Unit Testing in Android with GitHub Action

Teaching Assistants

Boston University

Apr. 15th, 2025

Outline

Previously: Google Test for C++

Today: Unit Testing in Android

Unit Testing Constraints

Test Suits with GitHub Actions

Code Merging after Passing Unit Testing

Weather Demo Unit Testing

Takeaways

Part 1 — Previously: Google Test (gtest) for C++

Keep on a straight path with proper unit testing
The more code you write without testing, the more paths you have to check for errors

Ensuring that 'the thing' does what it is supposed to do

Credits: Introduction to Google Test: An Open Source C/C++ Unit-Testing Framework

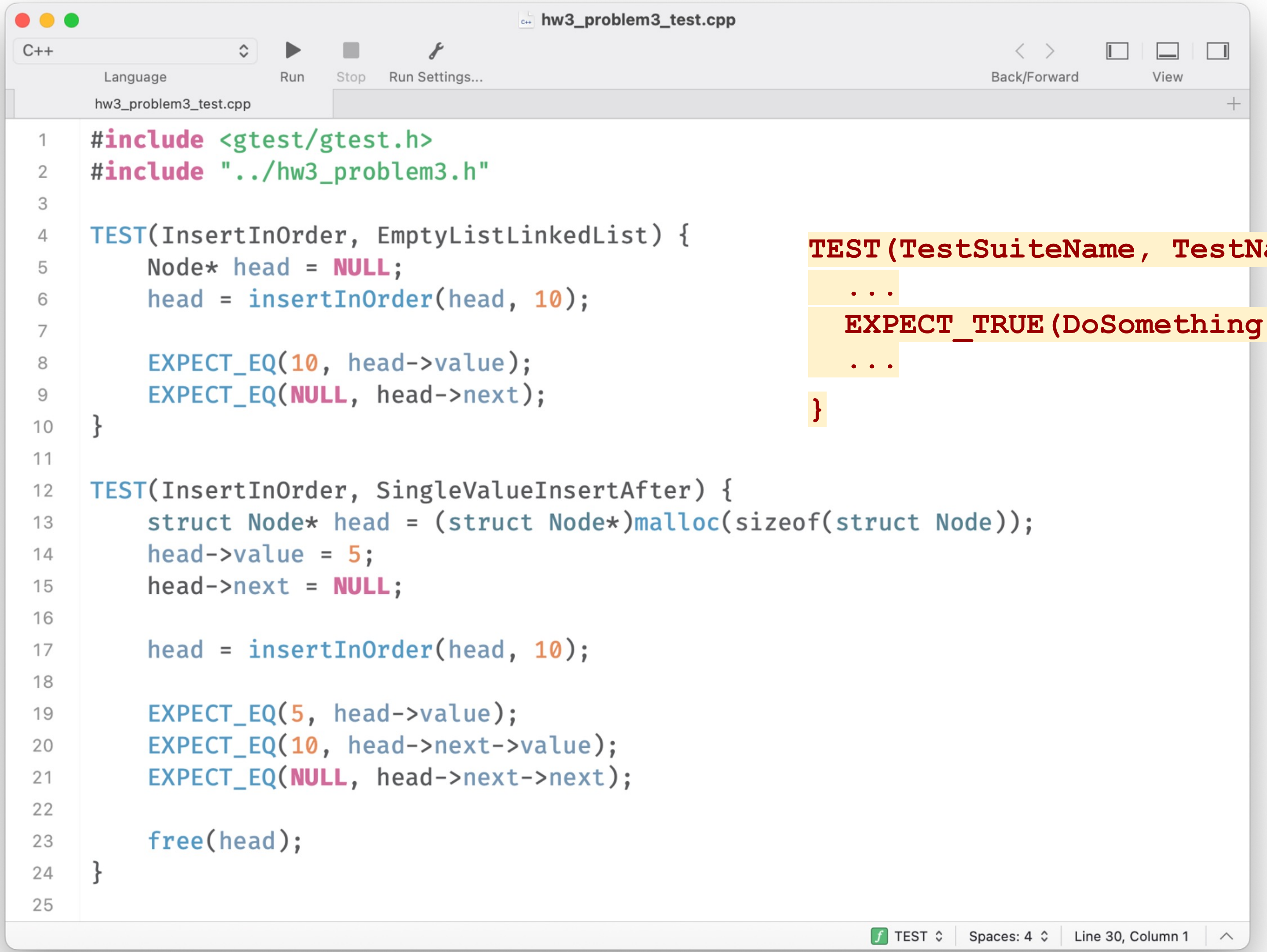
Part 1 — Previously: Google Test (gtest) for C++

