Testing

Sang Shin
Michèle Garoche
www.javapassion.com
“Learn with Passion!”

Disclaimer

- Portions of this presentation are modifications based on work created and shared by the Android Open Source Project
  - http://code.google.com/policies.html
- They are used according to terms described in the Creative Commons 2.5 Attribution License
  - http://creativecommons.org/licenses/by/2.5/

Topics

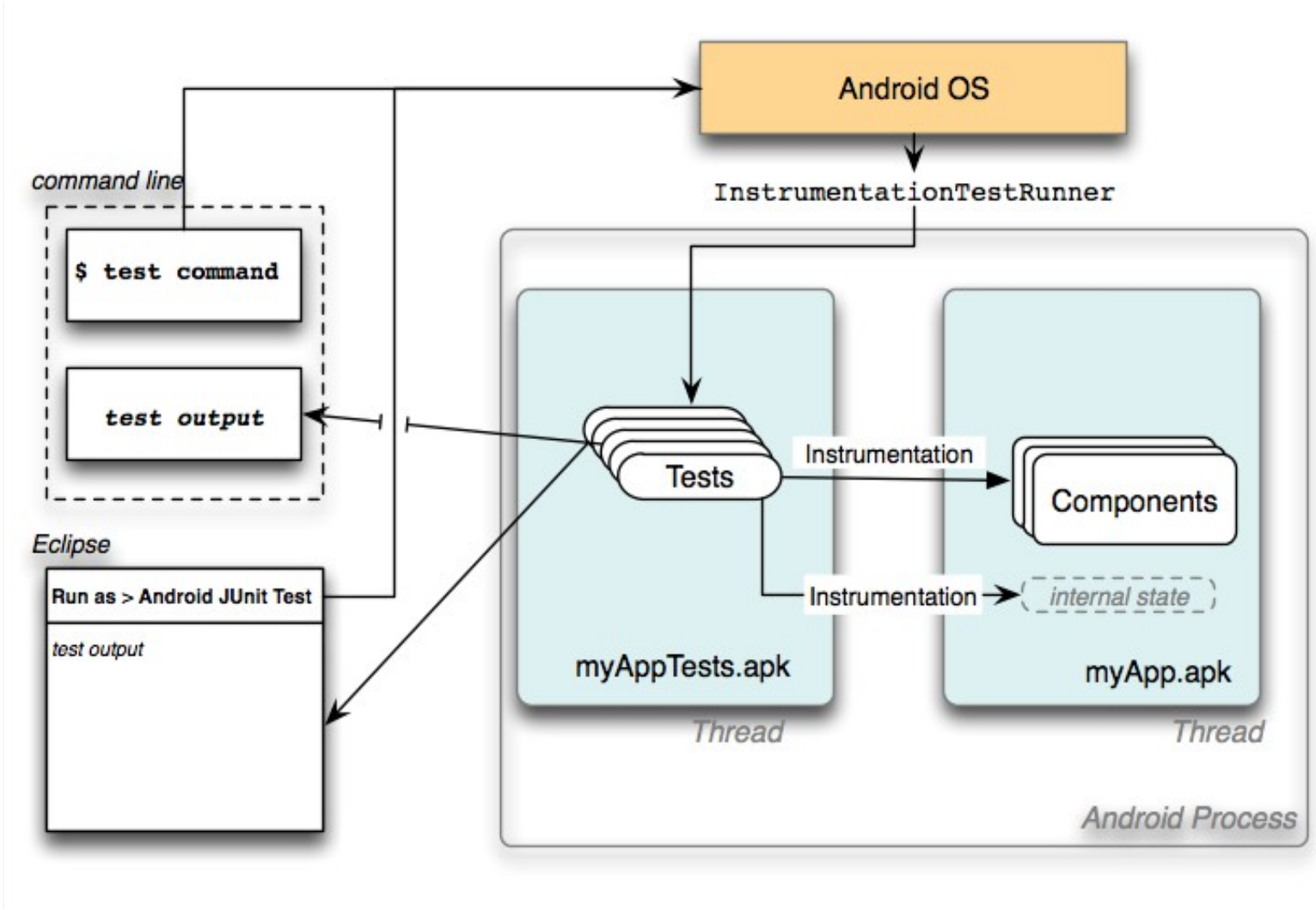
- Android testing framework
- Types of testing methods
- Testing APIs

Android Testing Framework

Android Testing Environment

- Android extensions to the JUnit framework that provide access to Android system objects.
- An instrumentation framework that lets tests control and examine the application.
- Mock versions of commonly-used Android system objects.
- Tools for running single tests or test suites, with or without instrumentation.
- Support for managing tests and test projects in the ADT Plugin for Eclipse and at the command line.

Android Testing Framework



Test Application

- Test application itself is an Android application
- You can create Test application when you create target application or after target application is created

Test application Manifest file

- Your test application is linked to the target application under test by means of an <instrumentation> element in the test application's manifest file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.javapassion.test" android:versionCode="1"
          android:versionName="1.0">
<application android:icon="@drawable/icon"
             android:label="@string/app_name">
    <uses-library android:name="android.test.runner" />
</application>
<uses-sdk android:minSdkVersion="8" />
<instrumentation android:targetPackage="com.javapassion"
           android:name="android.test.InstrumentationTestRunner" />
</manifest>
```

Types of Test Methods

Types of test methods

- Initial conditions test
- UI test
- State management test

Initial Conditions Test

- Tests that the application under test initializes correctly.
- This is also a unit test of the application's *onCreate()* method.
- Testing initial conditions also provides a confidence measure for subsequent tests.

UI test

- Tests that the main UI operation works correctly.
- This test demonstrates the instrumentation features available in activity testing.
- It shows that you can automate UI tests by sending key events from the test application to the main application.

State management tests

- Test the application's code for saving state.
- This test demonstrates the instrumentation features of the test runner, which are available for testing any component.

Testing API

JUnit test case classes

- Extend the JUnit TestCase but do not use the instrumentation framework.
- Contain methods for accessing system objects such as the Context of the application under test.
  - With this Context, you can look at its resources, files, databases, and so forth.
- The base class is *AndroidTestCase*, but you usually use a subclass associated with a particular component.

Subclasses of AndroidTestCase class

- ApplicationTestCase
  - A class for testing an entire application.
- ProviderTestCase2
  - A class for isolated testing of a single ContentProvider
- ServiceTestCase
  - A class for isolated testing of a single Service

ActivityInstrumentationTestCase2

- Subclass of *ActivityTestCase* class
- Provides functional testing of a single activity.
  - The activity under test will be created using the system infrastructure (by calling InstrumentationTestCase.launchActivity()) and you will then be able to manipulate your Activity directly
- Protected methods of this class
  - runTest(), setUp(), tearDown()
- Public methods of this class
  - getActivity(), setActivityInitialTouchMode(boolean initialTouchMode), setActivityIntent(Intent i)

Thank you!

Check JavaPassion.com Codecamps!
http://www.javapassion.com/codecamps
"Learn with Passion!"