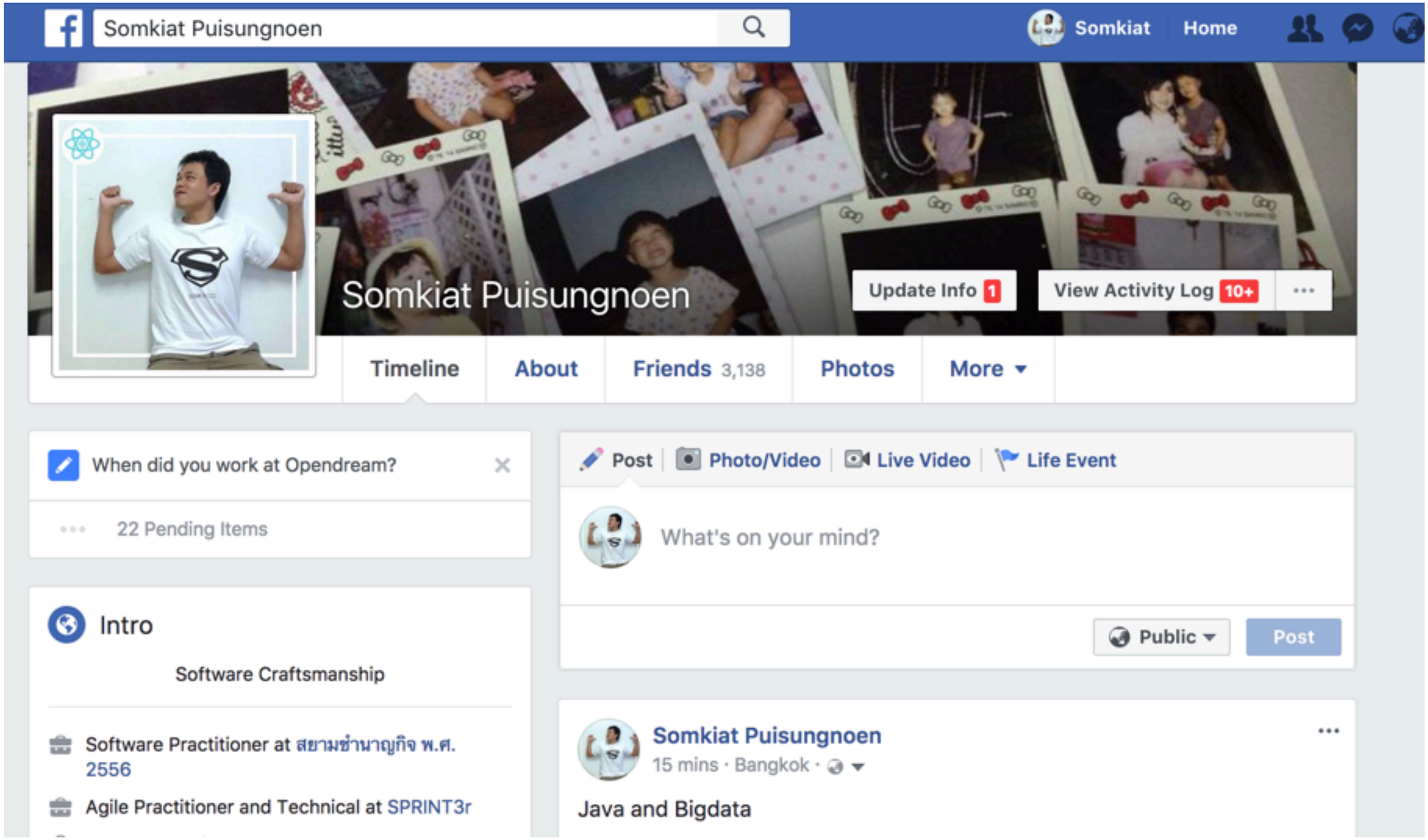
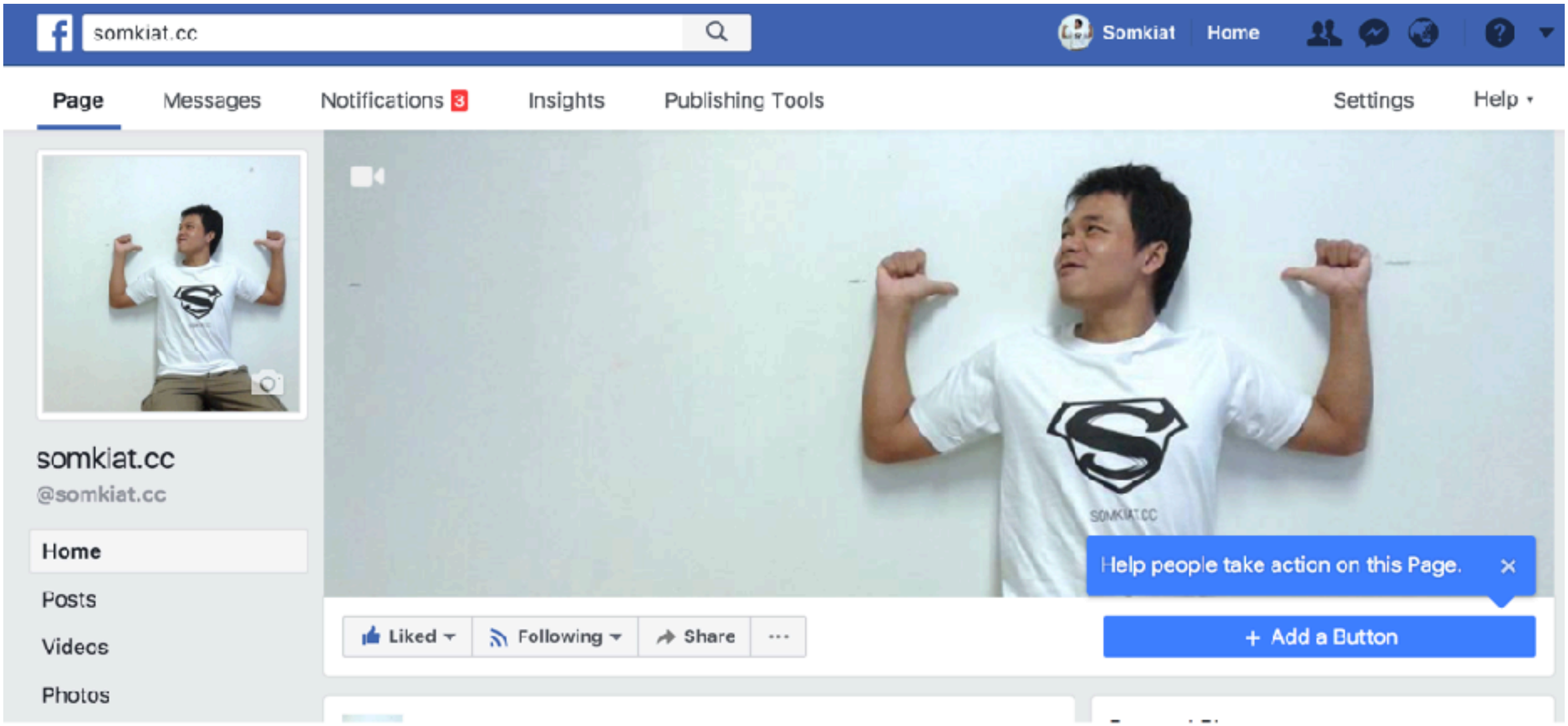
สยามชำนาญกิจ
SIAM CHAMNAN KIT

Basic of Android Testing





https://github.com/up1/workshop-basic-android-testing

Agenda in 3 hours

Introduction of testing

Why we need to test ?

Types of tests

Testing pyramid concept

Android testing

Workshop (step-by-step)

Homework and assignment

Testing for Android app

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

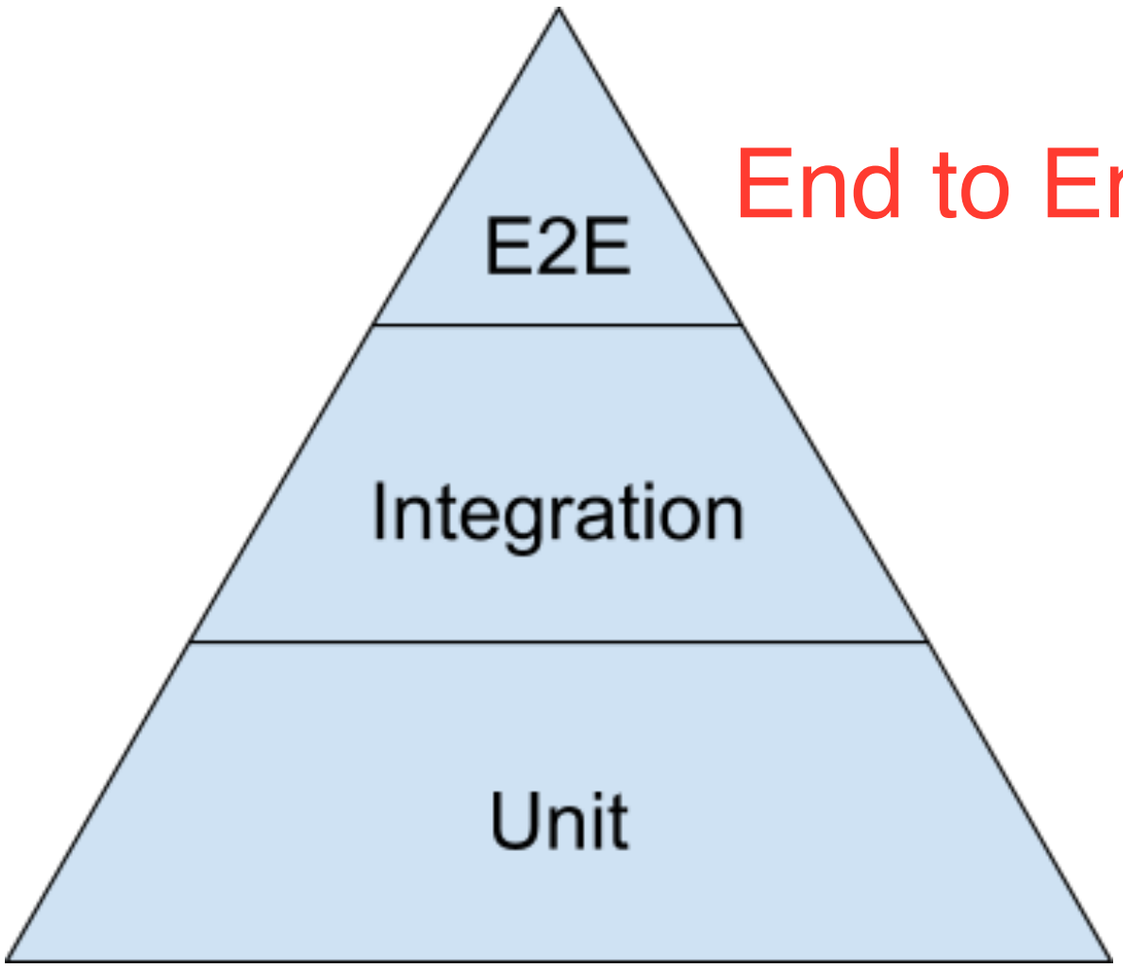
How to **THINK** when and where you should test

What you need to know ?

