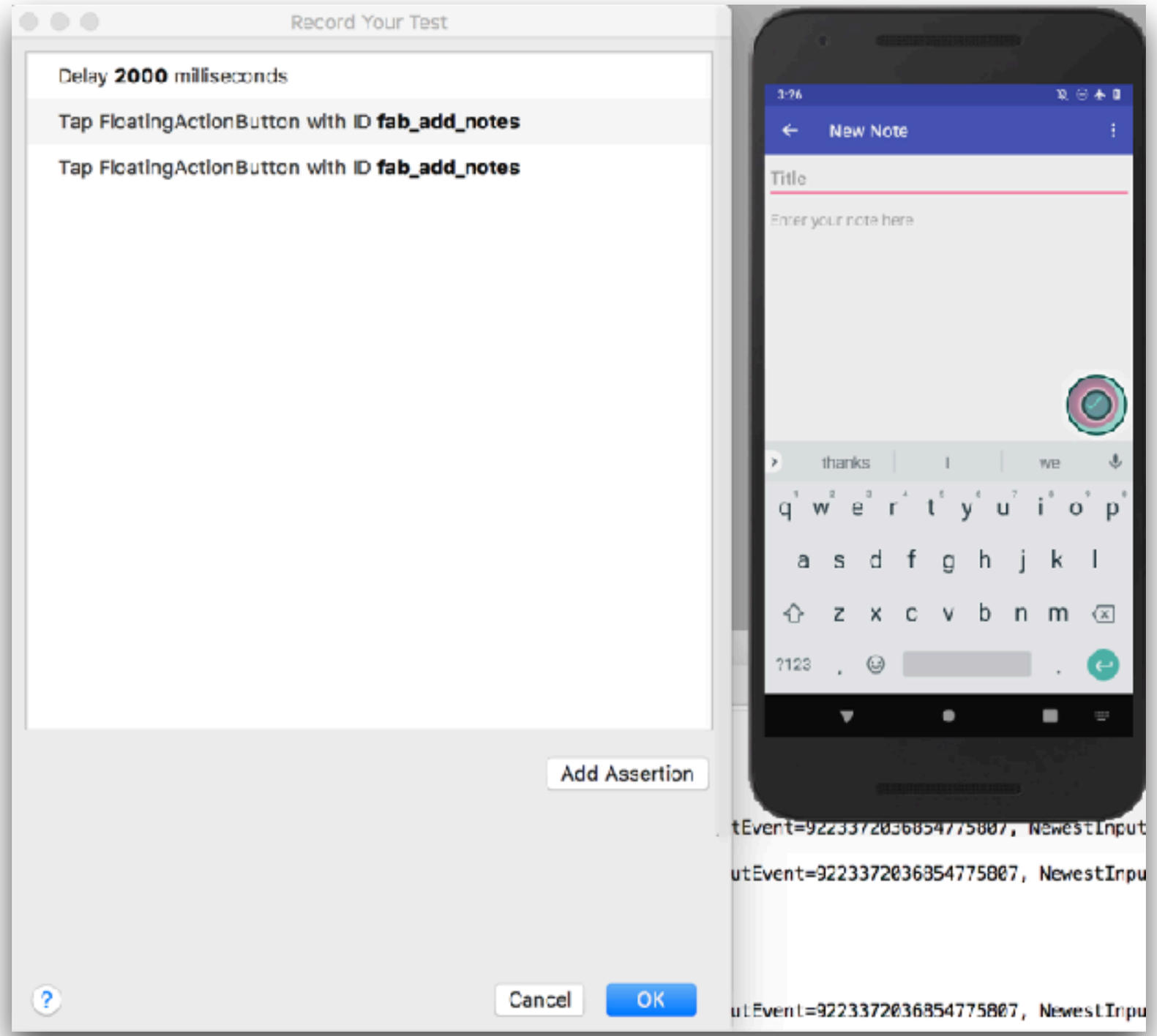
Record Espresso Test

Record Espresso Test

Run -> Record Espresso Test



Record Espresso Test

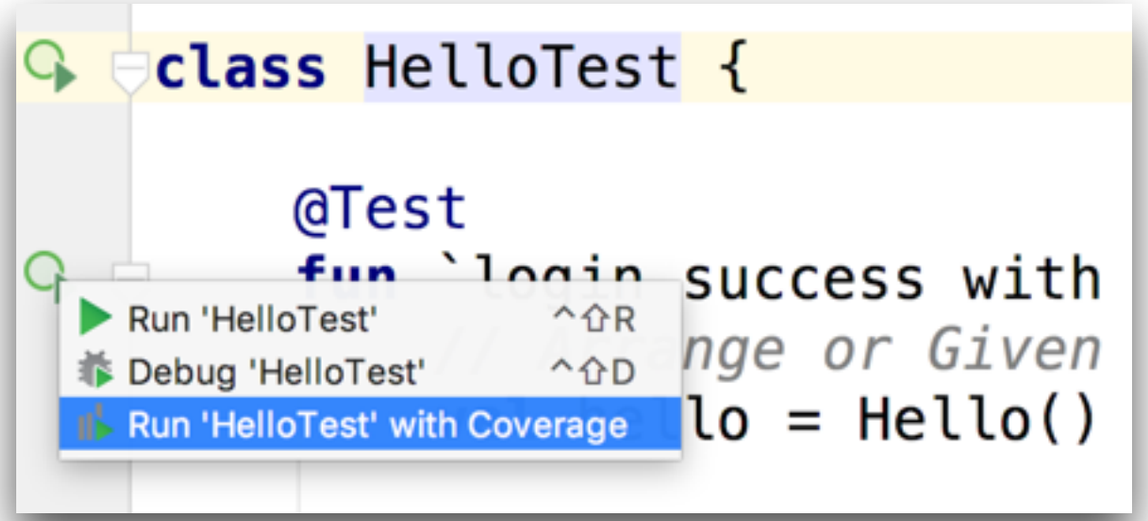


Code coverage problem

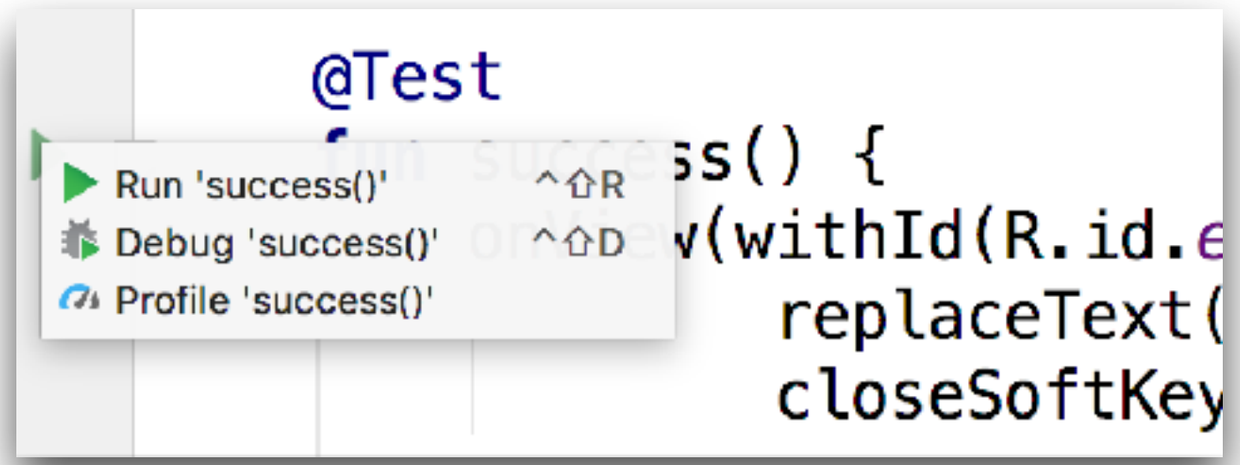
Code coverage for android

Separate unit test and android test

Unit test



androidTest



1. 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/deb
    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/connect
    ])
}
```

2. Run test with code coverage

$gradlew clean jacocoTestReport

Result in /app/build/reports/jacoco/jacocoTestReport

app > demo.loginapplication

demo.loginapplication

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines
MainActivity		82%		50%	7	13	1	13
Hello		100%		n/a	0	2	0	2
Total	18 of 119	84%	7 of 14	50%	7	15	1	15

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("somkiat@xxx.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

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 specif
ckage_name'
[07:19:15]: The package name of the app under test (e.g.
com.yourcompany.yourapp):
```

