Android Developer Fundamentals V2

Testing, debugging, and using support libraries

Lesson 3

3.2 App testing

Contents

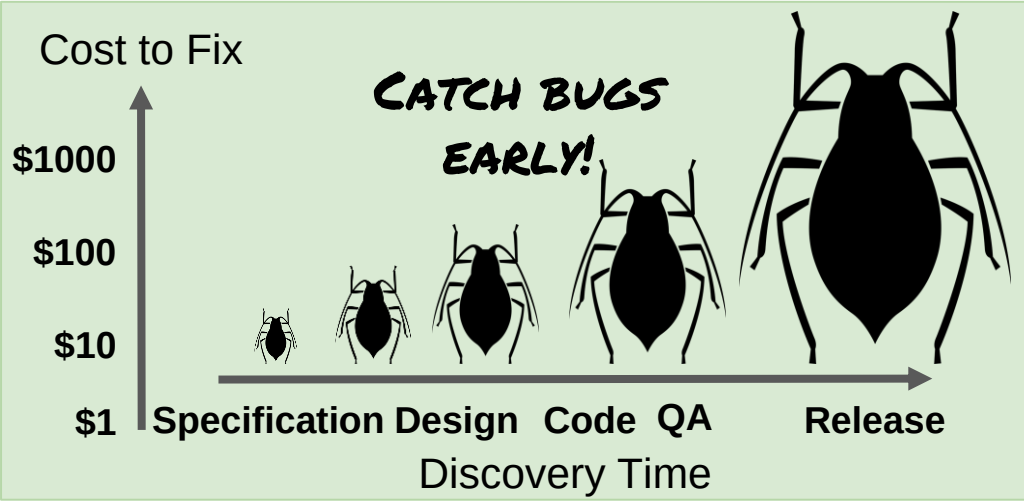
- Why testing is worth your time
- Unit testing

Note: User interface testing (instrumented testing) is covered in another chapter

Testing rocks

Why should you test your app?

- Find and fix issues early
- Less costly
- Takes less effort
- Costs to fix bugs increases with time



Types of testing

- Levels of Testing
  - Component, integration, protocol, system
- Types of Testing
  - Installation, compatibility, regression, acceptance
  - Performance, scalability, usability, security
- User interface and interaction tests
  - Automated UI testing tools
  - Instrumented testing (covered in another chapter)

Test-Driven Development (TDD)

1. Define a test case for a requirement
2. Write tests that assert all conditions of the test case
3. Write code against the test
4. Iterate on and refactor code until it passes the test
5. Repeat until all requirements have test cases, all tests pass, and all functionality has been implemented

Tests in your project

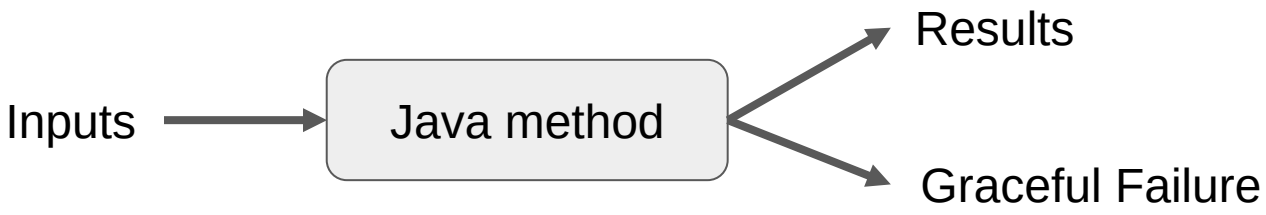
Android Studio creates three source sets for your project

- **main**—code and resources
- **(test)**—local unit tests
- **(androidTest)**—instrumented tests

Local Unit Tests

Unit tests

- Smallest testable parts of your program
- Isolate each component and demonstrate the individual parts are correct
- Java Method tests



Local unit tests in JUnit

- Compiled and run entirely on your local machine with the Java Virtual Machine (JVM)
- Use to test the parts of your app (such as the internal logic):
  - If you don't need access to Android framework or device/emulator
  - If you can create fake (mock) objects that pretend to behave like the framework equivalents
- Unit tests are written with JUnit, a common unit testing framework for Java.