Android
Android Studio
JUnit 4
Espresso

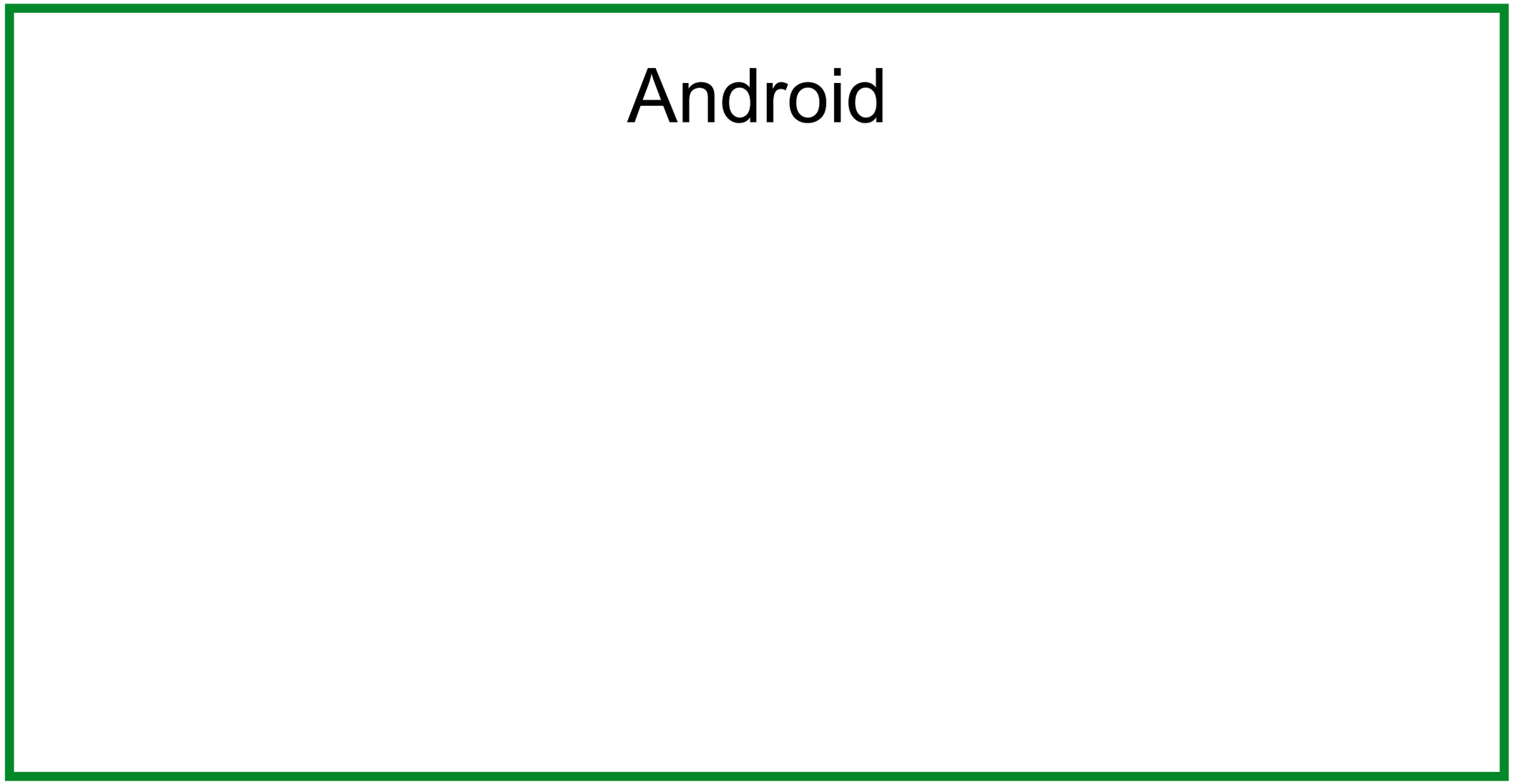
Type of testing

Testing Pyramid

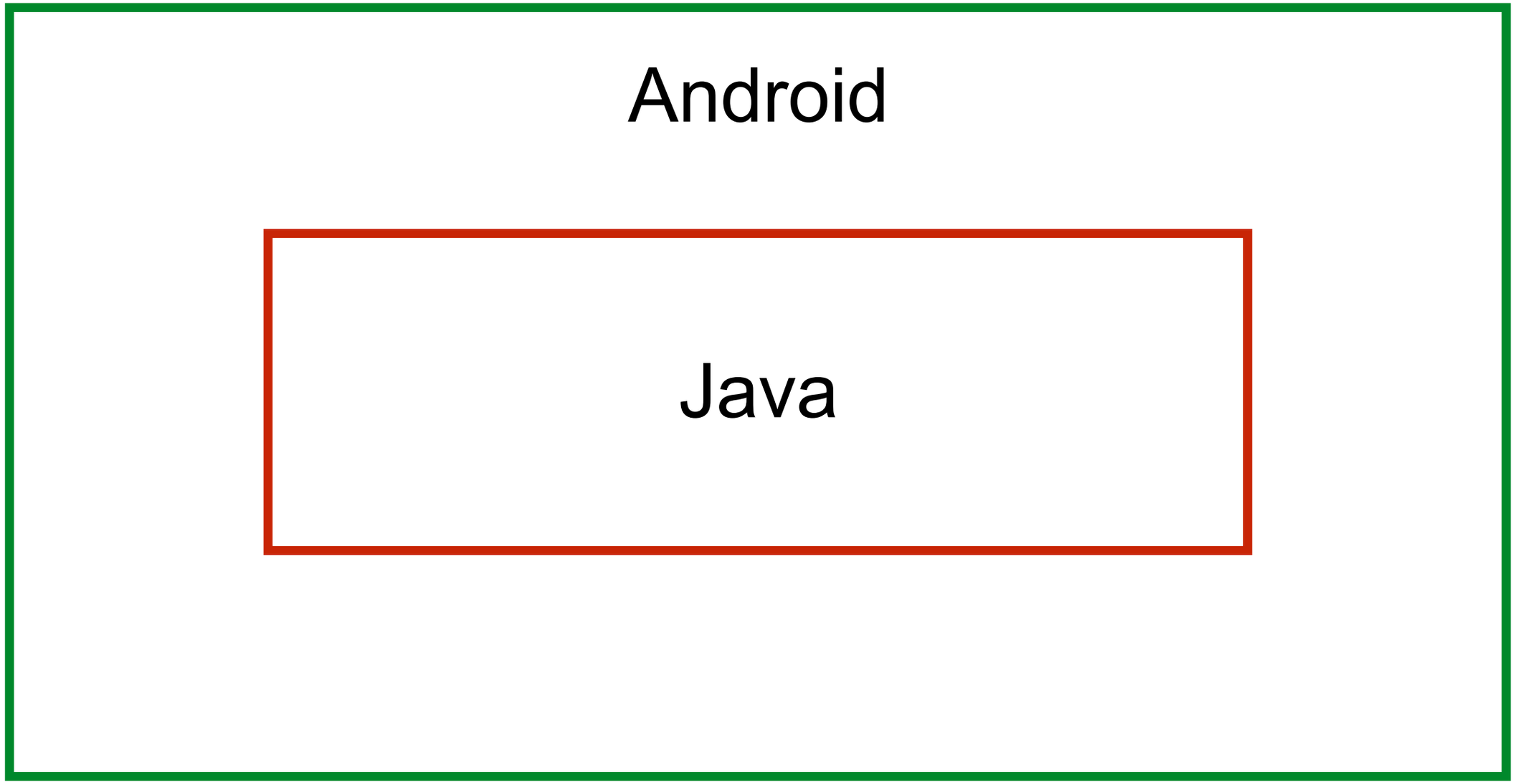


End to End test

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

JVM unit test

/src/test

Business logic with pure java code

Android Testing

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		Instrumentation UI test

/src/test /src/androidTest

Android Testing

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

UI TESTING FOR ANDROID
espresso

/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

Android Testing

	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 and MVP	Instrumentation UI test

/src/test /src/androidTest

Rule of thumb

Instrumentation tests are slower than JVM tests.

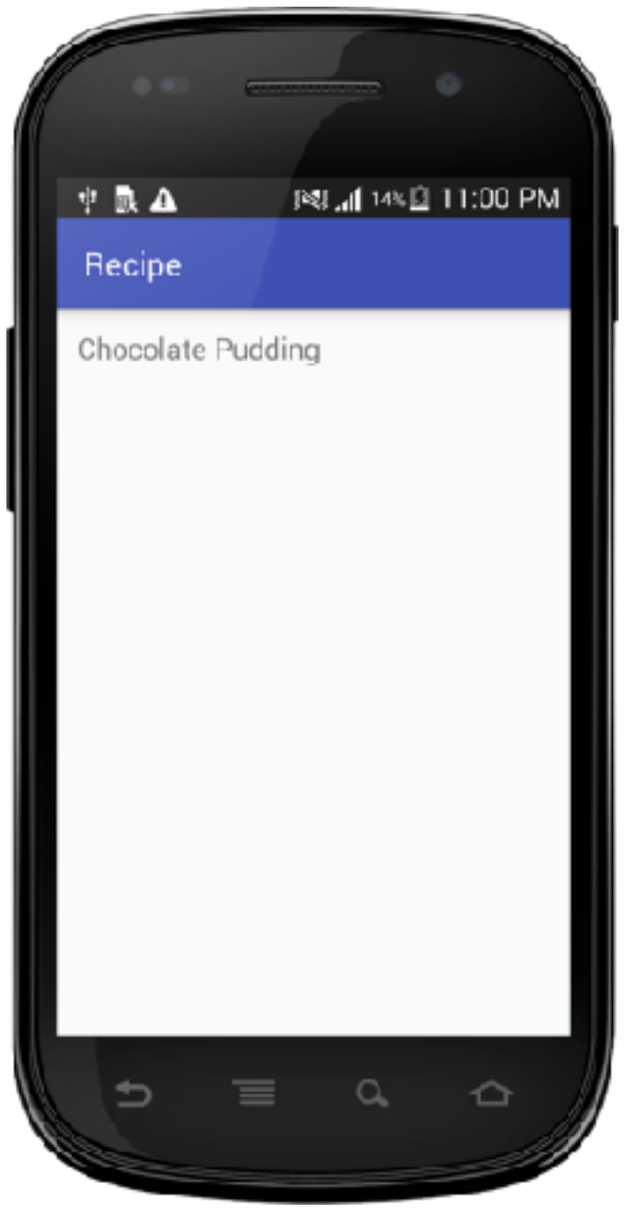
Try to separate the standard Java code from Android-dependent code.