An Open Source C/C++ Unit-Testing Framework

- **Ensuring that 'the thing' does what it is supposed to do**
- Tests should be independent and repeatable.
- Tests should be well organized and reflect the structure of the tested code.
- Tests should be portable and reusable.
- Some example codes: Android-Unit-Testing/tree/main/GoogleTest/tests

Credits: Introduction to Google Test: An Open Source C/C++ Unit-Testing Framework

Part 1 — Previously: Google Test (gtest) for C++

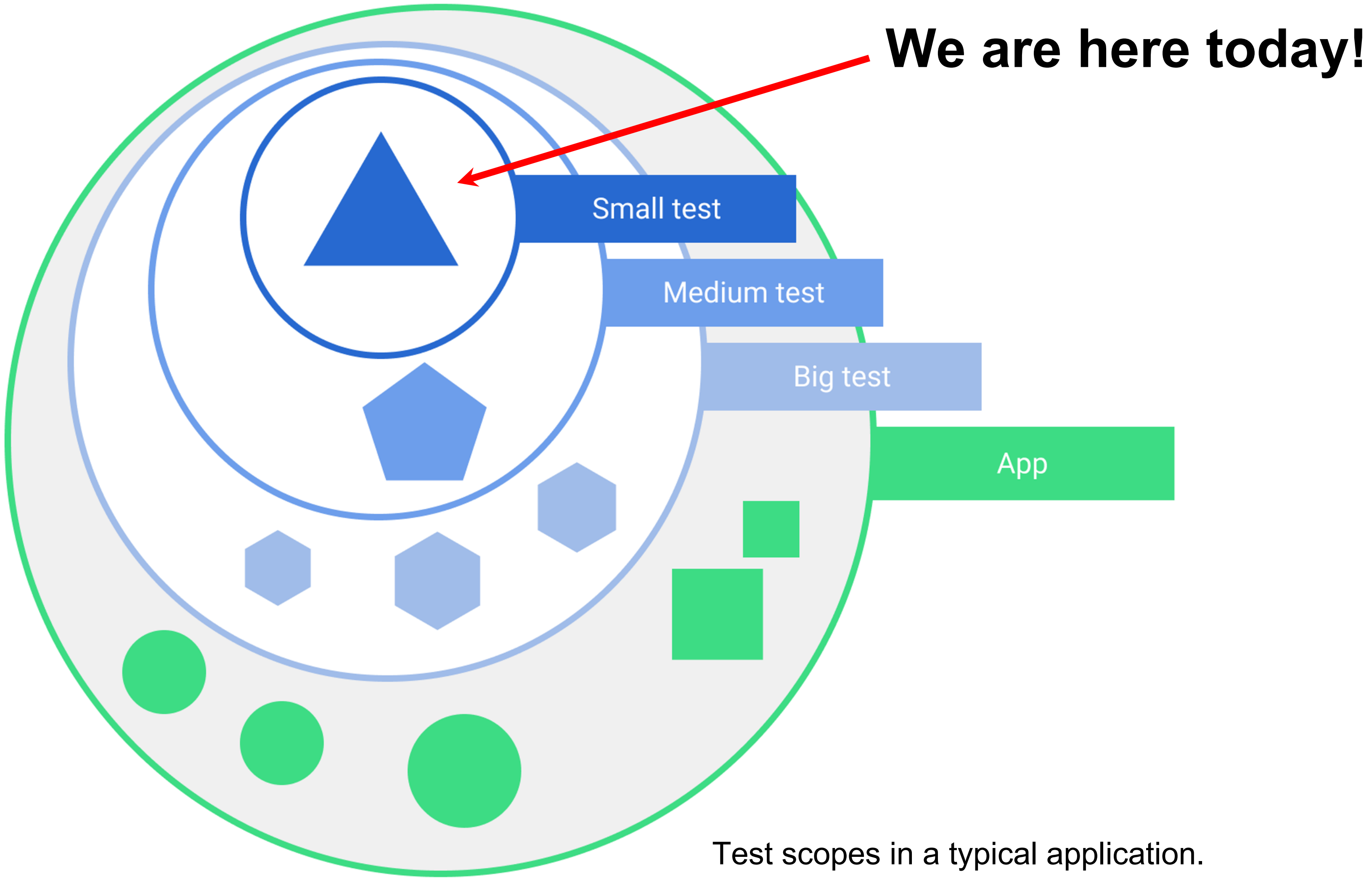


```cpp
#include <gtest/gtest.h>
#include "../hw3_problem3.h"

TEST(InsertInOrder, EmptyListLinkedList) {
    Node* head = NULL;
    head = insertInOrder(head, 10);

    EXPECT_EQ(10, head->value);
    EXPECT_EQ(NULL, head->next);
}

TEST(InsertInOrder, SingleValueInsertAfter) {
    struct Node* head = (struct Node*)malloc(sizeof(struct Node));
    head->value = 5;
    head->next = NULL;

    head = insertInOrder(head, 10);

    EXPECT_EQ(5, head->value);
    EXPECT_EQ(10, head->next->value);
    EXPECT_EQ(NULL, head->next->next);

    free(head);
}
```

```cpp
TEST(TestSuiteName, TestName) {
  ...
  EXPECT_TRUE(DoSomething(GetParam()));
  ...
}
```