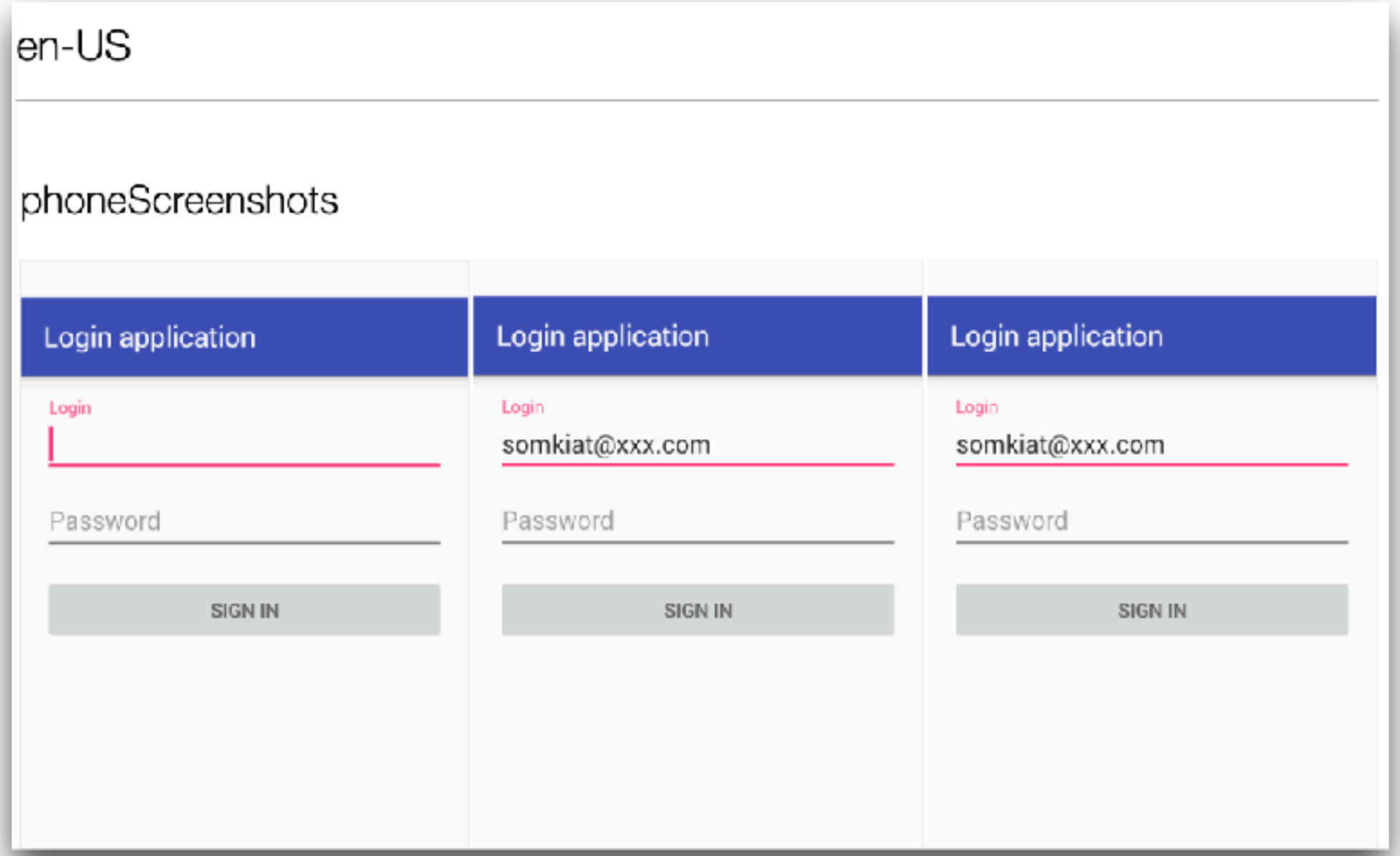
6. Try to config screngrabfile

$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**



Problem with Screengrab

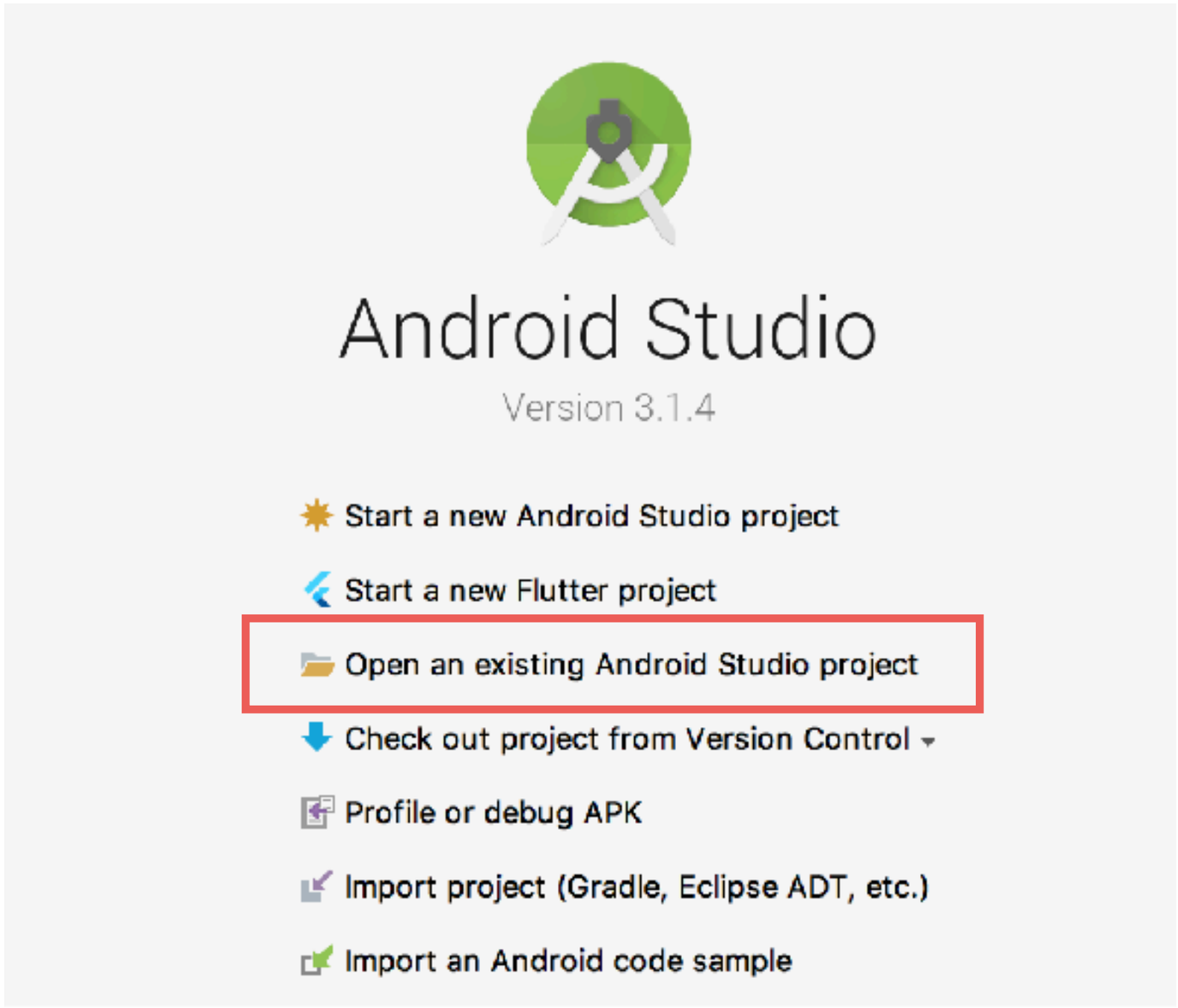
Version of screen grab for Android N+