Workshop with Testing

step-by-step to write tests

Recipe application

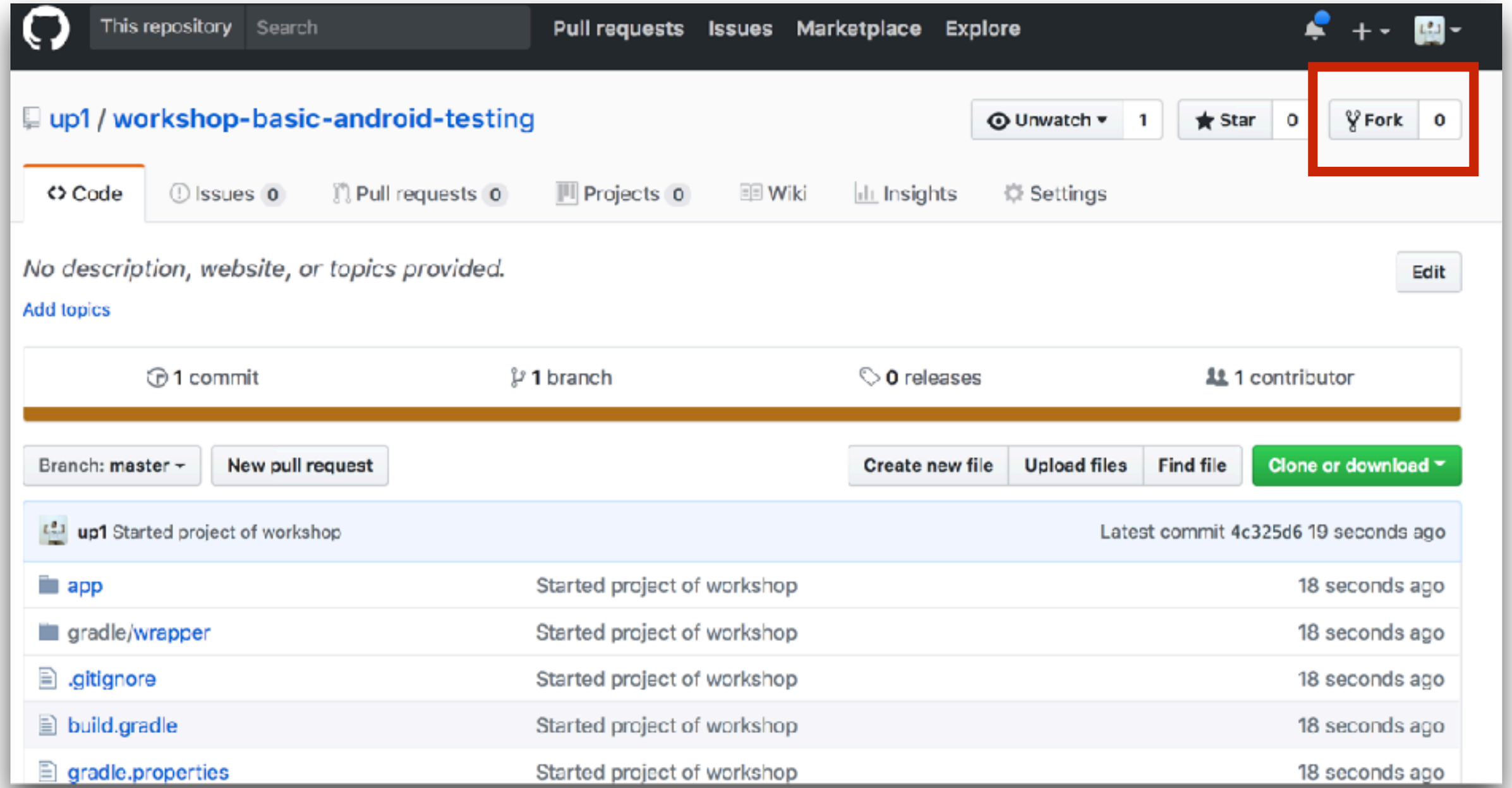
List of recipes



Detail of recipe

Prepare your project

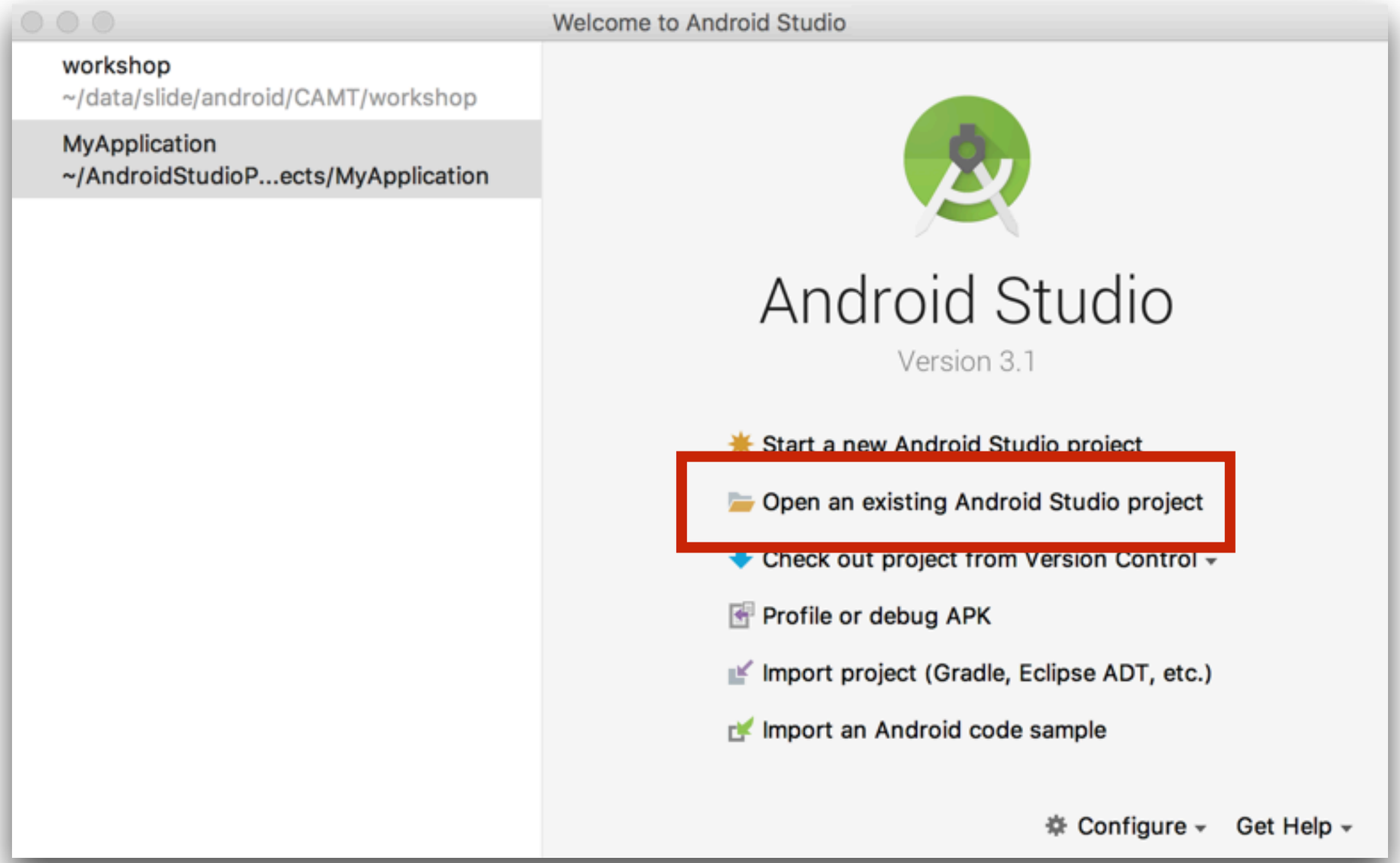
1. Fork from Github repository



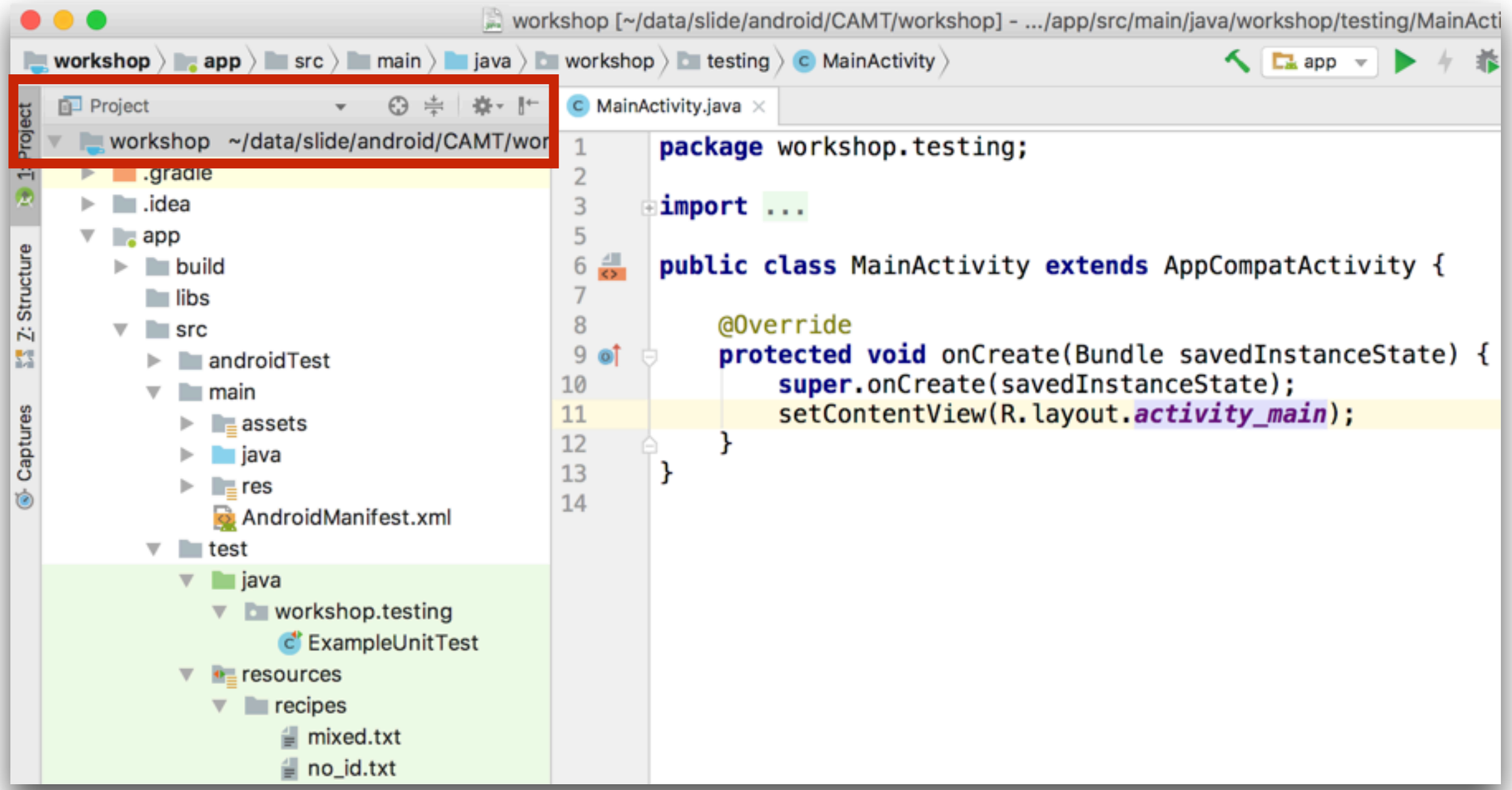
2. Clone from your repository

$git clone https://github.com/**<username>**/workshop-basic-android-testing