Credits: https://google.github.io/googletest/primer.html

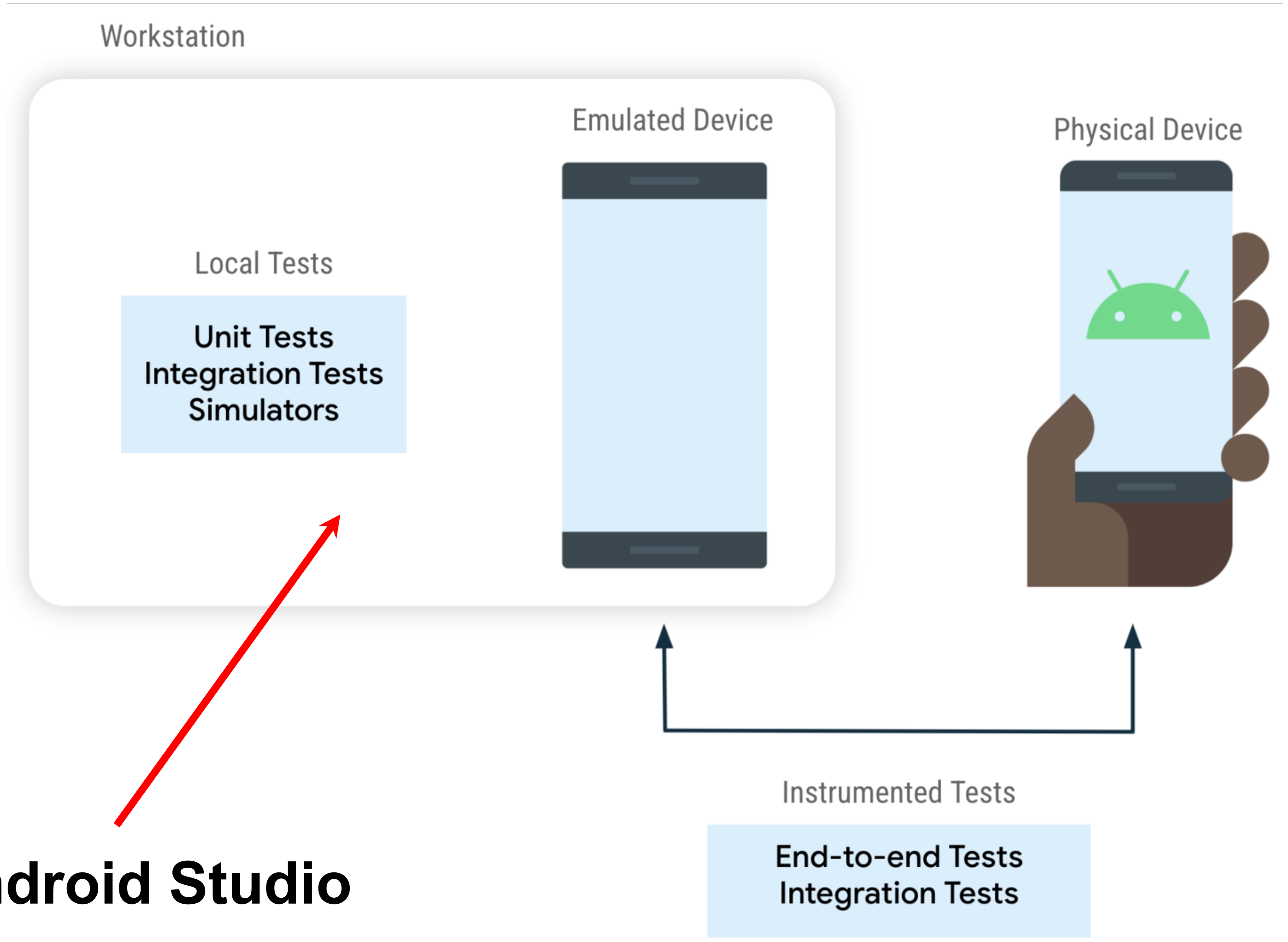
Part 2 — Today: Unit Testing in Android



Test scopes in a typical application.

Credits: [Fundamentals of testing Android apps]

Part 2 — Today: Unit Testing in Android



Android Studio

Credits: [Fundamentals of testing Android apps]()

Different types of tests depending on where they run.

