Understanding Instrumented Testing

Jim Wilson
MOBILE SOLUTIONS DEVELOPER & ARCHITECT

@hedgehogjim blog.jwhh.com

What to Expect from This Module

- Instrumented Testing Overview
- Implementing Instrumented Tests
- Setting Up UI Tests
- Basic UI Test Interactions
- Testing AdapterViews
- Verifying Test Behavior
- Espresso Test Recorder

Android Testing

Android applications

- Java-based behavior
- Android-based behavior

Testing Java-based behavior

- Local JVM tests

Testing Android-based behavior

- Instrumented tests

Instrumented Tests

Run on an emulator or physical device

- Have the full Android environment

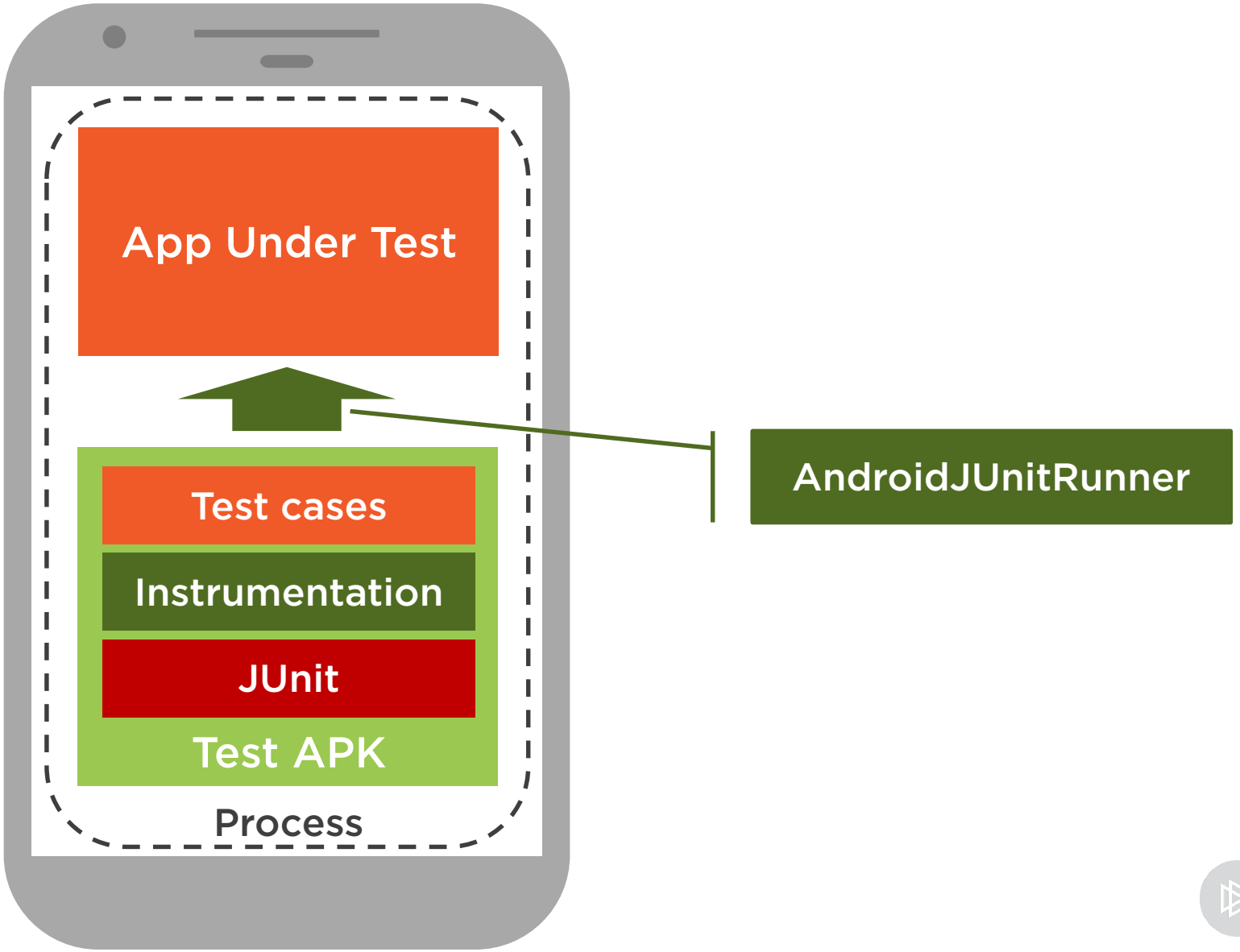
Instrumented Unit Tests

- Unit tests
- Rely on Android features/capabilities

Automated User Interface Tests

- Integration tests
- App behaviors in response to UI actions

Instrumented Tests



Implementing Instrumented Tests

Uses JUnit 4

- Test methods marked with @Test
- Supports pre/post-processing methods

Uses Assert class

- Indicate expectations
- Fails test if expectations not met

Test managed with Android Studio

- Can run or debug tests
  - Single test, group of tests, or all tests
- Displays tests results

Implementing Instrumented Tests

Organized separate from JVM tests

- In androidTest source set

Relies on Android JUnit test runner

- Class must have @RunWith annotation
  - Pass AndroidJUnit4.class

Requires Android environment

- Run on emulator or device

Implementing Instrumented Tests

```
@RunWith(AndroidJUnit4.class)
public class MyExampleTestClass {
  @BeforeClass
  public static void classSetUp() { . . . }
  @Before
  public void testSetUp() { . . . }
  @Test
  public void myTestMethod() {
    // Android dependent test code
  }
}
```

Creating UI Test Interactions

UI tests require a series of view interactions

- Need way to specify view of interest