Solution :: reduce version of screengrab

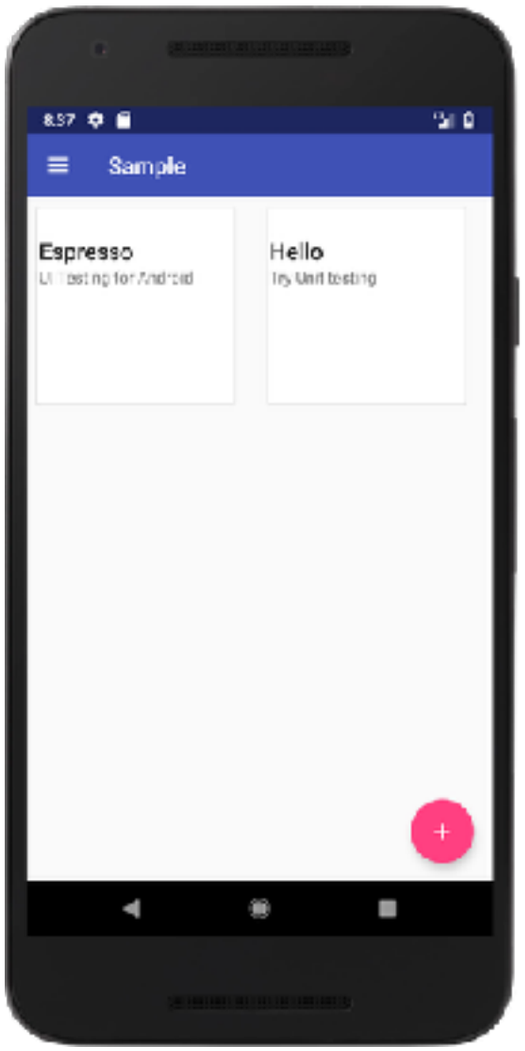
Workshop with Testing

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

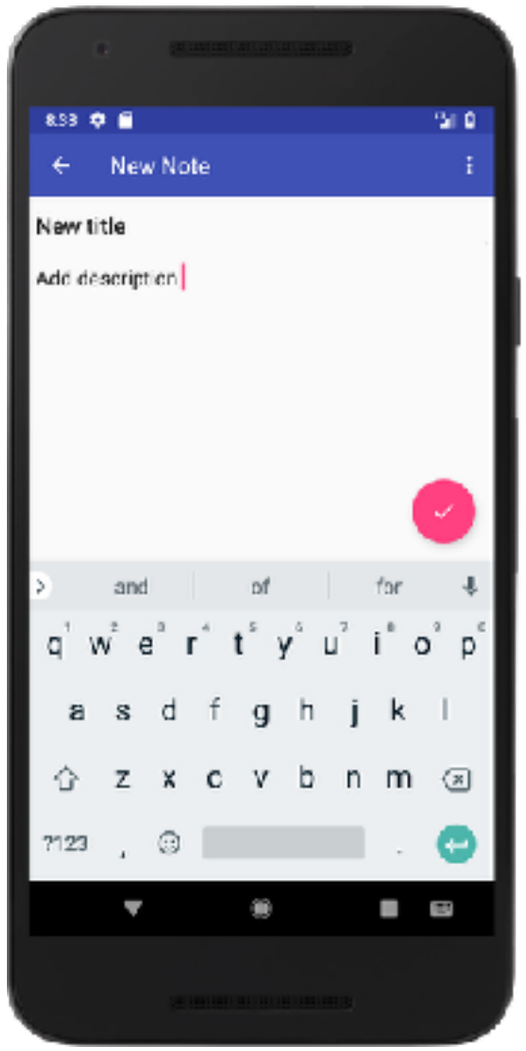
Open an existing project



