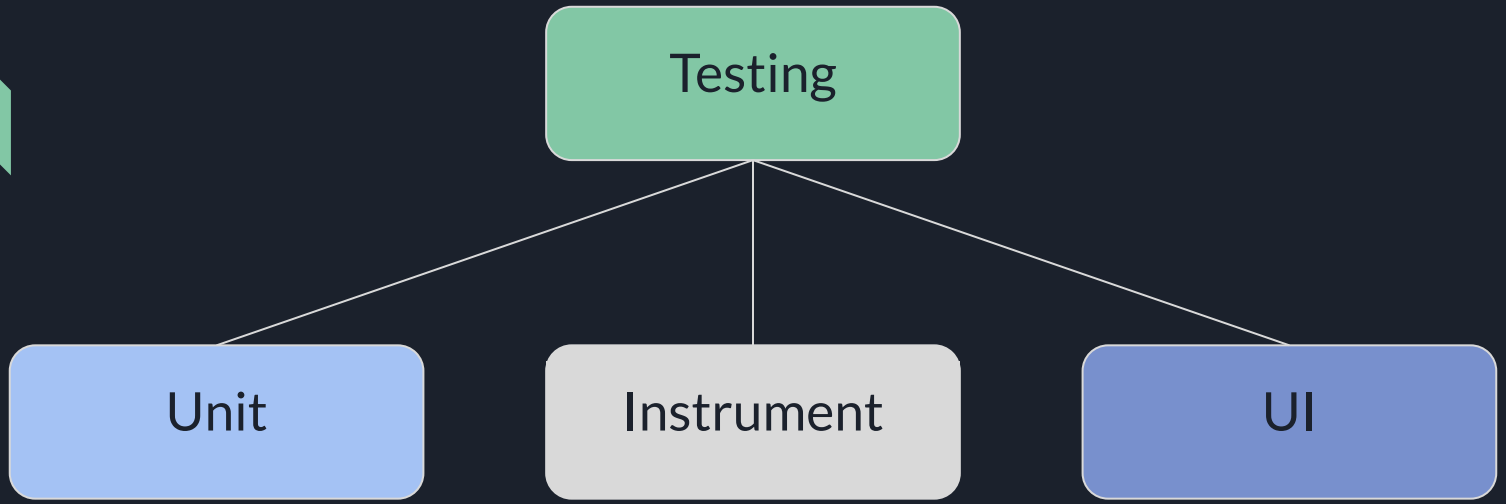
Android Testing

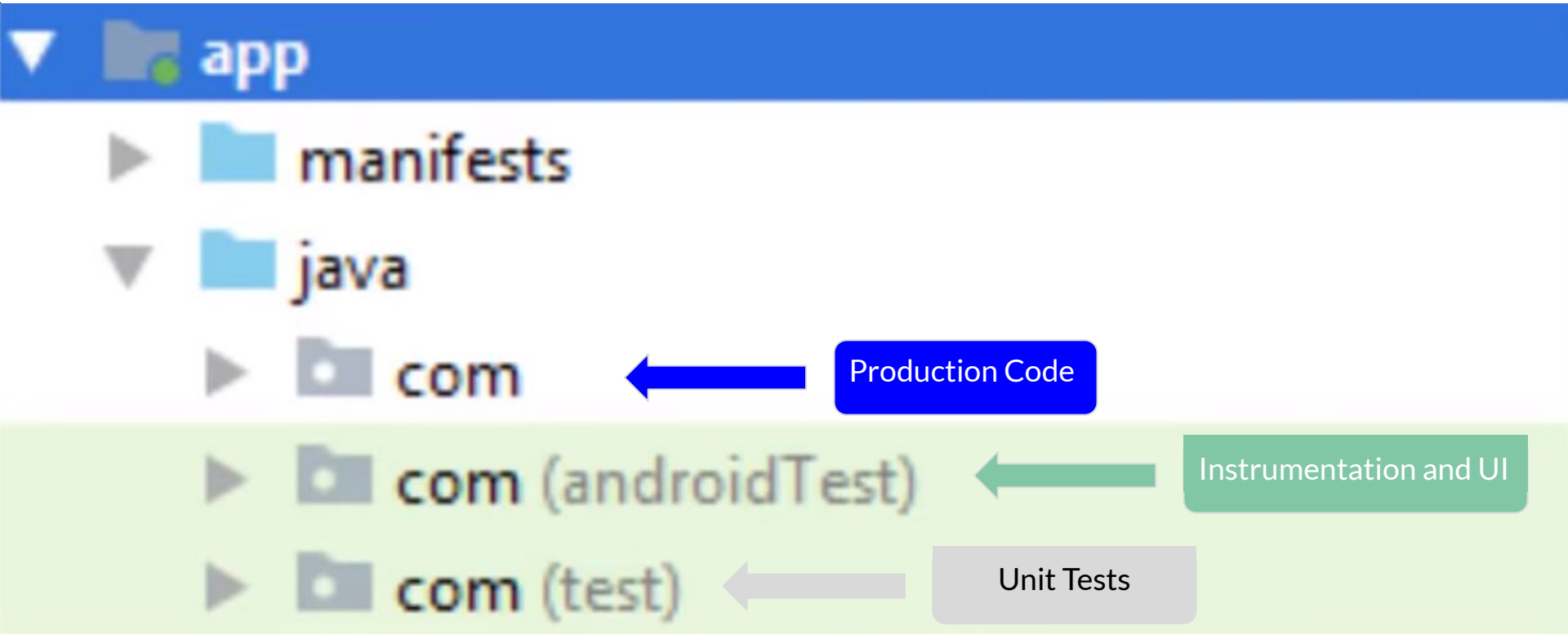
Alex Rhodes
Alexander Balsamo



- JUnit
- Mockito

- JUnit
- Mockito

- Espresso


app
manifests
java
com
Production Code
com (androidTest)
Instrumentation and UI
com (test)
Unit Tests

Unit Tests

- Local tests on personal computers
- Use internal Java Virtual Machine (JVM)
- Test logic of methods
  - Fast testing
- Libraries: JUnit, Mockito

Setup Your Environment for Unit Tests

In your apps top `build.gradle` file:

```
dependencies {
    // Required -- JUnit 4 framework
    testImplementation 'junit:junit:4.12'
    // Optional -- Robolectric environment
    testImplementation 'androidx.test:core:1.0.0'
    // Optional -- Mockito framework
    testImplementation 'org.mockito:mockito-core:1.10.19'
}
```

Basic Unit Test

- Ctrl+Shift+’t’ on a method in your MainActivity
- Chose to put unit tests in com.example…(test)

```
import com.google.common.truth.Truth.assertThat;
import org.junit.Test;

public class EmailValidatorTest {
    @Test
    public void emailValidator_CorrectEmailSimple_ReturnsTrue() {
        assertThat(EmailValidator.isValidEmail("name@email.com")).isTrue();
    }
}
```

- Can also assert is(), equalTo(), null(), notNull(), etc.

Instrument Tests

- Tests android specific functionalities
  - Activities, fragments, context, lifecycle, etc.
- Tested on device or emulator
- Libraries: JUnit, Mockito

Setup your Environment for Instrumental Tests

In your apps top `build.gradle` file:

```
dependencies {
    androidTestImplementation 'androidx.test:runner:1.1.0'
    androidTestImplementation 'androidx.test:rules:1.1.0'
    // Optional -- Hamcrest library
    androidTestImplementation 'org.hamcrest:hamcrest-library:1.3'
    // Optional -- UI testing with Espresso
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.0'
    // Optional -- UI testing with UI Automator
    androidTestImplementation 'androidx.test.uiautomator:uiautomator:2.2.0'
}
```

To use JUnit 4 test classes, specify:

```
android {
    defaultConfig {