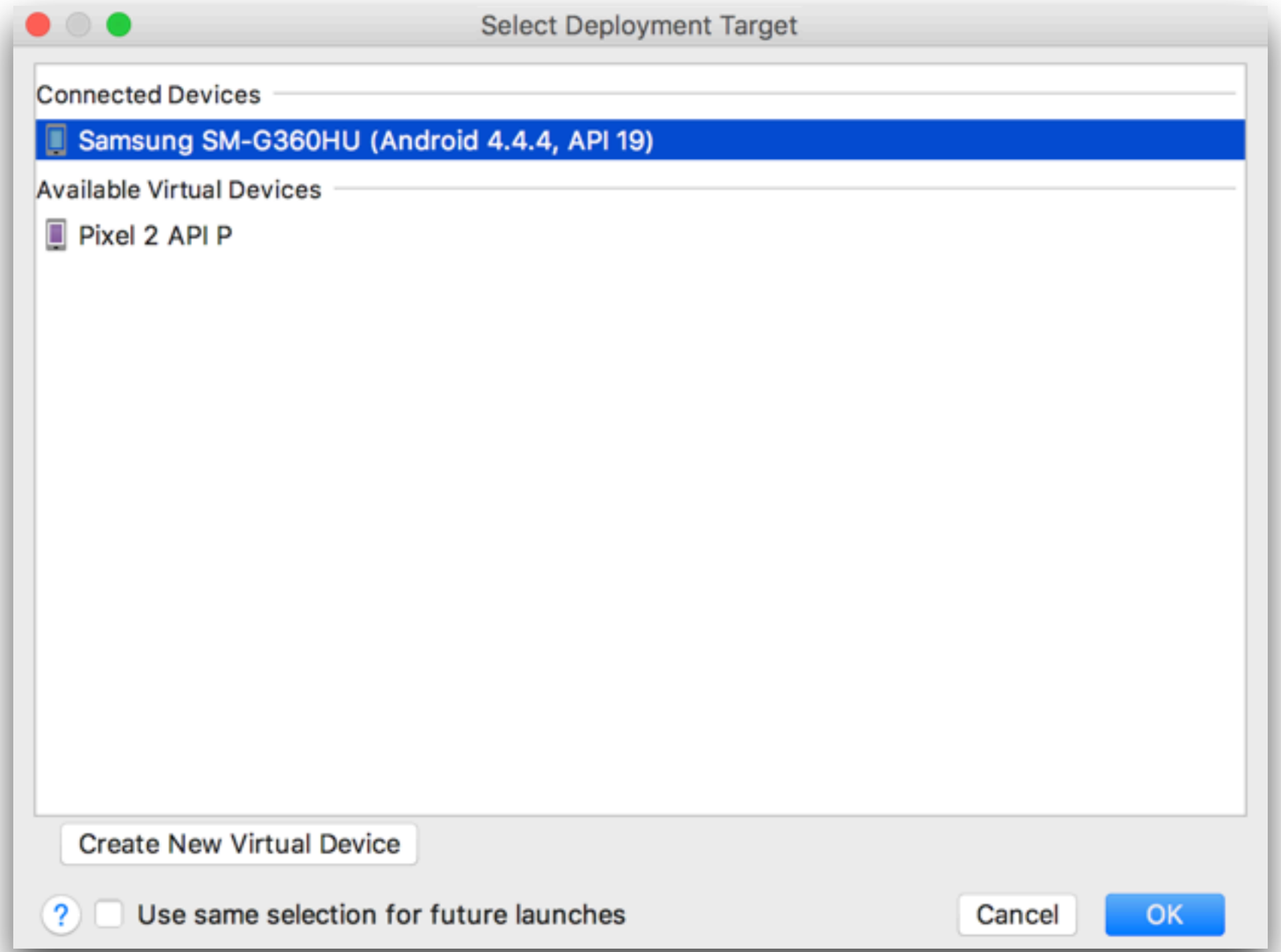
3. Import to Android Studio



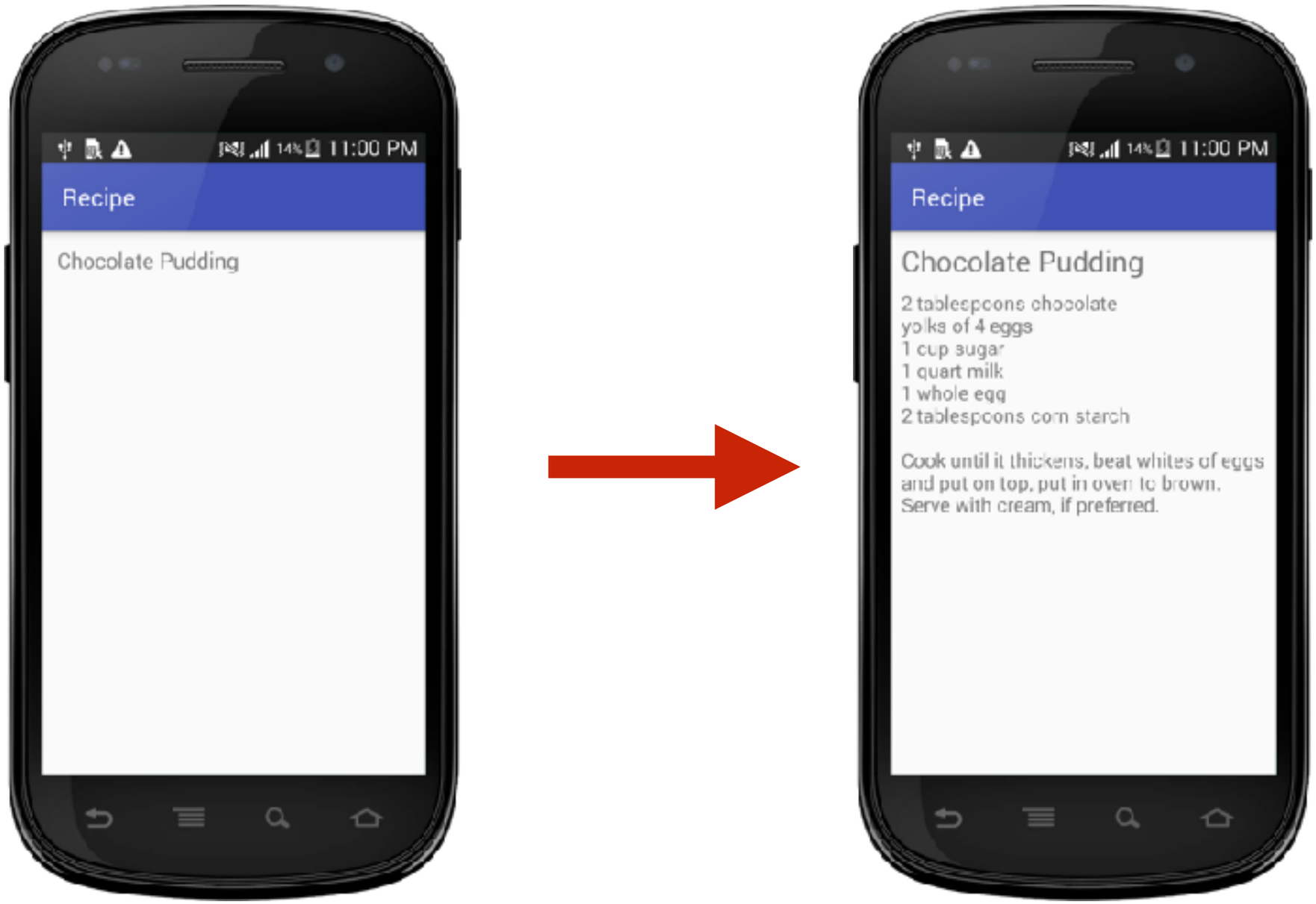
4. Switch to project view



5. Try to run on device/emulator



Ready to start

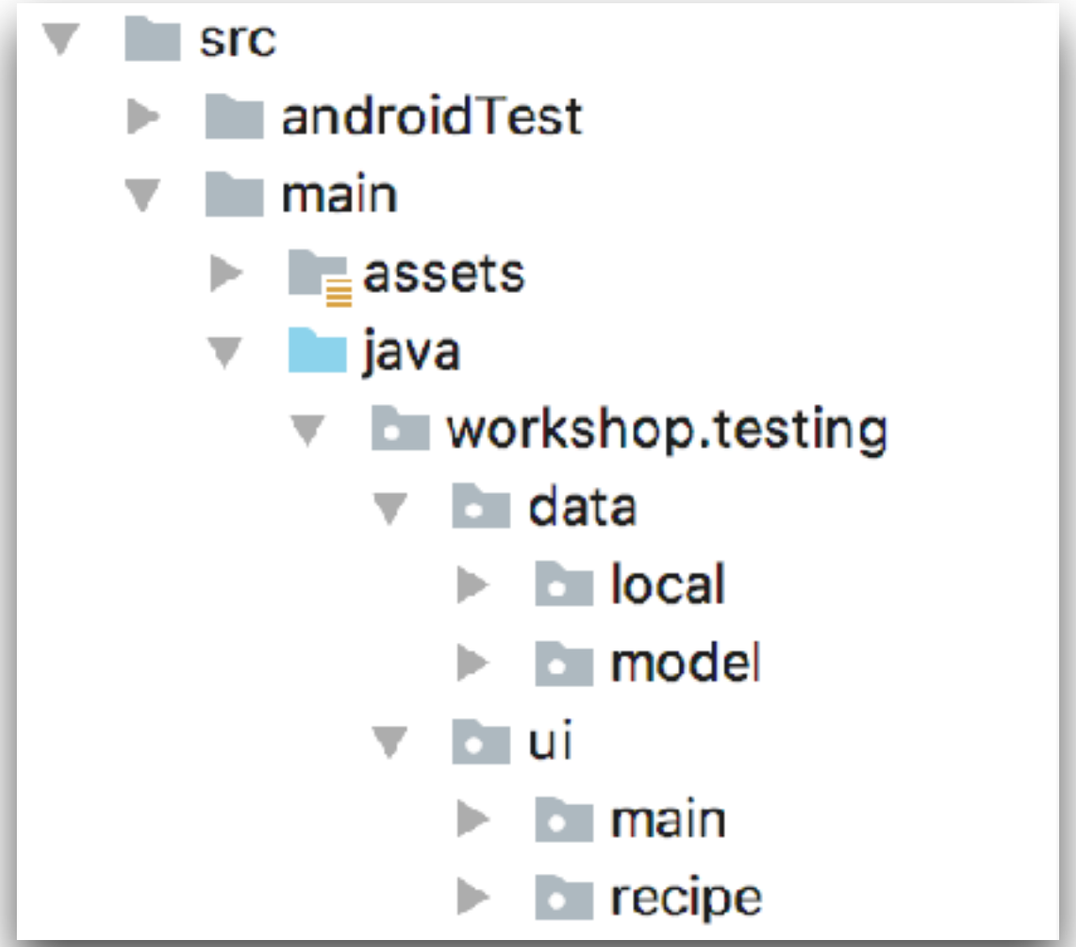


Rule of workshop

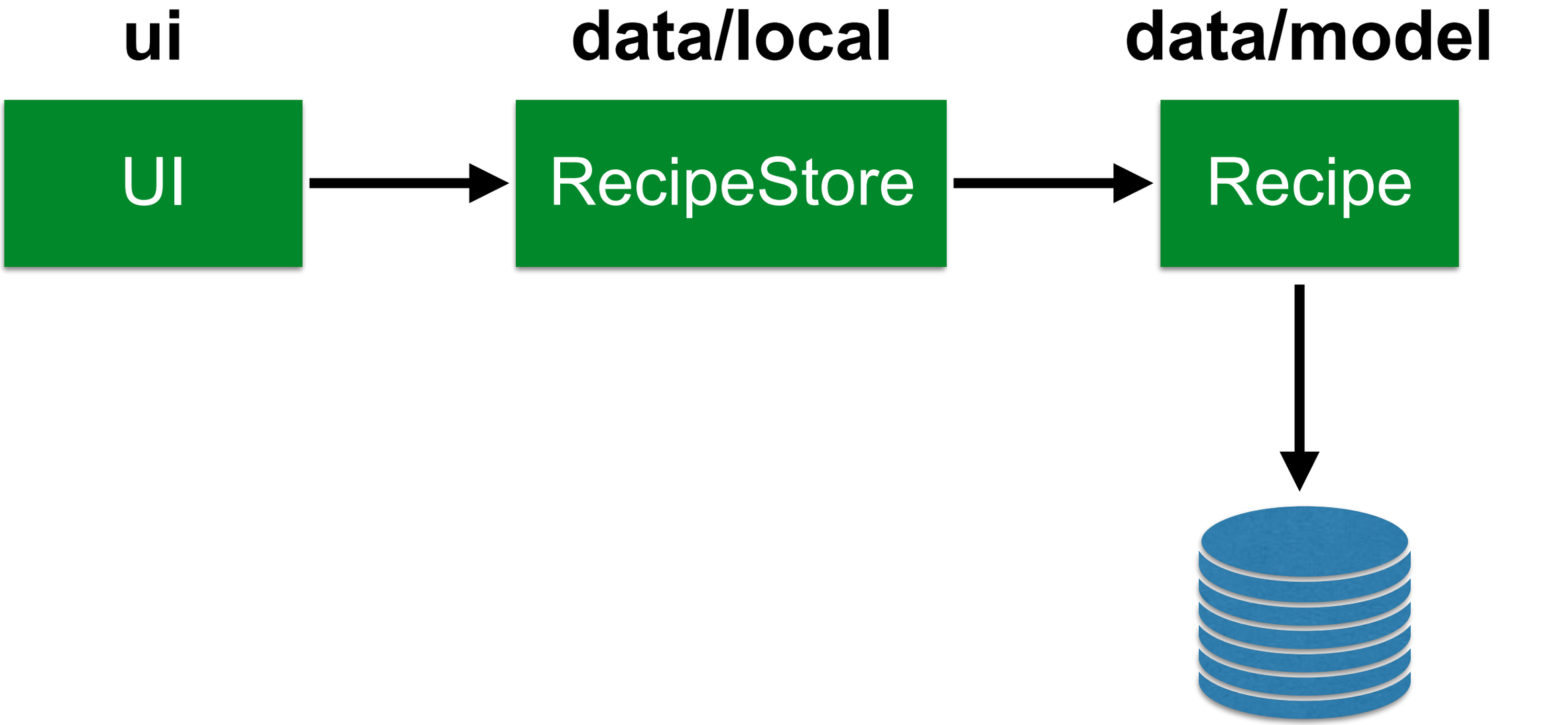
One test case per commit and push to Github repository