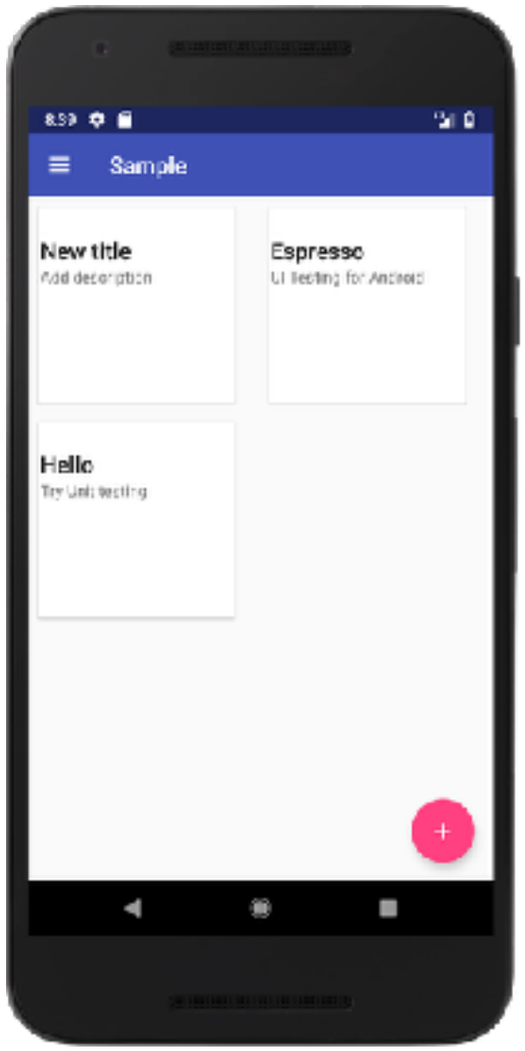
Main



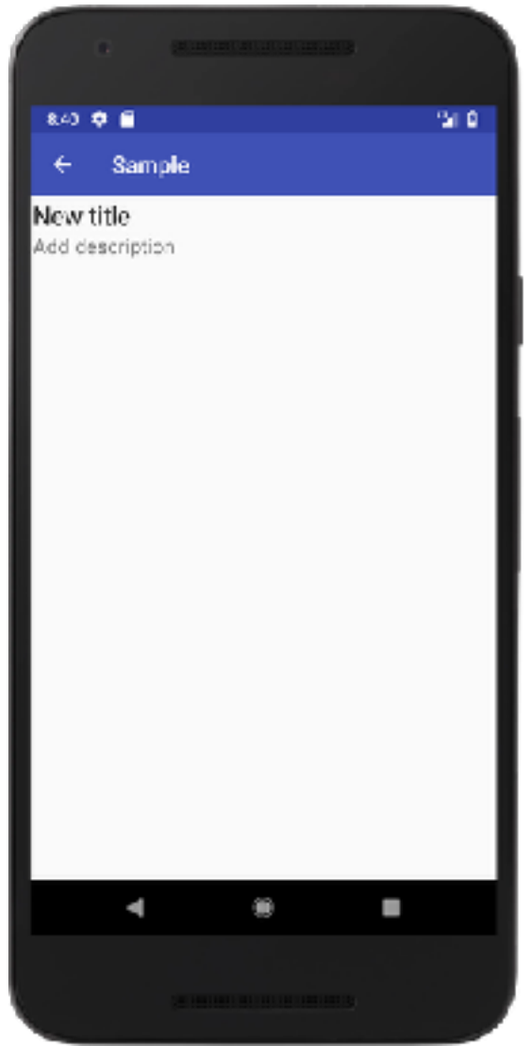
Add



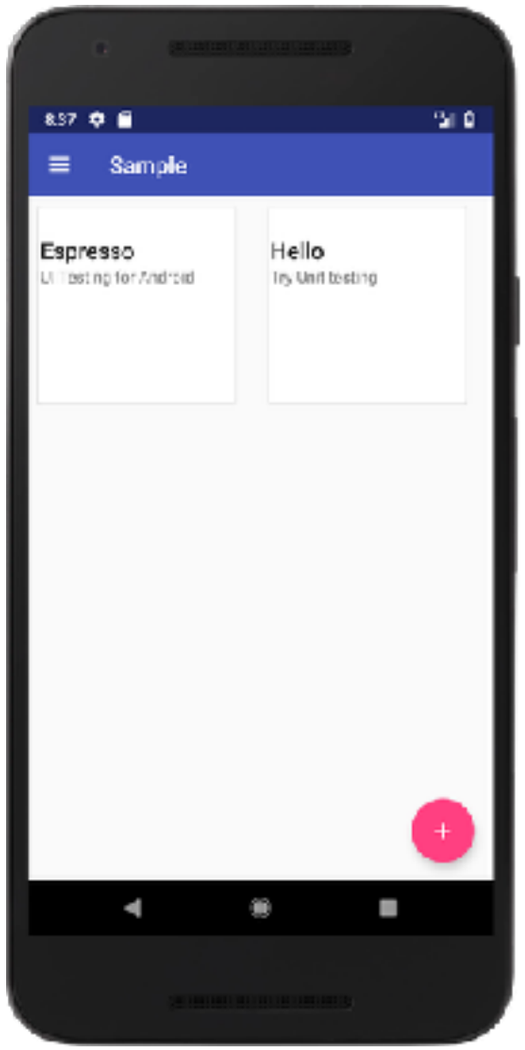
Main



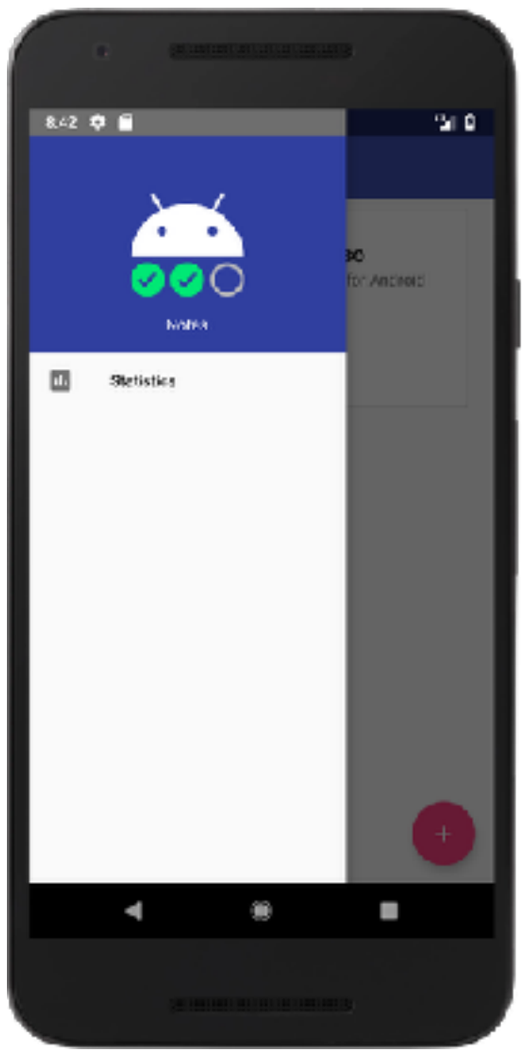
Detail



Main



Menu



Test cases ?

