Implementing Local Unit Tests

Jim Wilson
MOBILE SOLUTIONS DEVELOPER & ARCHITECT

@hedgehogjim blog.jwhh.com

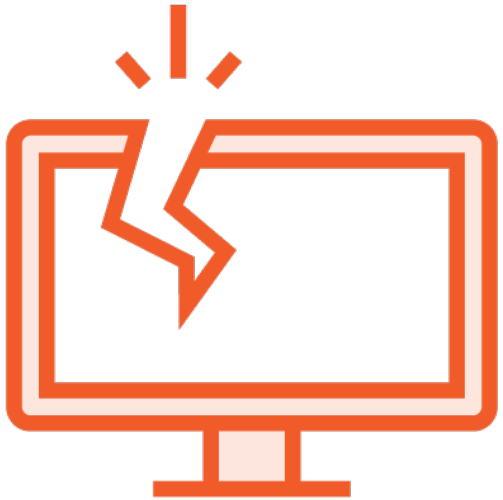
What to Expect from This Module

- Testing basics
- Efficiently running unit tests
- Creating unit tests
- Assert class
- Assuring test consistency

Testing

Testing needs to be a core task

Essential to delivering quality software

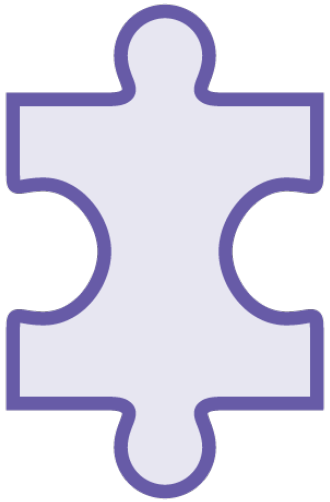


Functional testing

Verify behaves as expected

Detect breaking changes

Testing

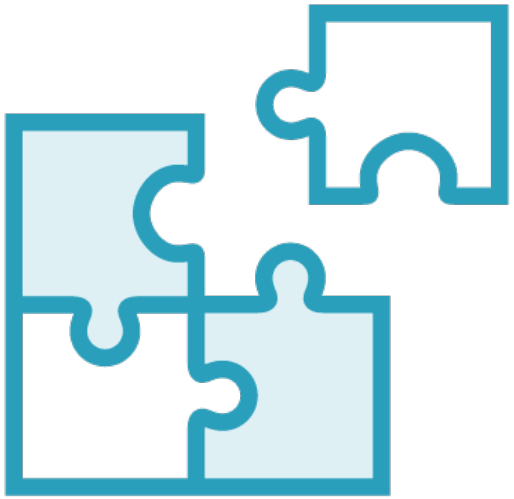


Unit testing

Testing of units of code

Tests specific feature/behavior

Generally will have many unit tests



Integration testing

Testing pieces of being put together

Application behaviors

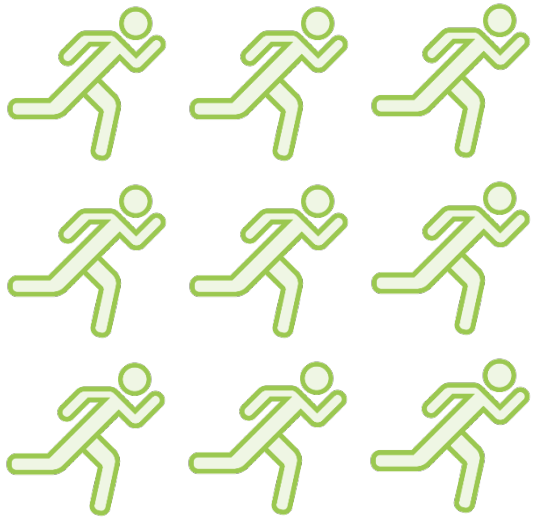
Often involve testing of UI

Unit Testing

Unit tests should be run often

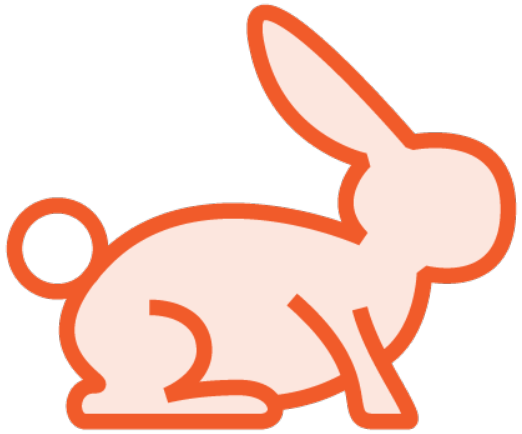
After code changes

Before check-in to main source branch



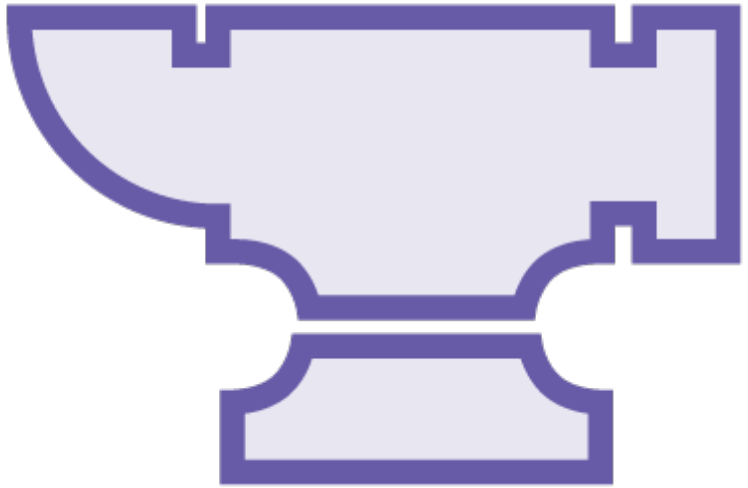
Generally want to run all unit tests

No change is complete until all tests pass



Ideally can be run reasonably quickly

Android App Testing



Challenges of testing Android apps

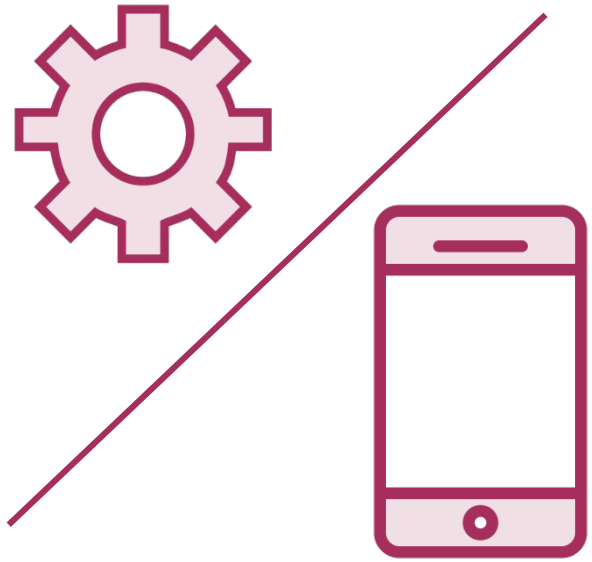
Full testing needs Android environment