Let’ start

Project structure



Project structure



Test structure

```
public class RecipeTest {

    @Test
    public void water() {
        //Arrange
        InputStream stream
                = RecipeTest.class.getResourceAsStream( name: "/recipes/water.txt");

        // Act
        Recipe recipe = Recipe.readFromStream(stream);

        // Assert
        assertNotNull(recipe);
        assertEquals( expected: "water", recipe.id);
        assertEquals( expected: "Water", recipe.title);
        assertEquals( expected: "Put glass under tap. Open tap. Close tap. Drink."
                , recipe.description);
    }
}
```

Test structure

Test class name is a group of tests

```
public class RecipeTest {

    @Test
    public void water() {
        //Arrange
        InputStream stream
                = RecipeTest.class.getResourceAsStream( name: "/recipes/water.txt");

        // Act
        Recipe recipe = Recipe.readFromStream(stream);

        // Assert
        assertNotNull(recipe);
        assertEquals( expected: "water", recipe.id);
        assertEquals( expected: "Water", recipe.title);
        assertEquals( expected: "Put glass under tap. Open tap. Close tap. Drink."
                , recipe.description);
    }
}
```

Test structure

Test annotation of JUnit 4 use to define the method as a test case

```
public class RecipeTest {

    @Test
    public void water() {
        //Arrange
        InputStream stream
                = RecipeTest.class.getResourceAsStream( name: "/recipes/water.txt");

        // Act
        Recipe recipe = Recipe.readFromStream(stream);

        // Assert
        assertNotNull(recipe);
        assertEquals( expected: "water", recipe.id);
        assertEquals( expected: "Water", recipe.title);
        assertEquals( expected: "Put glass under tap. Open tap. Close tap. Drink."
                , recipe.description);
    }
}
```

Test structure

Arrange section to setup data and states of test case

```
public class RecipeTest {

    @Test
    public void water() {
        //Arrange
        InputStream stream
                = RecipeTest.class.getResourceAsStream( name: "/recipes/water.txt");

        // Act
        Recipe recipe = Recipe.readFromStream(stream);

        // Assert
        assertNotNull(recipe);
        assertEquals( expected: "water", recipe.id);
        assertEquals( expected: "Water", recipe.title);
        assertEquals( expected: "Put glass under tap. Open tap. Close tap. Drink."
                , recipe.description);
    }
}
```

Test structure

Act section to call the target method to check and verify behavior

```
public class RecipeTest {

    @Test
    public void water() {
        //Arrange
        InputStream stream
                = RecipeTest.class.getResourceAsStream( name: "/recipes/water.txt");

        // Act
        Recipe recipe = Recipe.readFromStream(stream);

        // Assert
        assertNotNull(recipe);
        assertEquals( expected: "water", recipe.id);
        assertEquals( expected: "Water", recipe.title);
        assertEquals( expected: "Put glass under tap. Open tap. Close tap. Drink."
                , recipe.description);
    }
}
```

Test structure

Assert section to check the result as we expected or not

```
public class RecipeTest {

    @Test
    public void water() {
        //Arrange
        InputStream stream
                = RecipeTest.class.getResourceAsStream( name: "/recipes/water.txt");

        // Act
        Recipe recipe = Recipe.readFromStream(stream);

        // Assert
        assertNotNull(recipe);
        assertEquals( expected: "water", recipe.id);
        assertEquals( expected: "Water", recipe.title);
        assertEquals( expected: "Put glass under tap. Open tap. Close tap. Drink."
                , recipe.description);
    }
}
```

Check code coverage

Code coverage

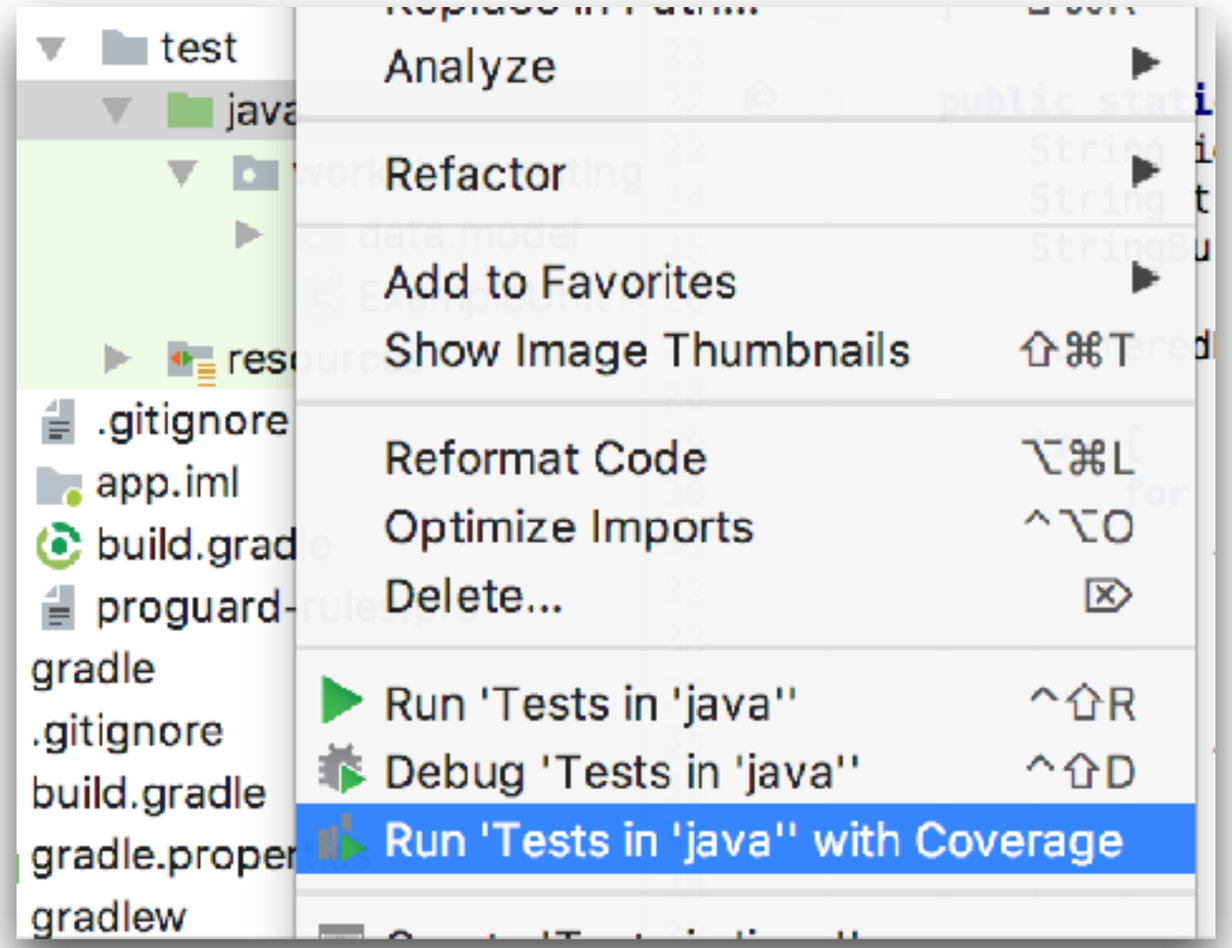
A tool to measure how much of your code is covered by tests that break down into classes, methods and lines.

Code coverage

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

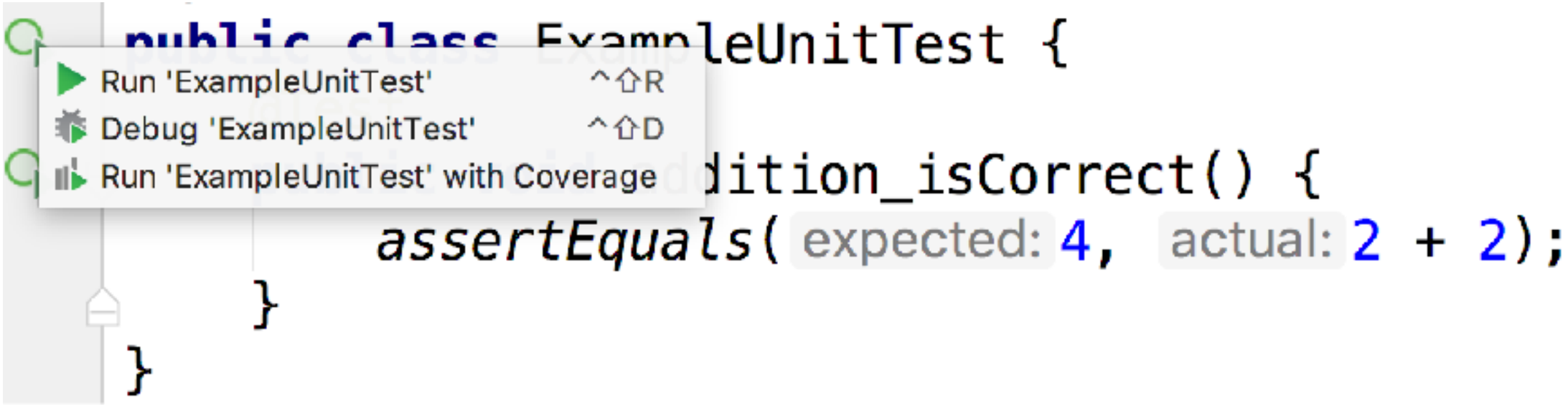
Check code coverage

Right click at test or androidTest directory



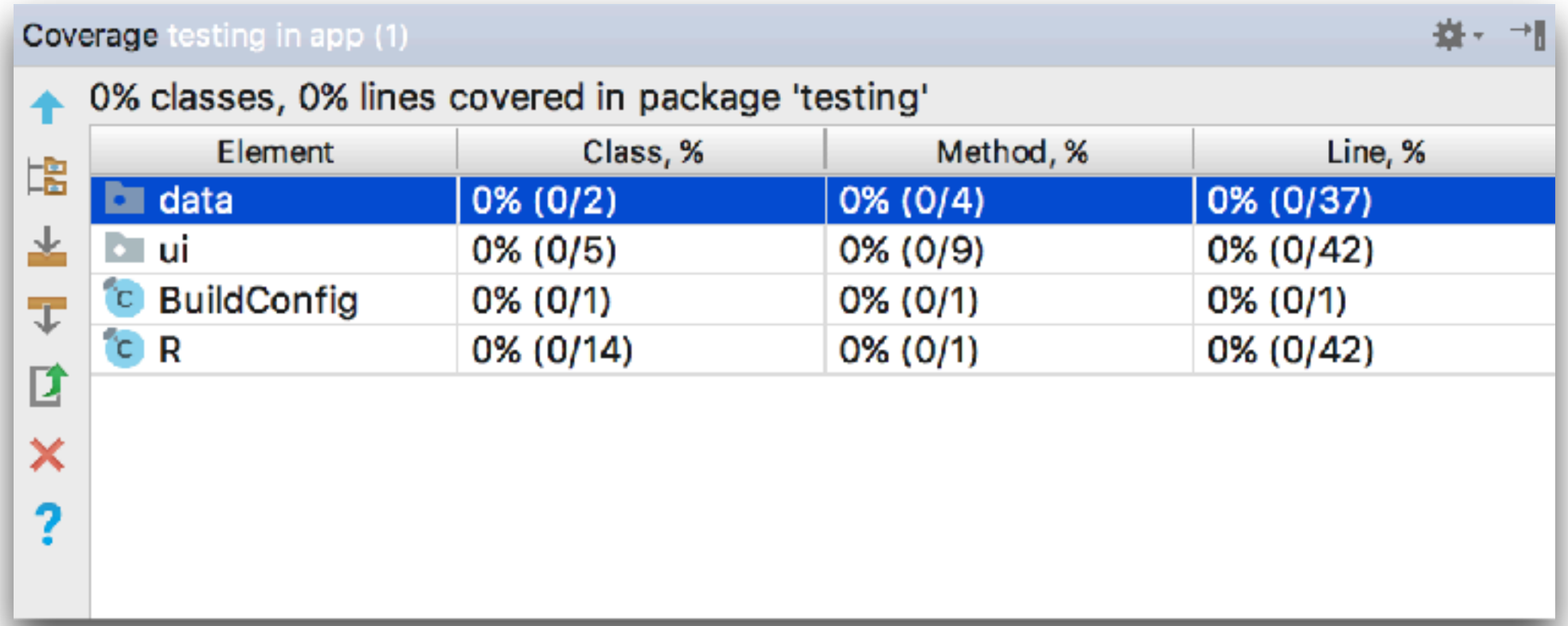
Check code coverage

Right click at test or androidTest directory



Check code coverage

See the result



Check code coverage

Export the result to HTML format

[all classes] [workshop.testing]