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
}
```

Example Instrumented Test

```
// @RunWith is required only if you use a mix of JUnit3 and JUnit4.
@RunWith(AndroidJUnit4.class)
@SmallTest
public class TempConverterTest {
        EditText et = (EditText)getActivity().findViewById(R.id.editText);
        float userInput = Float.valueOf(et.getText().toString());
        float myInput = 100;
        float output;
        float expected = 212;

        TempConverterClass converter = new TempConverterClass();
        output = converter.convertCelciustoFahrenheit(userInput);

        // Verify that the received data is correct.
        assertEqual(myInput, userInput);
        assertEqual(output, expected);
    }
}
```

UI Tests

- Simulates user interaction
  - onClicks, widgets, editTexts, etc.
- Can test faster than normal user inputs could
- Need android framework → real device or emulator
- Libraries: Espresso

Automated UI Tests with Espresso

- Run in an emulator or real device
- Human tester can perform operations on the target app
- Creates tests without writing any test code!
- Record a test scenario and add assertions to verify UI elements
  - Run → Record Espresso Test
  - Select Deployment Target (choose device)
  - Record your test window appears
- Assertions:
  - Text is, exists, does not exist, etc.
- Export test into human readable code!
  - This test code can actually be edited afterwards in the com.example… (AndroidTest) folder

Tips!

- Build your code modularly for more effective testing
- Consider edge cases
  - JUnit asserts for method logic
- Use Espresso for expected (and unexpected) user actions
- Use Mockito/Espresso to assert objects are (in)visible/(un)clickable
- Don’t test every value/input at infinitum
  - Focus on base cases/branching factors

Literature Cited

Tabian, Mitch. “CodingWithMitch.com.”

https://developer.android.com/training/testing/unit-testing/local-unit-tests

https://developer.android.com/studio/test/espresso-test-recorder

https://developer.android.com/training/testing/ui-testing