Local unit tests in your project

- Tests are in the same package as the associated application class
- Only org.junit imported — no Android classes
- Project path for test classes: .../module-name/src/**test**/java/

Imports for JUnit

```
// Annotations
import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;

// Basic JUnit4 test runner
import org.junit.runners.JUnit4;

// assertThat method
import static org.junit.Assert.assertThat;
```

Testing class

```
/**
* JUnit4 unit tests for the calculator logic.
* These are local unit tests; no device needed
*/
@RunWith(JUnit4.class) // Specify the test runner
public class CalculatorTest { // Name it what you are testing
}
```

ExampleTest

```
/
**
* Test for simple addition.
* Each test is identified by a @Test annotation.
*/
@Test
public void addTwoNumbers() {
   double resultAdd = mCalculator.add(1d, 1d);
   assertThat(resultAdd, is(equalTo(2d)));
}
```

@Test Annotation

- Tells JUnit this method is a test method (JUnit 4)
- Information to the test runner
- Not necessary anymore to prefix test methods with "test"

setUp() method

```
/**
* Set up the environment for testing
*/
@Before
public void setUp() {
    mCalculator = new Calculator();
}
```

- Sets up environment for testing
- Initialize variables and objects used in multiple tests

tearDown() method

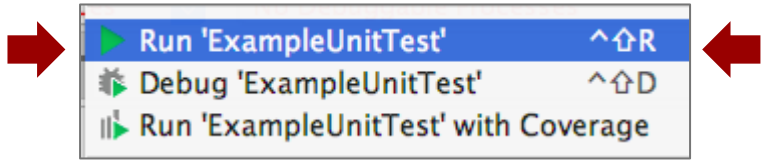
```
/**
* Release external resources
*/
@After
public void tearDown() {
    ....
}
```

- Frees resources

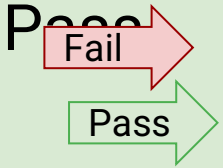
Running tests in Android Studio

Starting a test run

- **Right-click** test class and select **Run 'app_name' test**
- **Right-click** test package and select **Run tests in 'package'**