Coverage Summary for Package: workshop.testing

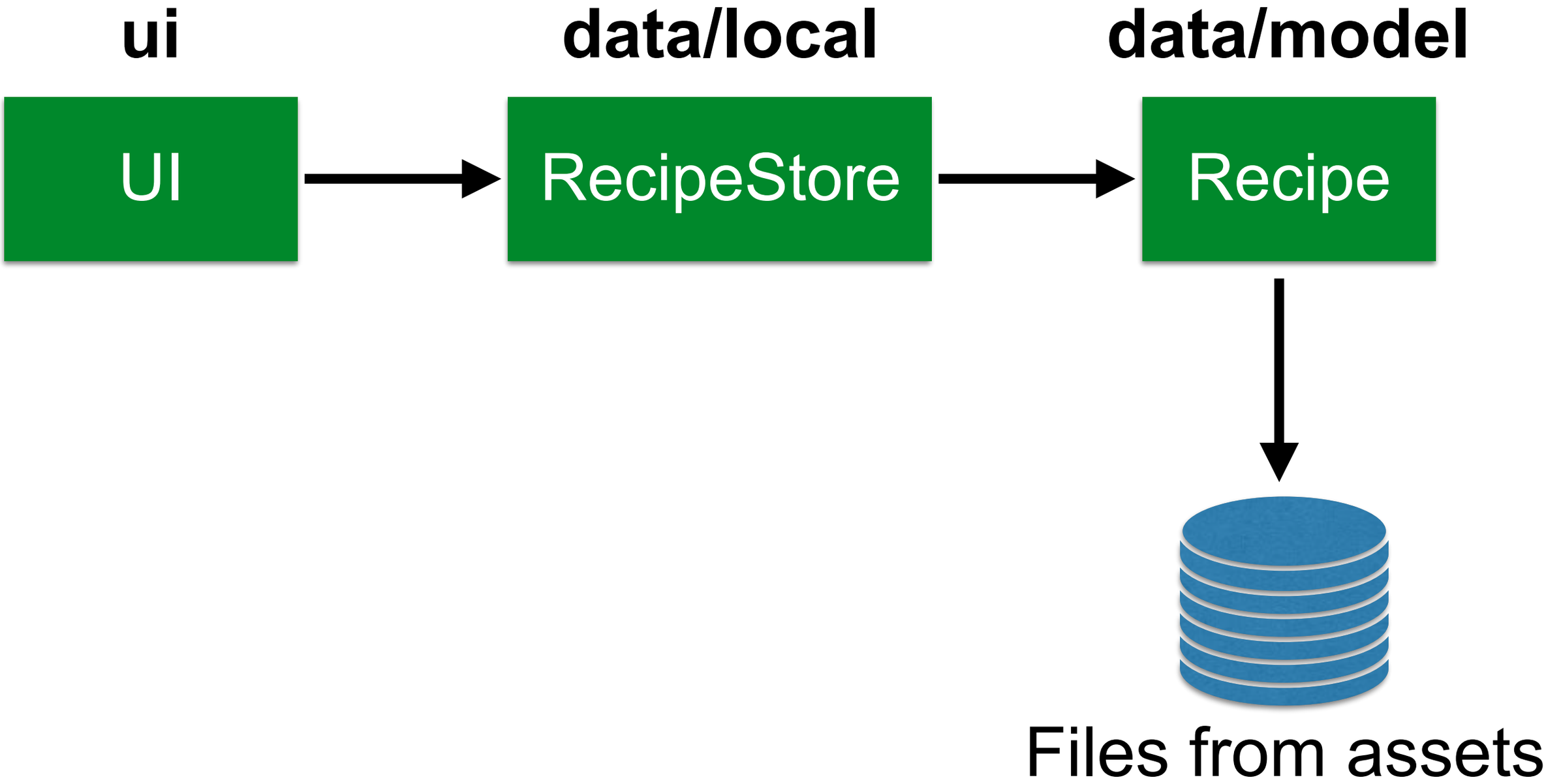
Package	Class, %	Method, %
workshop.testing	0% (0/ 15)	0% (0/ 17)

Class	Class, %	Method, %
BuildConfig	0% (0/ 1)	0% (0/ 2)
R	0% (0/ 14)	0% (0/ 15)

0% is good point to improve !!

Let’ coding with tests

What and Where to test ?



Steps to develop app with tests

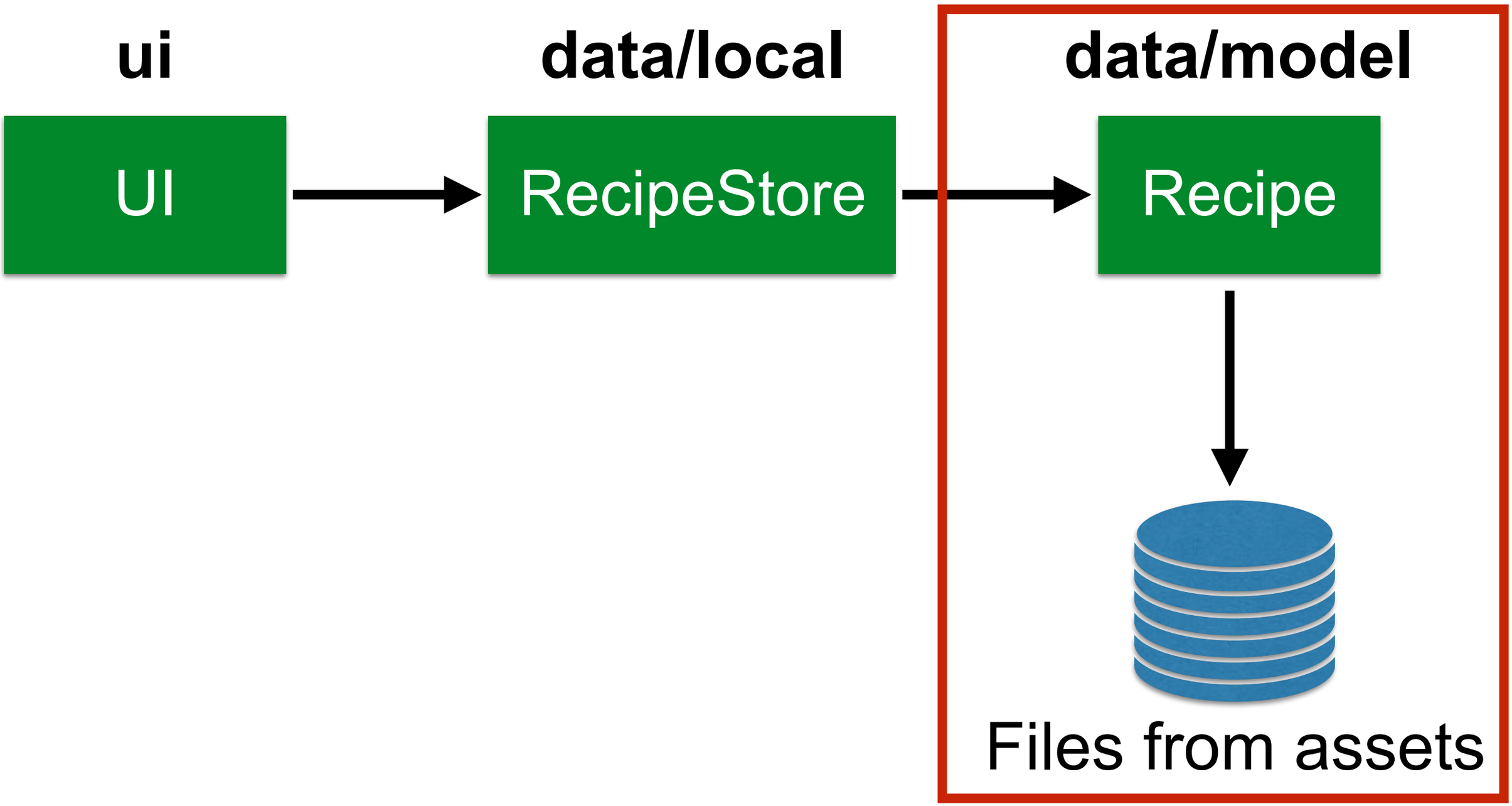
Read data of recipe from file system

Show detail of recipe in Activity

more …

1. Read data of recipe from file system

Read data from file



Read data from file

What type of Android testing ?

Android Testing

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

/src/test /src/androidTest

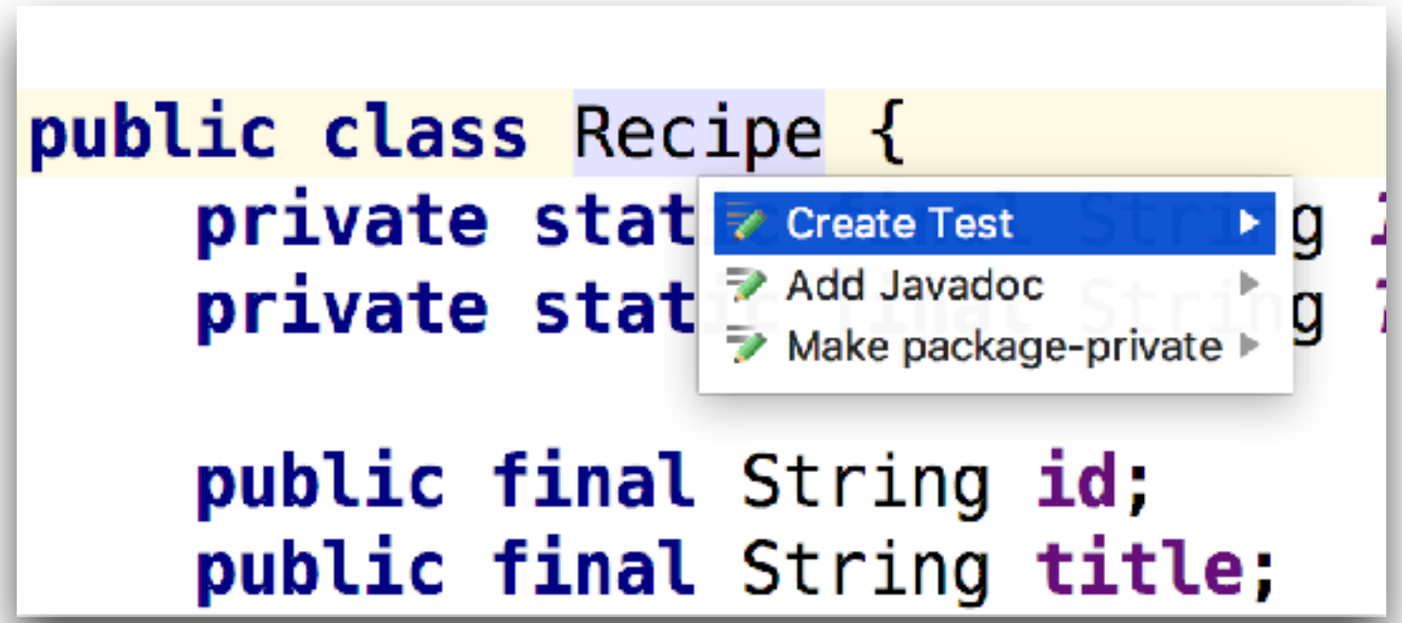
Read data from file

Q: What type of Android testing ?