Test Case Name			Expected Result
Add new note			
Show detail of note			

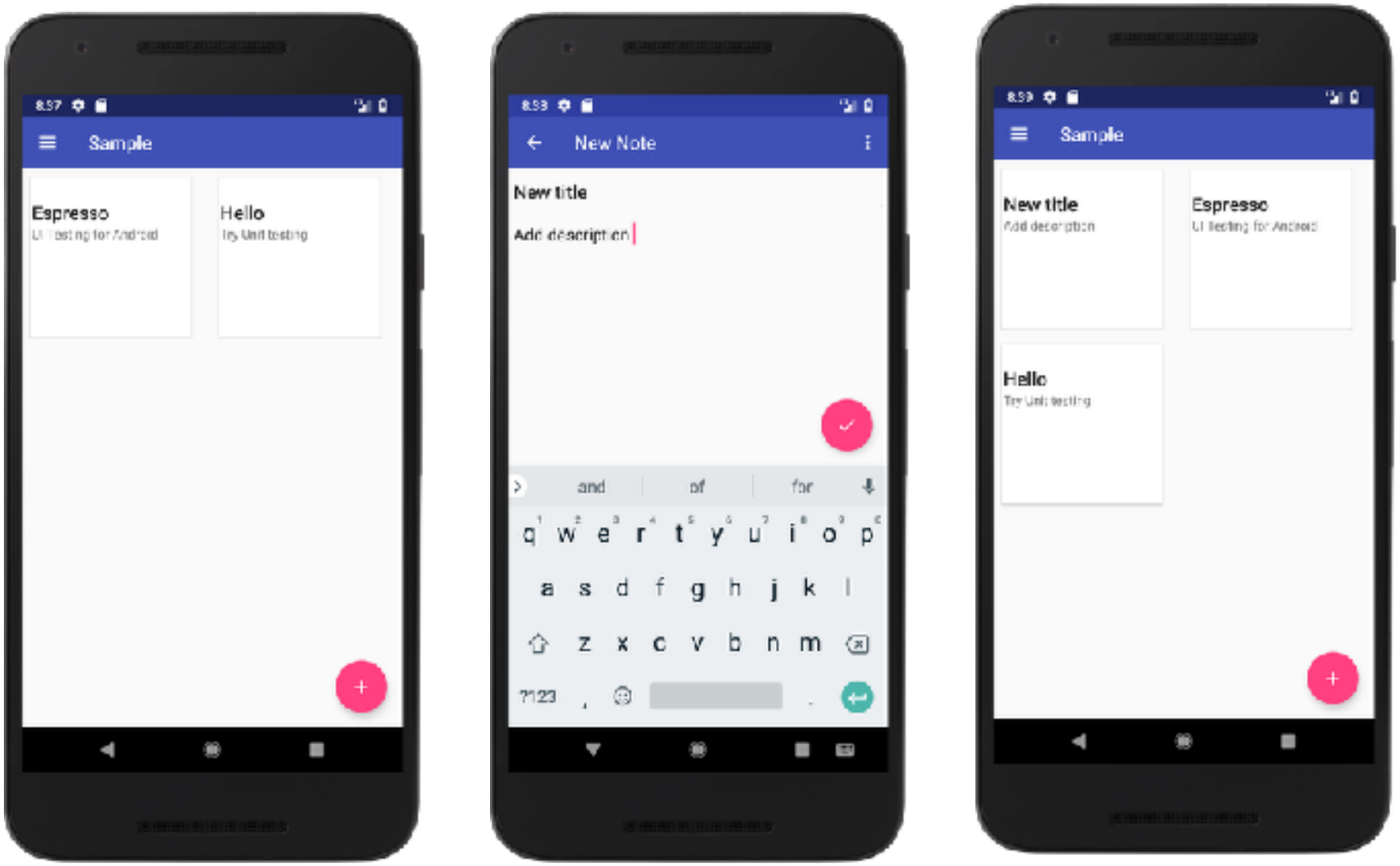
Add new note

Add new note

Main

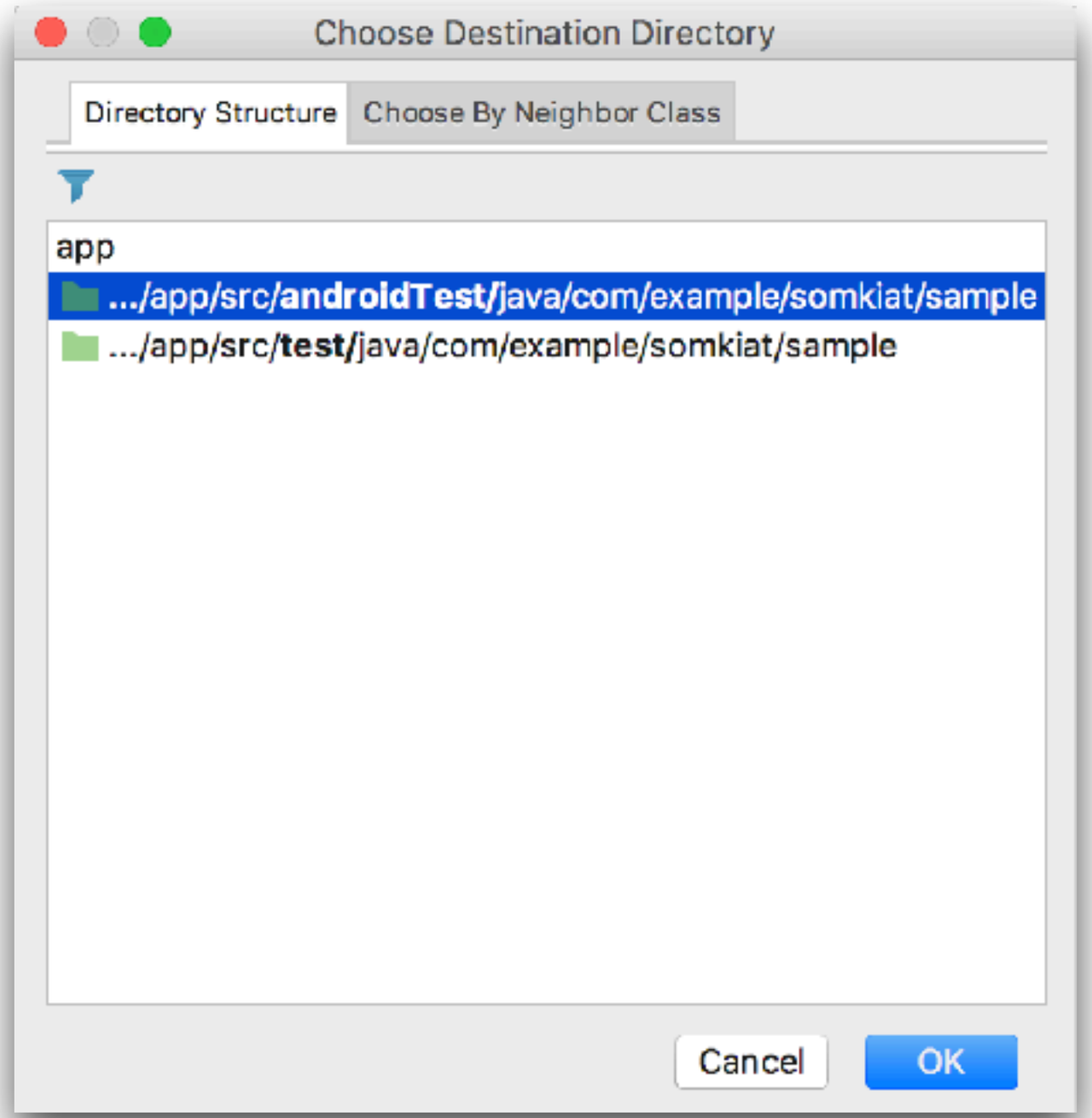
Add

Main



Create file MainActivityTest.java

In folder /app/src/androidTest/java



Test case 1 :: Click add button

Try in click add button

```
@Rule
public ActivityTestRule<MainActivity> mainPage =
        new ActivityTestRule<>(MainActivity.class);

@Test
public void click_add_button_should_be_open_add_note_page() {
    // 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()));
}
```

Test case 2 :: Add new note

```
@Test
public void add_new_note_to_list_of_notes() {
    onView(withId(R.id.fab_add_notes))
            .perform(click());

    // Add new note with title and description
    onView(withId(R.id.add_note_title))
            .perform(typeText("New Title"),
                    closeSoftKeyboard());
    onView(withId(R.id.add_note_description))
            .perform(typeText("New description"),
                    closeSoftKeyboard());

    onView(withId(R.id.fab_add_notes))
            .perform(click());


    // Assert ?
}
```

Working with RecyclerView

1. Add new dependency in /app/build.gradle
2. Scroll to new note
3. Check data in recycler view

```
implementation 'com.android.support:support-v4:27.1.1'
testImplementation 'junit:junit:4.12'
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'
androidTestImplementation "com.android.support.test.espresso:espresso-contrib:3.0.2"
androidTestImplementation "com.android.support.test.espresso:espresso-intents:3.0.2"
```

Scroll to new note

Use scrollTo() from
android.support.test.espresso.contrib.RecyclerViewActions

Working with RecyclerView

Save and check result

Working with RecyclerView

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(withItemText(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

Run test in command line

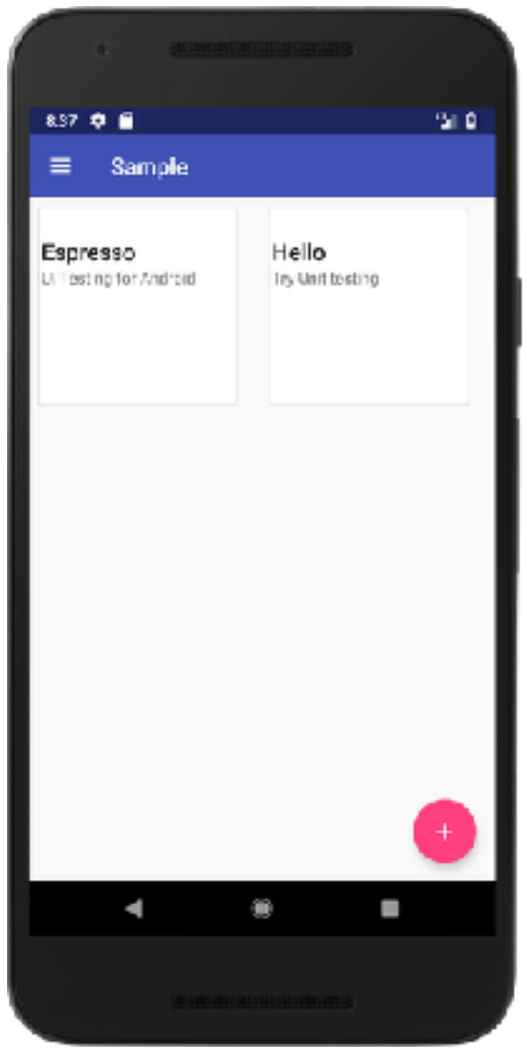
$./gradlew clean cAT

Try to add more note !!