Requires emulator or physical device

Need way to efficiently run unit tests

Limit how often full Android environment is needed

Efficiently Running Unit Tests



Android apps

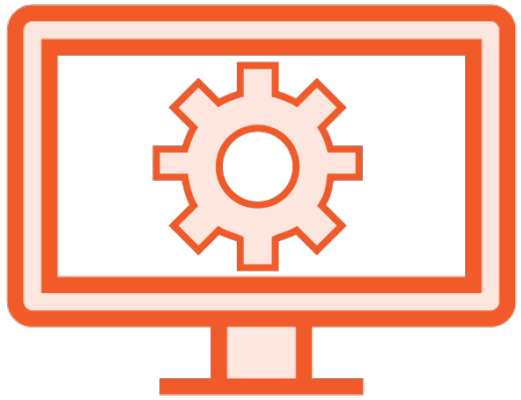
Logic-based behavior

Android-based behavior

Separate tests

Test logic-based behavior

Test Android-based behavior

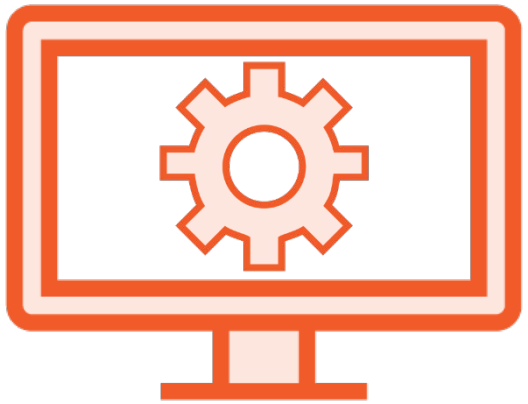


Efficiently run unit tests

Test logic-based behavior locally

Leverage JVM on desktop

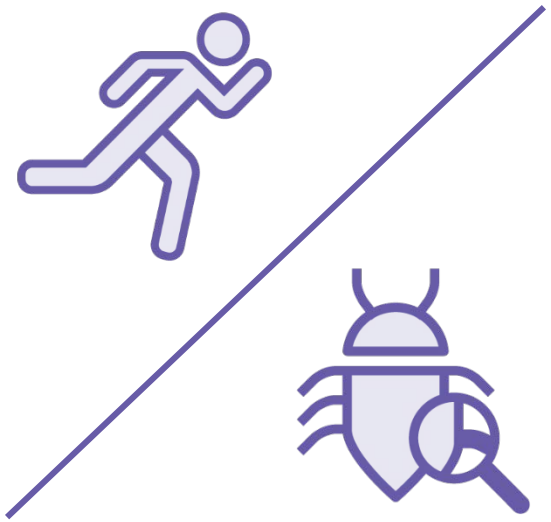
Local JVM Tests



Android JVM testing

Separate source set

Uses JUnit



Can run or debug tests

Single test

Group of tests

All tests

Display test results

Success/failure
indicated by color

Testing with JUnit

Each test is a separate method

Marked with @Test annotation

JUnit handles execution details

Tests grouped within classes

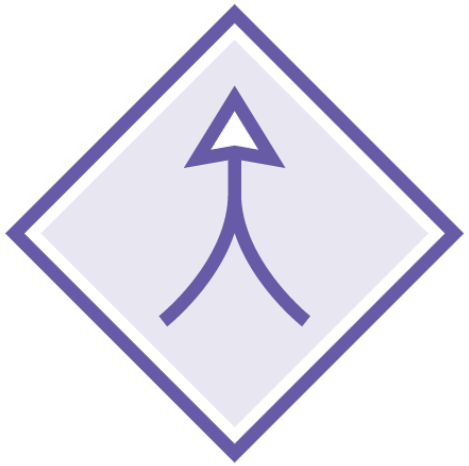
Primarily for organization convenience

Allows execution grouping

Assert class

- Use to indicate expected results
- Fails test when expectation not met

Assert Class



assertSame

2 references to same object

assertEquals

2 objects equal (equals method)

assertNull

Reference is null

Negative versions of most methods

assertNotSame, assertNotEqual, etc.

Assuring Test Consistency

Test reliability