A: JVM Unit test

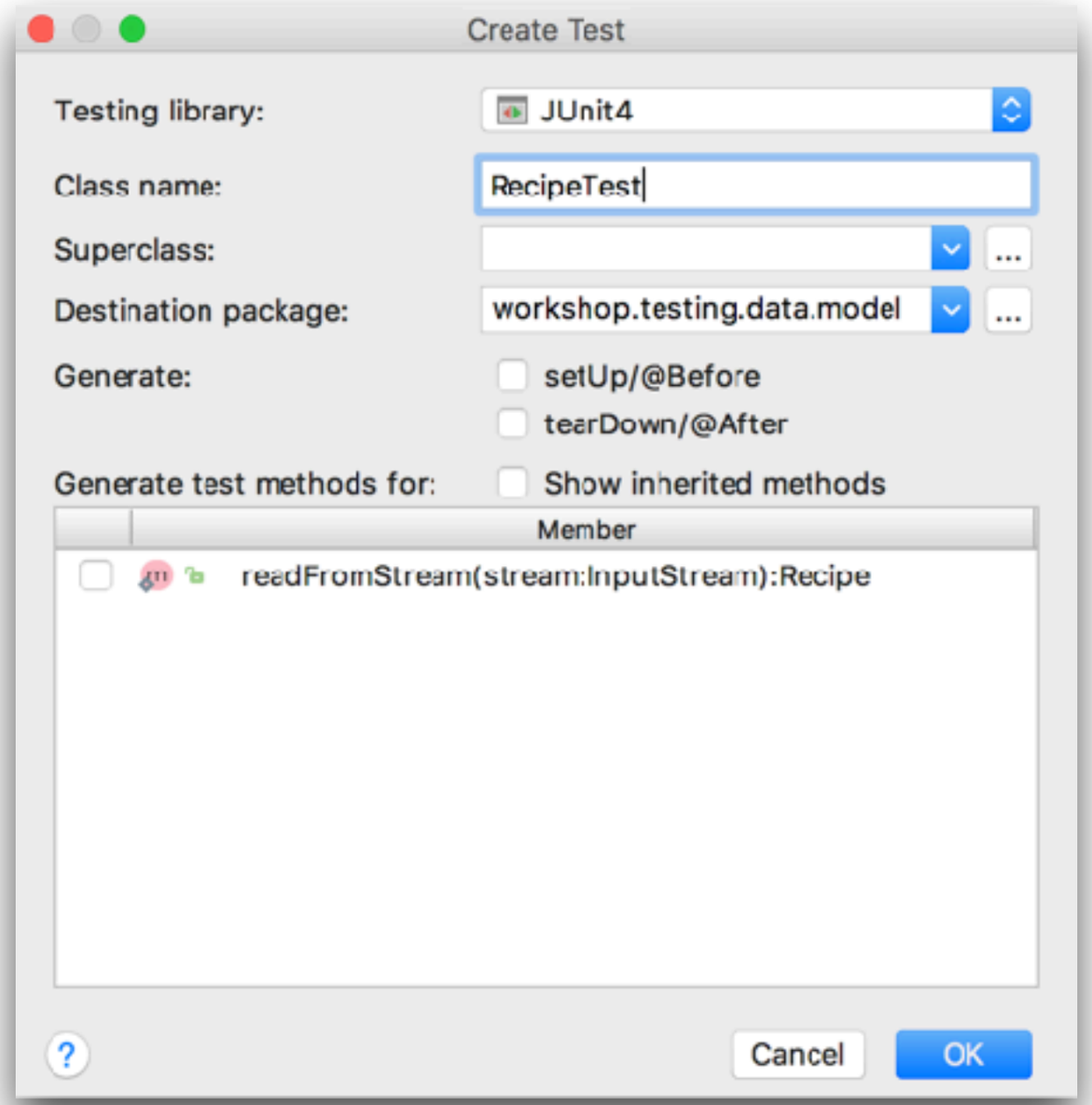
Read data from file

Create the new test class with Recipe
(ALT + Enter)



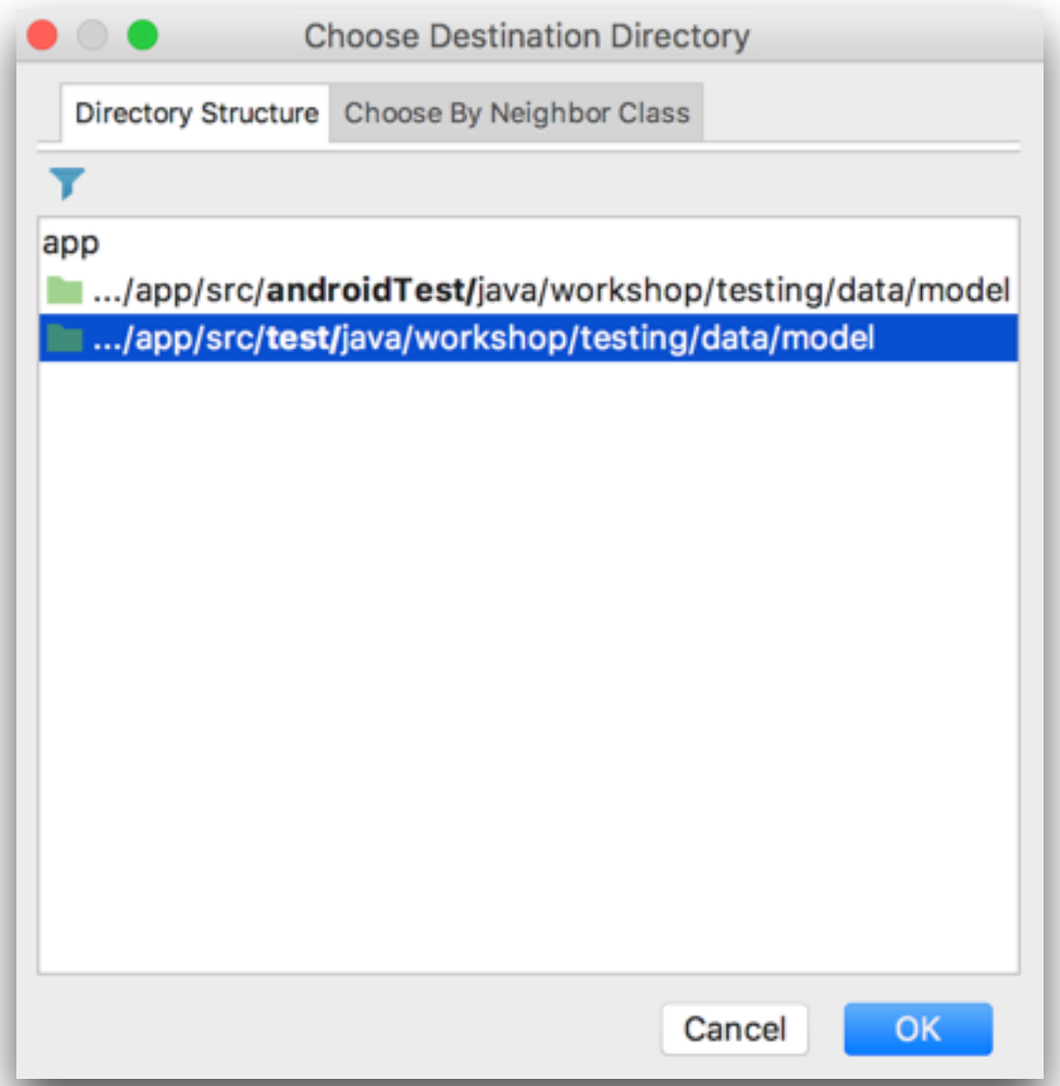
Read data from file

Choose JUnit 4



Read data from file

Choose the destination to **test** directory



Read data from file

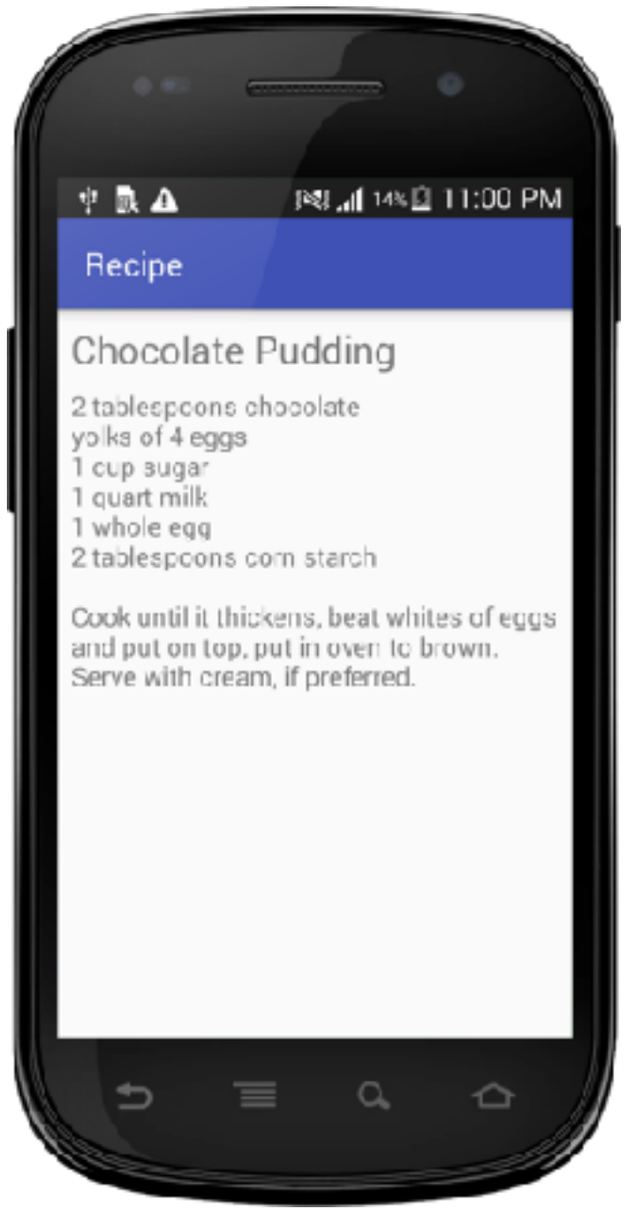
First test case :: read data from water.txt

```
@Test
public void water() {
    InputStream stream
            = RecipeTest.class.getResourceAsStream( name: "/recipes/water.txt");

    Recipe recipe = Recipe.readFromStream(stream);

    assertNotNull(recipe);
    assertEquals( expected: "water", recipe.id);
    assertEquals( expected: "Water", recipe.title);
    assertEquals( expected: "Put glass under tap. Open tap. Close tap. Drink."
            , recipe.description);
}
```