- Need way to specify action on the view

Espresso.onView method

- Accepts a Matcher parameter
  - Specifies view matching criteria
- Returns a ViewInteraction reference
  - Associated with matching view
  - Used to perform action on view

Specifying View of Interest

Uses Hamcrest matchers

- Provides declarative matching
- General purpose Java framework
- http://hamcrest.org

ViewMatchers class

- Provides matchers for Android Views
- Methods return a Hamcrest matcher
- Easily combined with Hamcrest general purpose matchers

Specifying View of Interest

Example ViewMatchers methods

- withId
  - Match views based on id property
- withText
  - Match views based on text property
- isDisplayed
  - Match views currently on screen
- isChecked
  - Match currently checked checkable views (Switch, CheckBox, etc.)

Specifying View of Interest

Example Hamcrest Matchers

- equalTo
  - Match based on equals method
- instanceOf
  - Match based on class type
- allOf
  - Accepts multiple Matchers
  - Match if all Matchers match
- anyOf
  - Accepts multiple Matchers
  - Match if any Matchers match

Performing View Action

ViewInteraction.perform method

- Performs one or more specified actions
- Specific action passed as a parameter

ViewActions class

- Provides action methods
- Each method returns specified action

Performing View Action

Example ViewActions methods

- click
  - Click on the view
- typeText
  - Type text into view
- replaceText
  - Replace view's text
- closeSoftKeyboard
  - Closes the soft keyboard

Starting the Target Activity

ActivityTestRule

- Automates test activity lifetime
- Starts activity before each test
- Terminates activity after each test
- Activity life includes @Before/@After methods

Using ActivityTestRule

- Declare & initialize as test class field
- Desired activity as type parameter
- Mark field with @Rule annotation