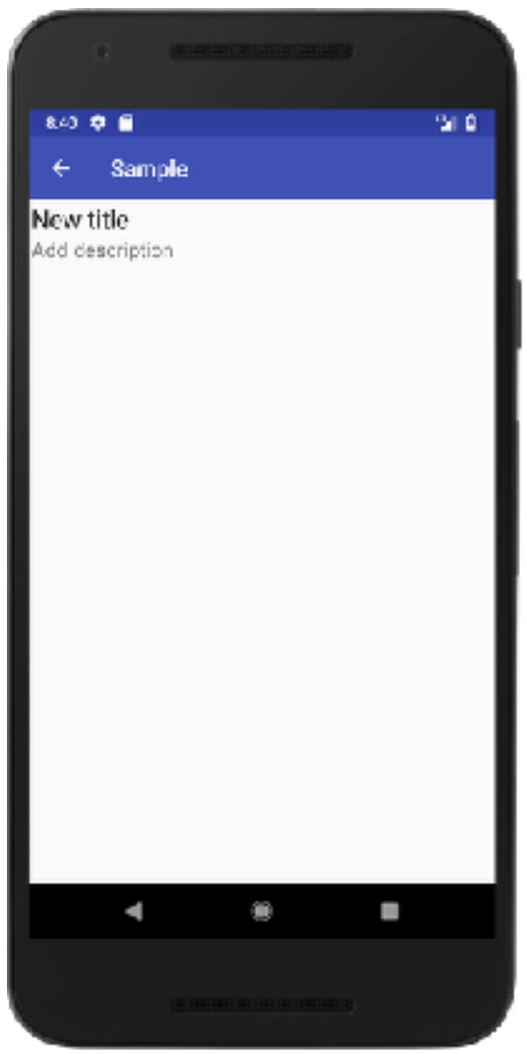
Show detail of note

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(""))) // Expected value ?
    onView(withId(R.id.note_detail_description))
            .check(matches(withText(""))); // Expected value ?
}
```

Problem ?

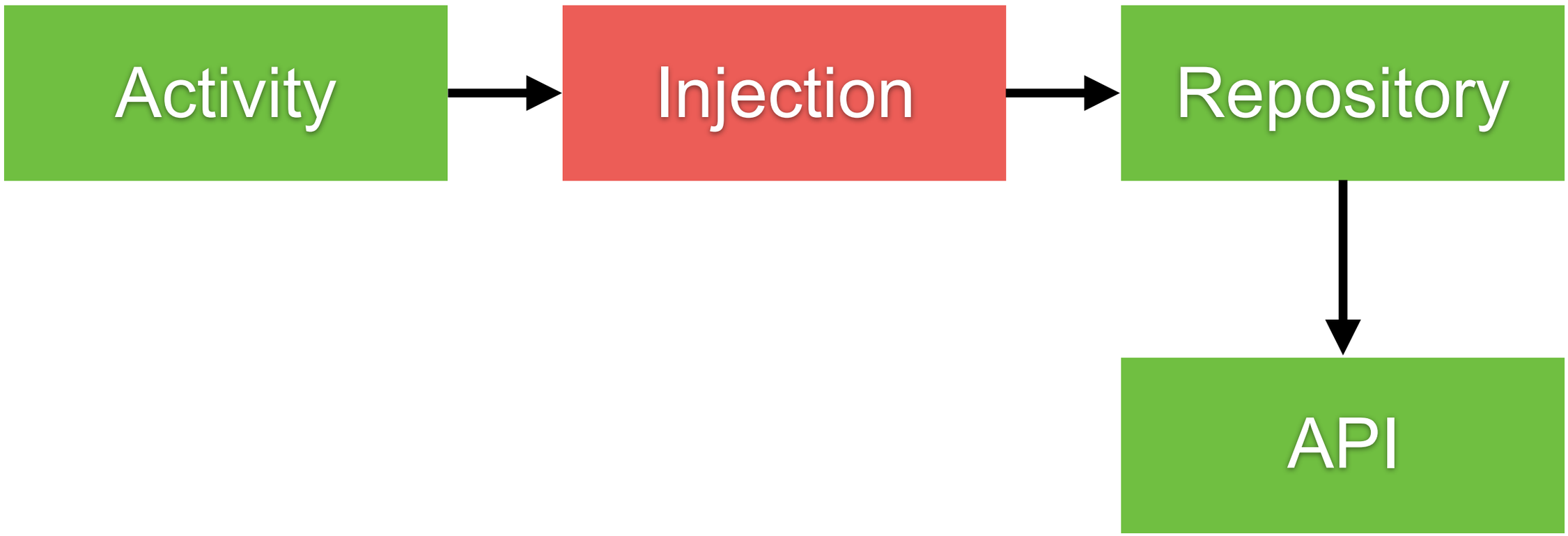
Expected result/value ?