Check code coverage

2. Show detail of recipe in Activity

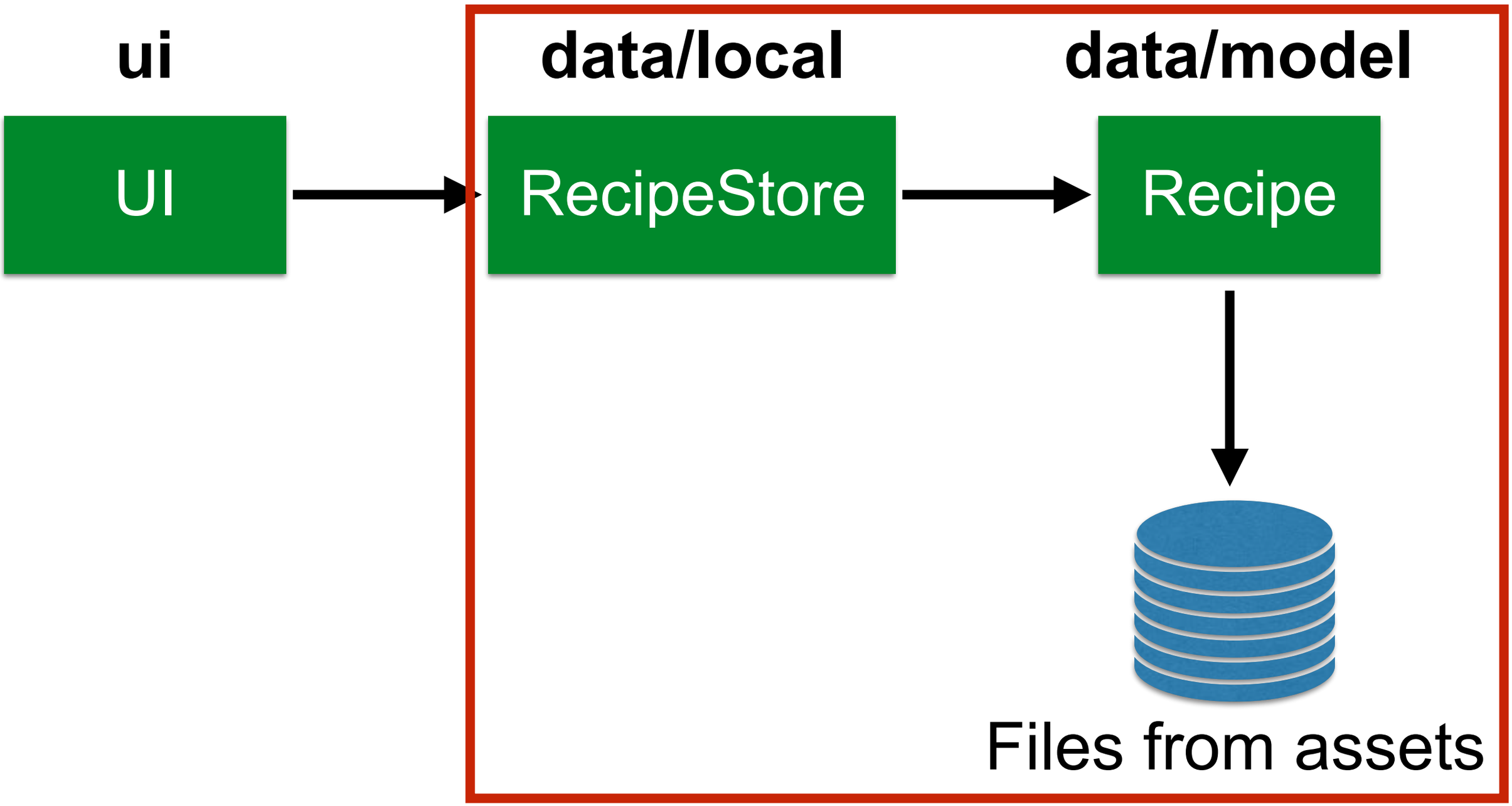
Show detail of recipe



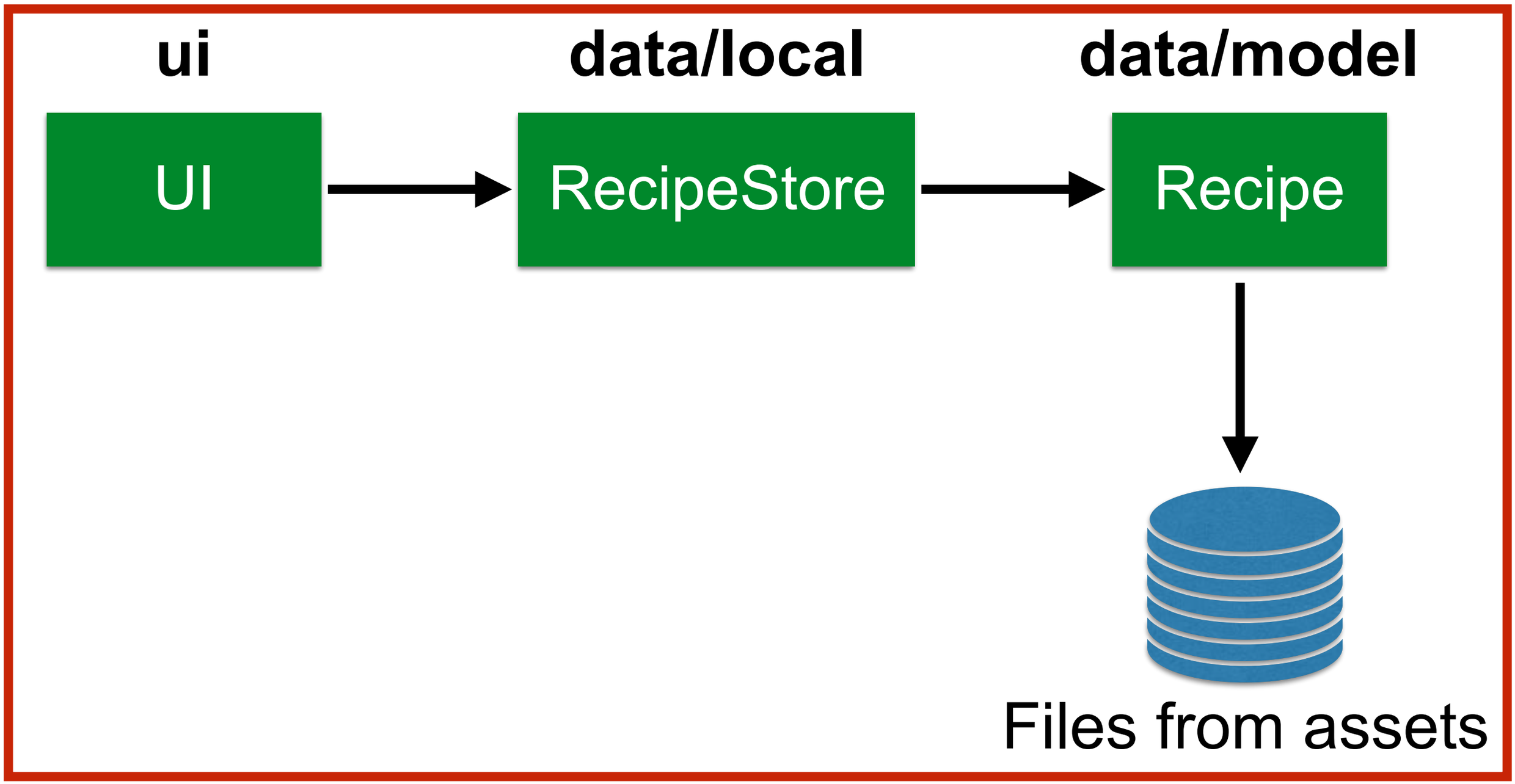
RecipeStore

Android Framework

Show detail of recipe (1)



Show detail of recipe (2)



Show detail of recipe

What type of Android testing ?

Android Testing

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

/src/test /src/androidTest

Show detail of recipe

Q: What type of Android testing ?

A: Separated tests in 2 types

Android Testing

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

/src/test /src/androidTest

1. Instrumentation Unit test

Q: What to test ?

A: Check and verify behavior of RecipeStore

1. Instrumentation Unit test

Q: What to test ?

Number of recipe(s)

Get detail of recipe

1. Instrumentation Unit test

Test case :: number of recipe(s)

```
@Test
public void number_of_recipe() {
    Context context = InstrumentationRegistry.getTargetContext();
    RecipeStore store = new RecipeStore(context, directory: "recipes");
    assertNotNull(store);
    assertNotNull(store.recipes);
    assertEquals( expected: 1, store.recipes.size());
}
```

Check code coverage

2. Instrumentation UI test

Q: What to test ?

A: Choose a recipe and show detail in Activity

2. Instrumentation UI test

Test case :: show detail of recipe in Activity

```
@Rule
public ActivityTestRule<RecipeActivity> activityRule
        = new ActivityTestRule<>(
        RecipeActivity.class,  initialTouchMode: true,  launchActivity: false);

@Test
public void show_detail_of_chocolate_pudding() {
    Intent intent = new Intent();
    intent.putExtra(RecipeActivity.KEY_ID,  value: "chocolate_pudding");
    activityRule.launchActivity(intent);

    onView(withId(R.id.title))
            .check(matches(withText("Chocolate Pudding")));
    onView(withId(R.id.description))
            .check(matches(withText("2 tablespoons chocolate\n" +
                    "yolks of 4 eggs\n" +
                    "1 cup sugar\n" +
                    "1 quart milk\n" +
                    "1 whole egg\n" +
                    "2 tablespoons corn starch\n" +
                    "\n" +
                    "Cook until it thickens, beat whites of eggs and pu
}
```

Step 1 :: Start activity

```
@Rule
public ActivityTestRule<RecipeActivity> activityRule
        = new ActivityTestRule<>(
        RecipeActivity.class, initialTouchMode: true, launchActivity: false);
```

```
@Test
public void show_detail_of_chocolate_pudding() {
    Intent intent = new Intent();
    intent.putExtra(RecipeActivity.KEY_ID, value: "chocolate_pudding");
    activityRule.launchActivity(intent);

    onView(withId(R.id.title))
            .check(matches(withText("Chocolate Pudding")));
    onView(withId(R.id.description))
            .check(matches(withText("2 tablespoons chocolate\n" +
                    "yolks of 4 eggs\n" +
                    "1 cup sugar\n" +
                    "1 quart milk\n" +
                    "1 whole egg\n" +
                    "2 tablespoons corn starch\n" +
                    "\n" +
                    "Cook until it thickens, beat whites of eggs and pu
}
```

Step 2 :: Create new test case

```
@Rule
public ActivityTestRule<RecipeActivity> activityRule
        = new ActivityTestRule<>(
        RecipeActivity.class, initialTouchMode: true, launchActivity: false);
```

```
@Test
public void show_detail_of_chocolate_pudding() {
    Intent intent = new Intent();
    intent.putExtra(RecipeActivity.KEY_ID, value: "chocolate_pudding");
    activityRule.launchActivity(intent);

    onView(withId(R.id.title))
            .check(matches(withText("Chocolate Pudding")));
    onView(withId(R.id.description))
            .check(matches(withText("2 tablespoons chocolate\n" +
                    "yolks of 4 eggs\n" +
                    "1 cup sugar\n" +
                    "1 quart milk\n" +
                    "1 whole egg\n" +
                    "2 tablespoons corn starch\n" +
                    "\n" +
                    "Cook until it thickens, beat whites of eggs and pu
}
```

Step 3 :: Pass data with intent

```
@Rule
public ActivityTestRule<RecipeActivity> activityRule
        = new ActivityTestRule<>(
        RecipeActivity.class, initialTouchMode: true, launchActivity: false);

@Test
public void show_detail_of_chocolate_pudding() {
    Intent intent = new Intent();
    intent.putExtra(RecipeActivity.KEY_ID, value: "chocolate_pudding");
    activityRule.launchActivity(intent);

    onView(withId(R.id.title))
            .check(matches(withText("Chocolate Pudding")));
    onView(withId(R.id.description))
            .check(matches(withText("2 tablespoons chocolate\n" +
                    "yolks of 4 eggs\n" +
                    "1 cup sugar\n" +
                    "1 quart milk\n" +
                    "1 whole egg\n" +
                    "2 tablespoons corn starch\n" +
                    "\n" +
                    "Cook until it thickens, beat whites of eggs and pu
}
```

Step 4 :: Verify data in activity

```
@Rule
public ActivityTestRule<RecipeActivity> activityRule
        = new ActivityTestRule<>(
        RecipeActivity.class, initialTouchMode: true, launchActivity: false);

@Test
public void show_detail_of_chocolate_pudding() {
    Intent intent = new Intent();
    intent.putExtra(RecipeActivity.KEY_ID, value: "chocolate_pudding");
    activityRule.launchActivity(intent);

    onView(withId(R.id.title))
            .check(matches(withText("Chocolate Pudding")));
    onView(withId(R.id.description))
            .check(matches(withText("2 tablespoons chocolate\n" +
                    "yolks of 4 eggs\n" +
                    "1 cup sugar\n" +
                    "1 quart milk\n" +
                    "1 whole egg\n" +
                    "2 tablespoons corn starch\n" +
                    "\n" +
                    "Cook until it thickens, beat whites of eggs and pu
}
```

Check code coverage

Assignments/Homework

Assignments/Homework

List of recipes from assets

Add more tests/more code coverage

Add more features such as Favorite

Push all changes to your Github repository

Summary of this course

Resources

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

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

Test-Driven Development