Passing and failing



Fail

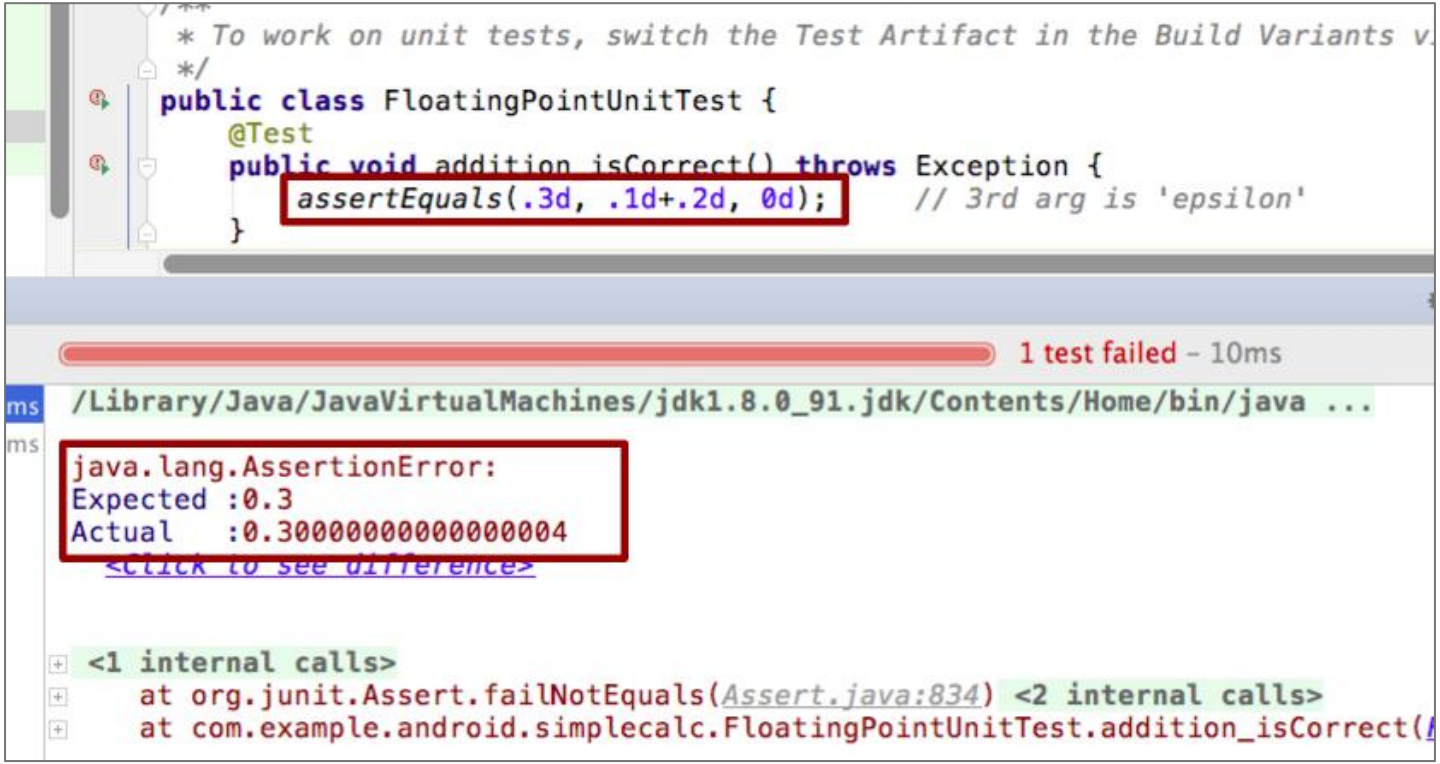
Result details

Testing floating point results

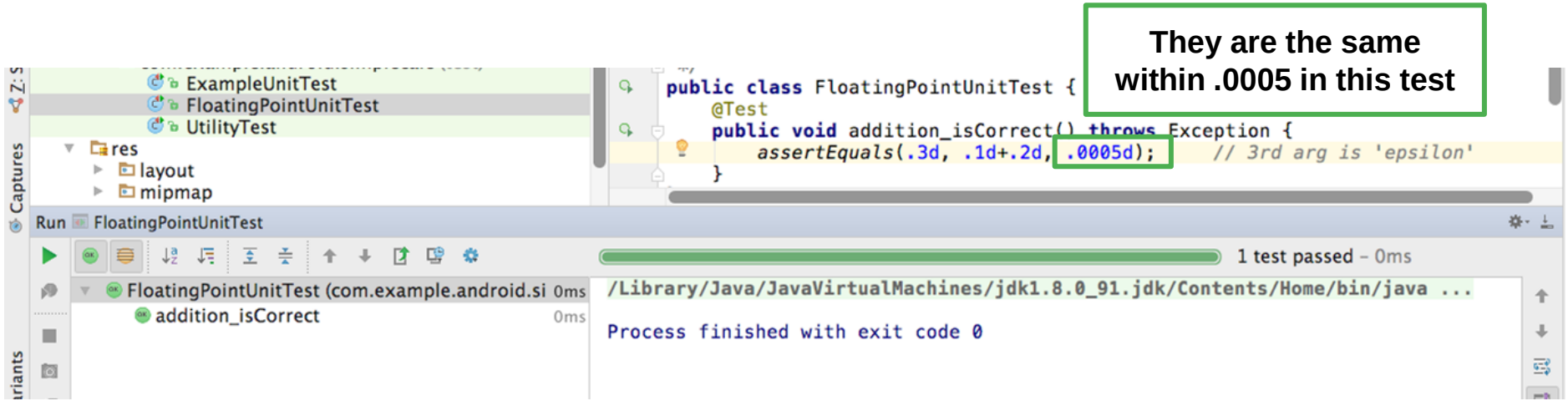
Testing floating point

- Be careful with floating point tests
- Recall from basic computer science:
  - Floating point arithmetic is not accurate in binary

Test fails with floating point numbers



Fix test with floating point numbers



Learn more

- Getting Started with Testing
- Best Practices for Testing
- Building Local Unit Tests
- JUnit 4 Home Page
- JUnit 4 API Reference
- Android Testing Codelab
- Android Tools Protip: Test Size Annotations
- Android Testing Support - Testing Patterns (video)

What's Next?

- Concept Chapter: 3.2 App testing
- Practical: 3.2 Unit tests

END