```
@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

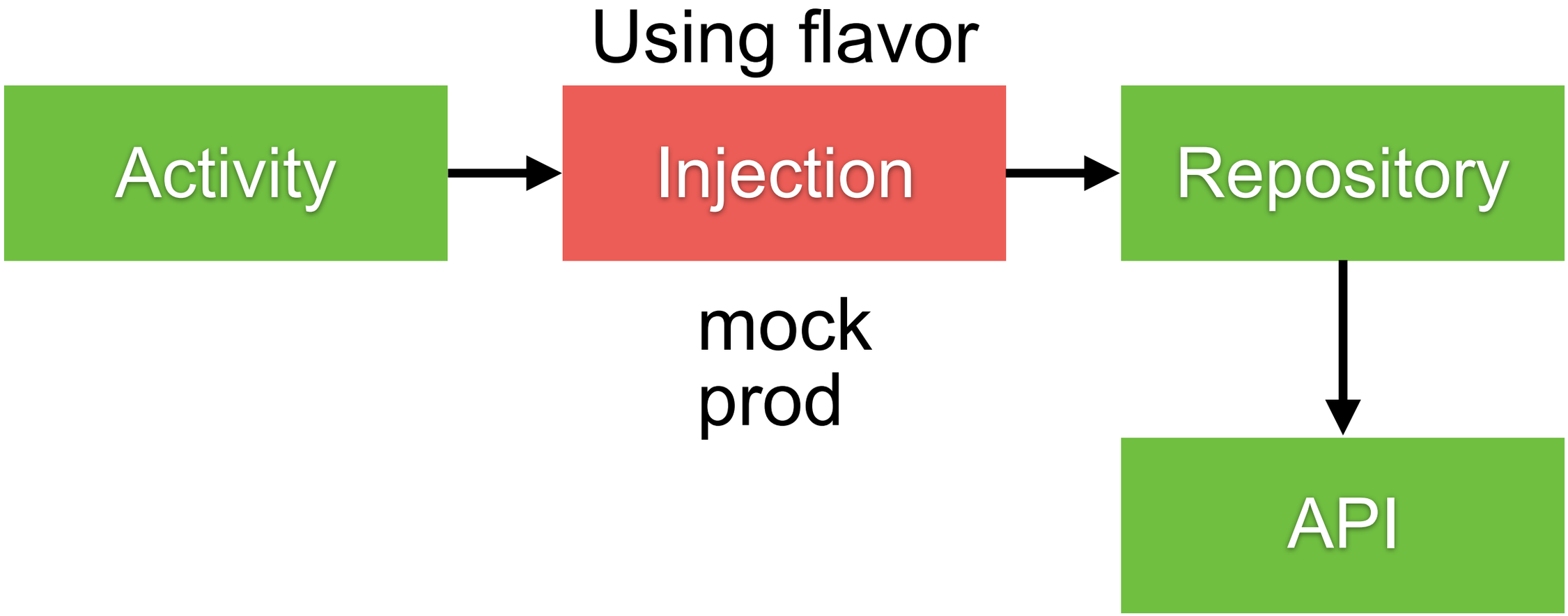


Injection.java

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

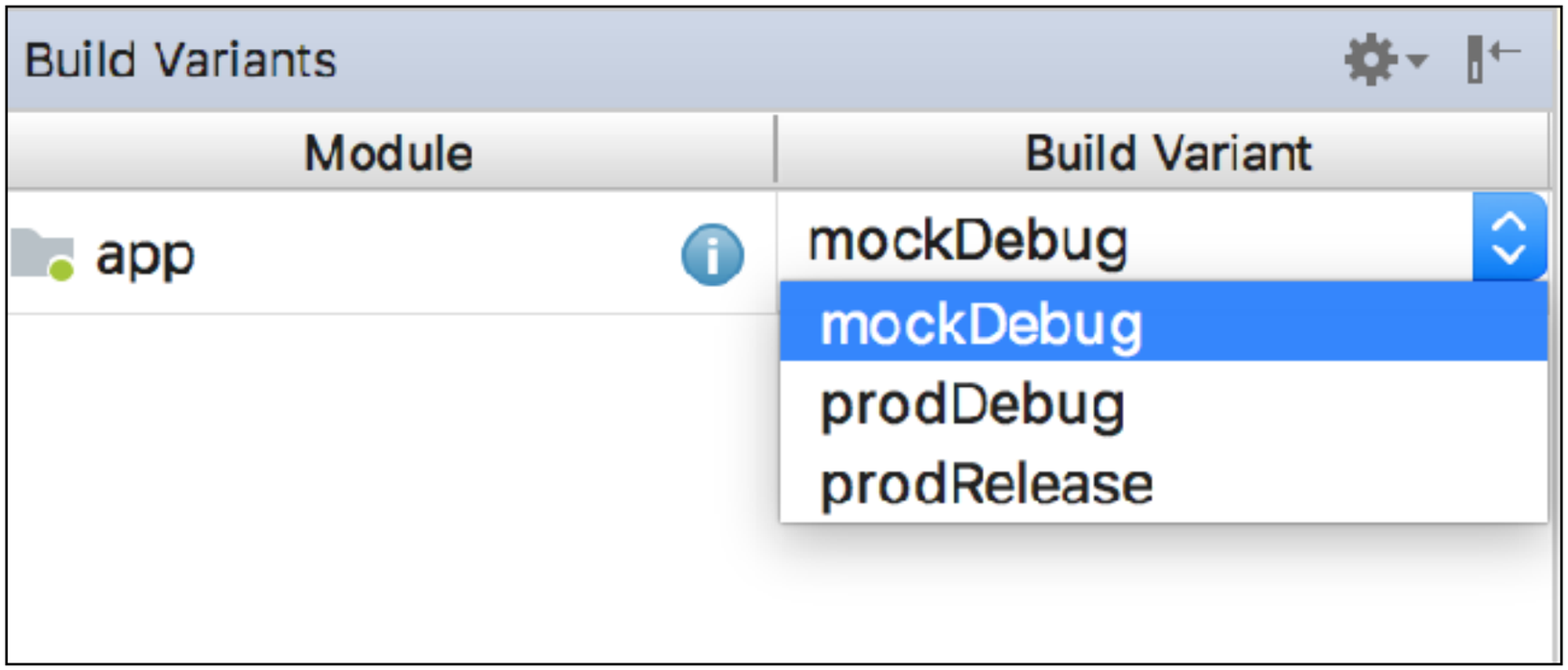
To file /app/build.gradle

```
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



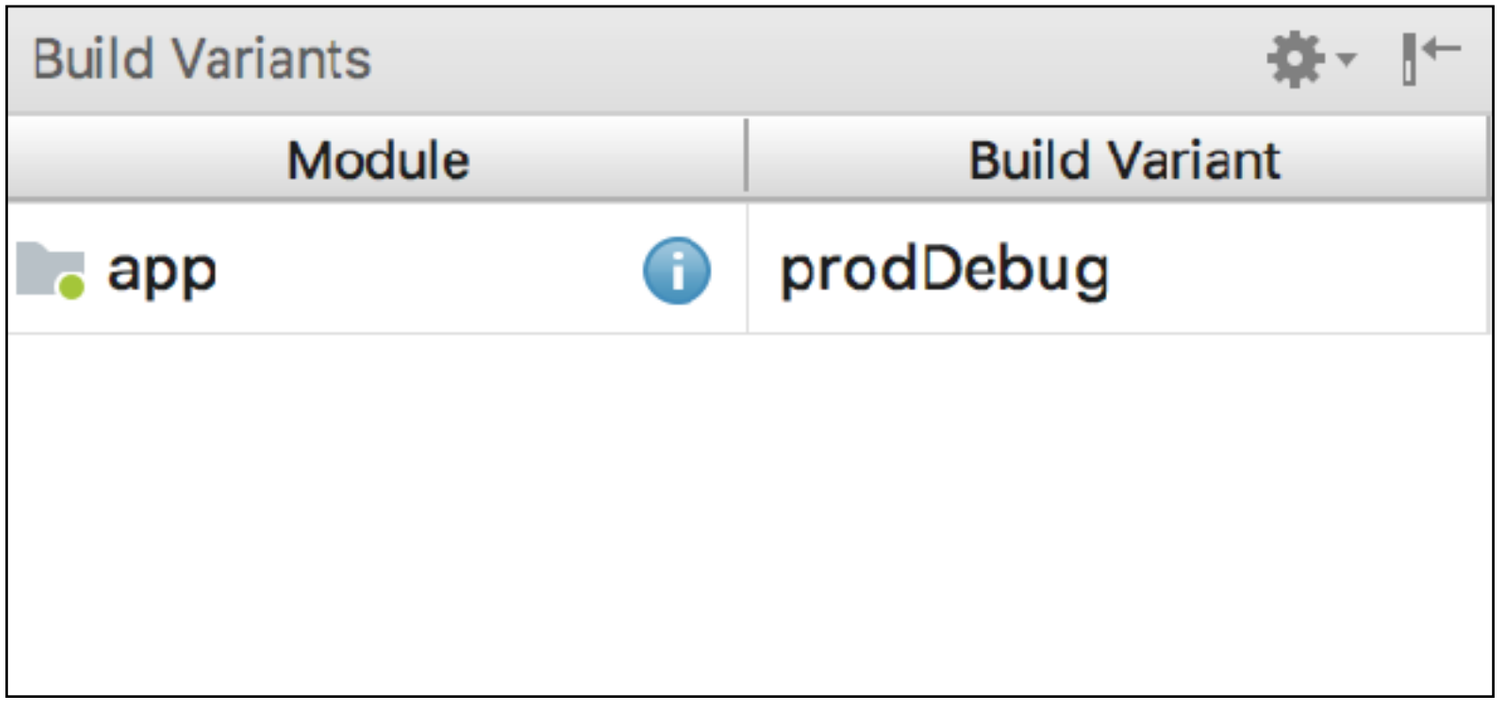
Create new folder in /app/src

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



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	0	11.064s