Testing Views That Use Adapters

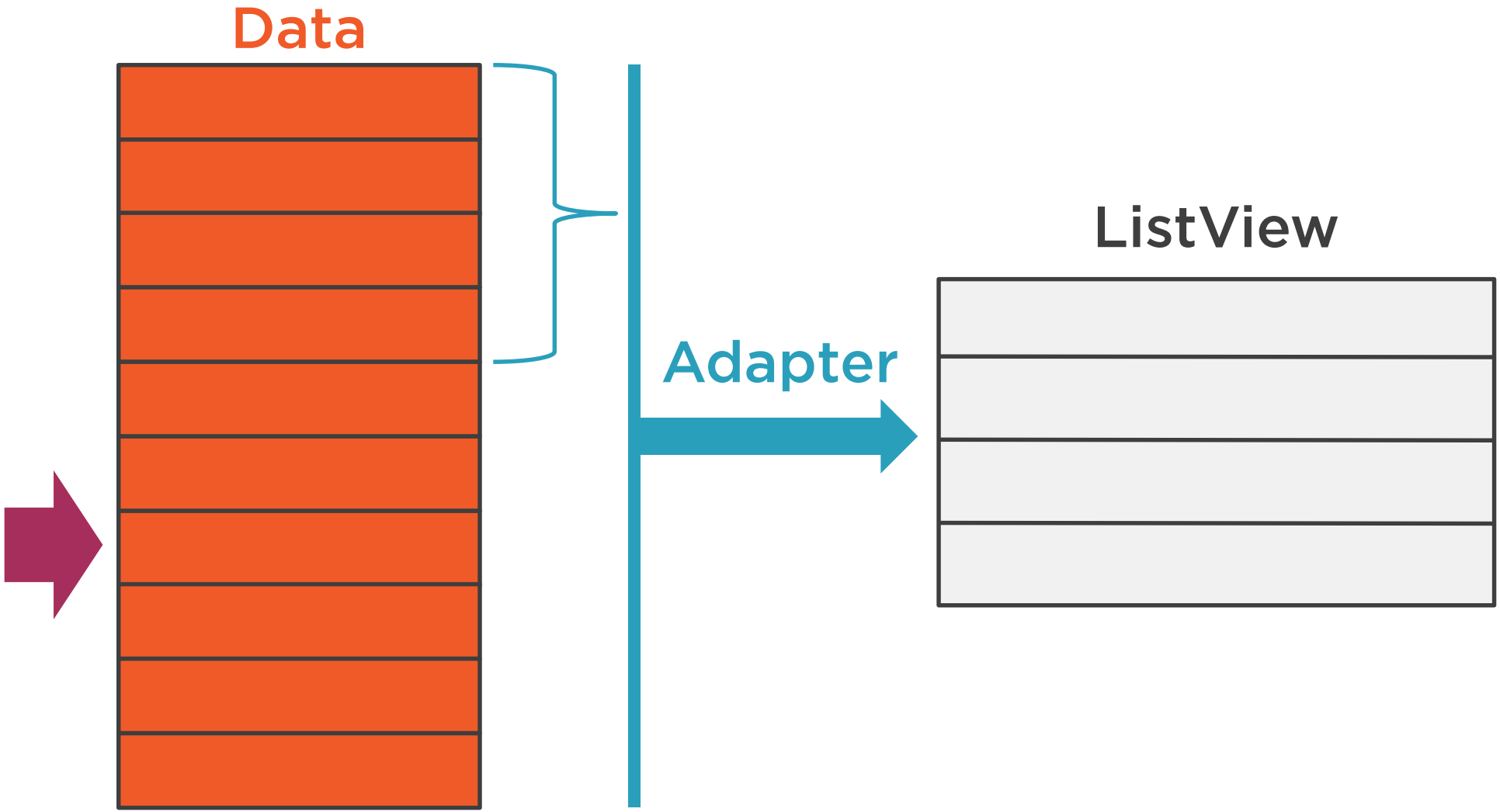
AdapterView derived views

- Load data from Adapter classes
- Examples include ListView & Spinner

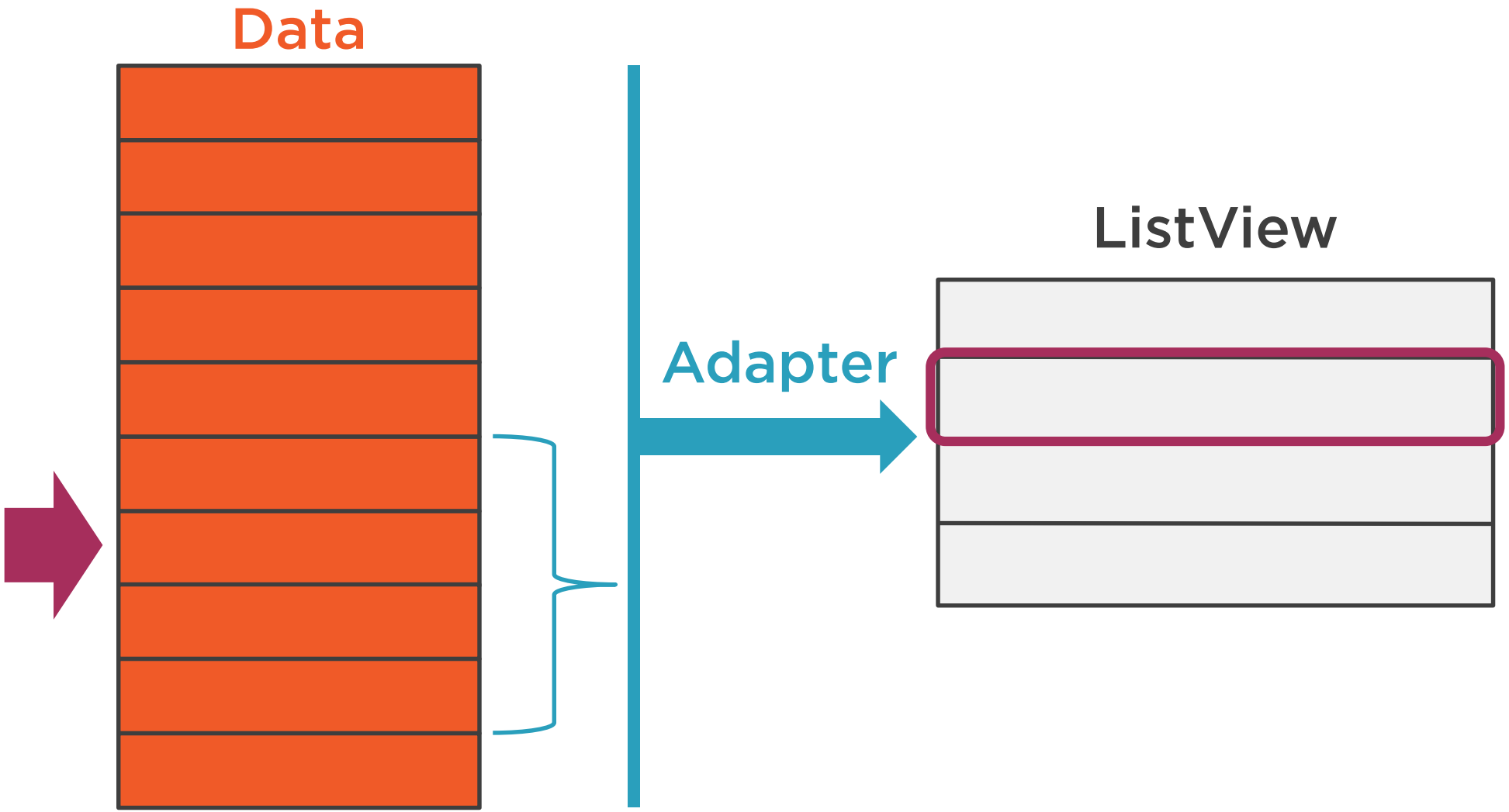
These views display multiple data items

- Only a subset may be loaded
- Test selection based on target data

Testing Views That Use Adapters



Testing Views That Use Adapters



Testing Views That Use Adapters

Espresso.onData

- Specify matcher based on target data
- Tend to use general purpose matchers

DataInteraction

- Provides methods for interacting with or narrowing match

Tend to use DataInteraction.perform

- Performs action on top-level view for entry in AdapterView

Back Button

Espresso.pushBack

- Performs action of pushing back button
- No reference to a view needed

Verifying Behavior

Tests meant to confirm expected behavior

- UI Behavior
- Logic Behavior

Verifying UI Behavior

ViewInteraction.check method

- Confirms some aspect of a view

ViewAssertions class

- Provides view assertion methods

Verifying UI Behavior

Common ViewAssertions methods

- matches
  - Confirms view matches passed matcher
  - Commonly used with ViewMatchers
  - Also confirms that view exists
- doesNotExist
  - Confirms that view does not exist

Verifying Logic Behavior

Assert methods still important

- Main way we confirm logic behavior

Summary

Instrumented tests

- Run on an emulator or device
- Have full Android environment

Types of instrumented tests

- Unit tests that rely on Android
- Automated UI tests

Summary

Instrumented tests use JUnit 4

- Test methods marked with @Test
- Support pre/post-processing methods
- Mark test class with @RunWith
  - Pass AndroidJUnit4.class

Automated UI tests

- A type of instrumented test
- Generally use ActivityTestRule
  - Manages test Activity lifetime

Summary

Espresso.onView method

- Locates view based on view criteria
- Returns a ViewInteraction reference

Espresso.onData method

- Used with AdapterViews
- Locates view based on data criteria
- Returns a DataInteraction reference

Criteria based on matchers

- Hamcrest matchers
- ViewMatchers class

Summary

Performing UI actions

- ViewInteraction.perform method
- DataInteraction.perform method

Verifying test behavior

- ViewInteraction.check method
- ViewAssertions class
- Assert class