Must run consistently

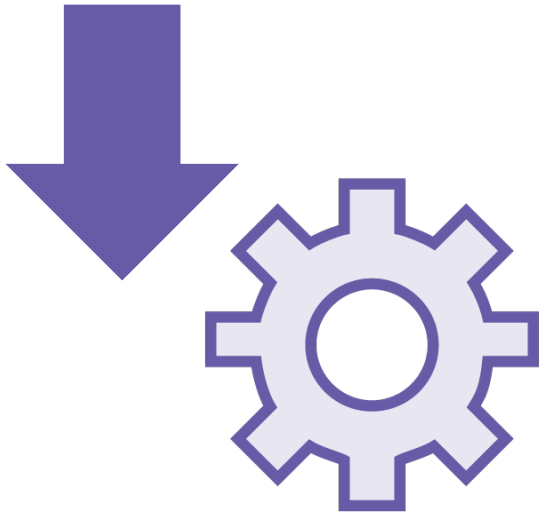
Can't depend on action of other tests

Must be safe from side effects of other tests

Must have consistent start state

Need way to set/reset test state

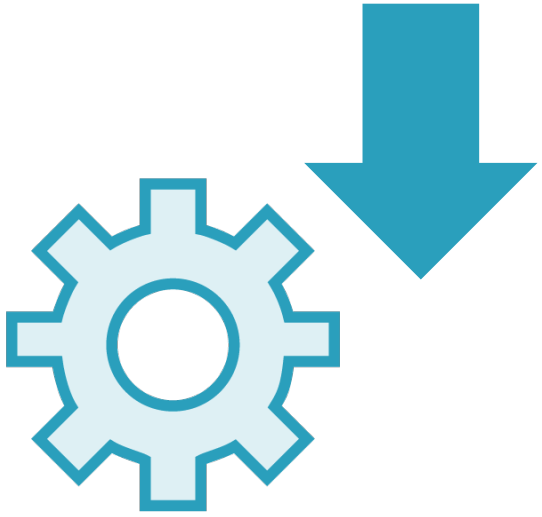
Assuring Test Consistency



Test pre-processing

Runs before each test

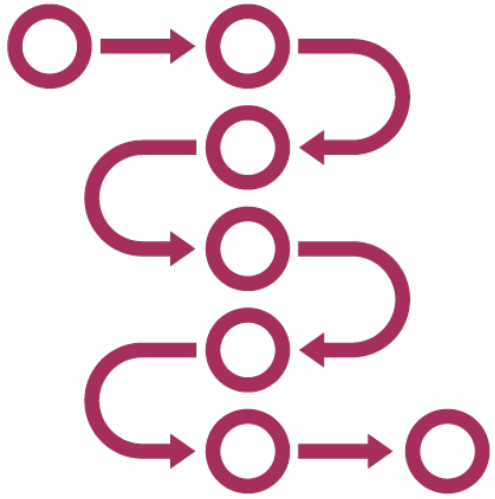
Method with @Before annotation



Test post-processing

Runs after each test

Method with @After annotation



Multiple pre/post methods

All will run

Order not guaranteed

Summary

Testing needs to be a core task

- Essential to delivering quality software

Unit Testing

- Units of relatively simple tests
- Focus is on specific feature/behavior
- Tests should be run frequently

Summary

Local JVM testing

- Focused on logic-based behavior
- Run directly on desktop

Managing tests in Android Studio

- Can run or debug tests
- Displays test success or failure

Summary

Test methods

- Grouped in classes
- Marked with @Test annotation

Assert class

- Use to indicate expected results
- Fails test when expectation not met

Summary

Tests need to be reliable

- Need to assure consistent testing

Can run test pre/post-processing

- Use @Before and @After
- Methods run for each @Test method