tests	failures	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%

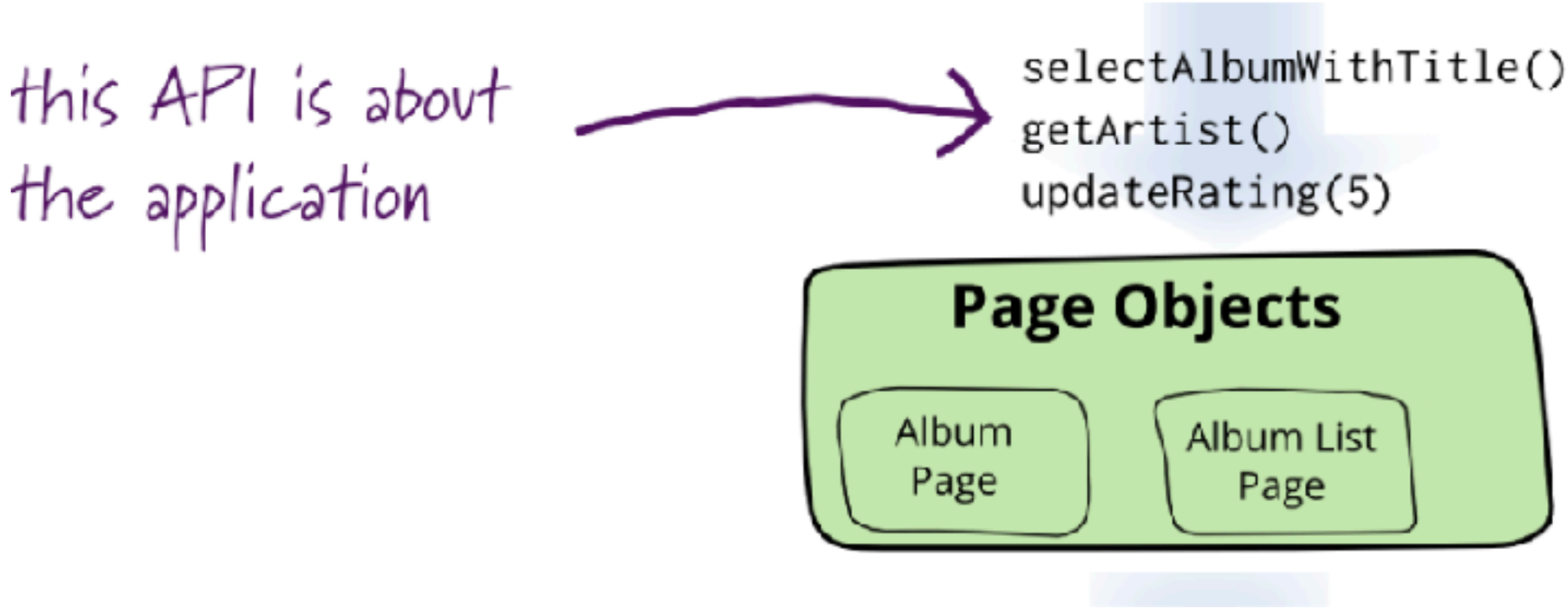
Improve UI Testing with Espresso

1. Improve test structure

Robot or Page Object pattern

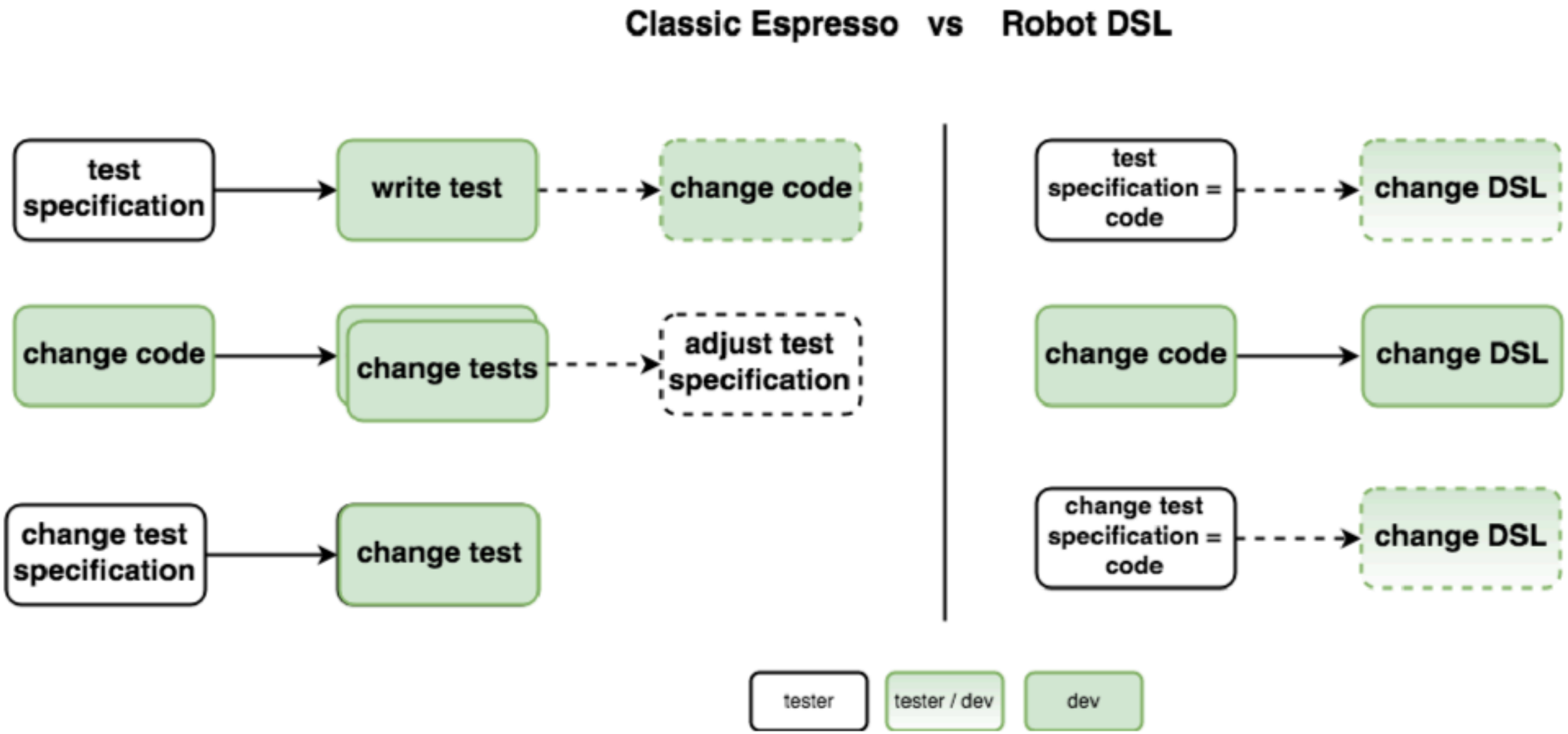
https://martinfowler.com/bliki/PageObject.html

Page Object / Robot pattern



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

Page Object / Robot pattern



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 …

Demo CI/CD with Jenkins