Part 2 — Today: Unit Testing in Android

A type of **software testing** where individual units/components of a software are tested

Done during the **development of an application**

The **objective** of Unit Testing is to verify its correctness

Usually performed by the developer

Tools that we use in Android: **JUnit**

Part 2 — Today: Unit Testing in Android

JUnit

Tools that we use in Android: **JUnit4**

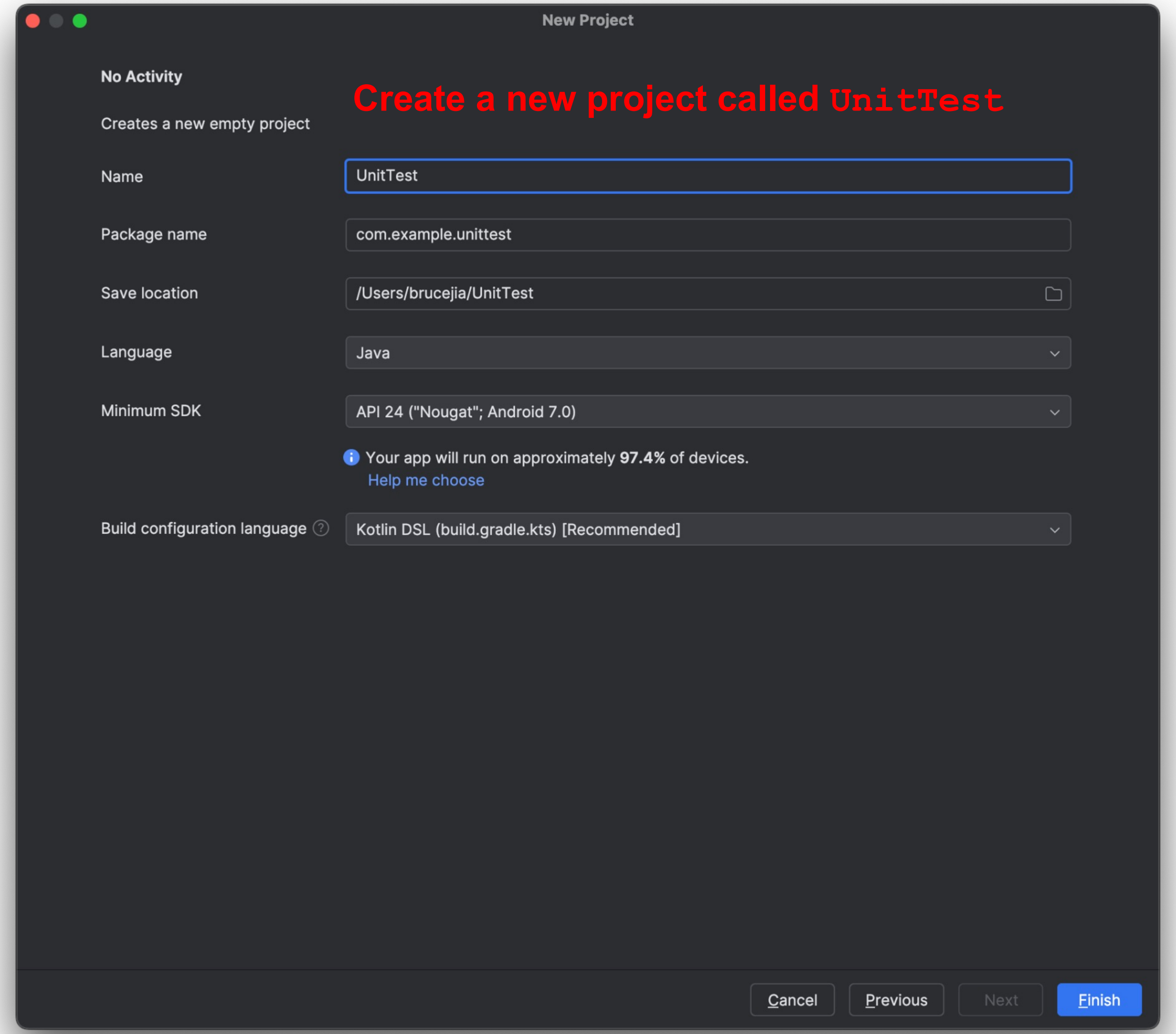
A programmer-oriented testing framework for Java.

By default, we will use [Gradle Build Tool]

Gradle Build Tool accelerates developer productivity

Credits: https://github.com/junit-team/junit4/wiki/Use-with-Gradle

Part 2 — Today: Unit Testing in Android


New Project

No Activity

Create a new project called `UnitTest`

Creates a new empty project

Name: UnitTest

Package name: com.example.unittest

Save location: /Users/brucejia/UnitTest

Language: Java

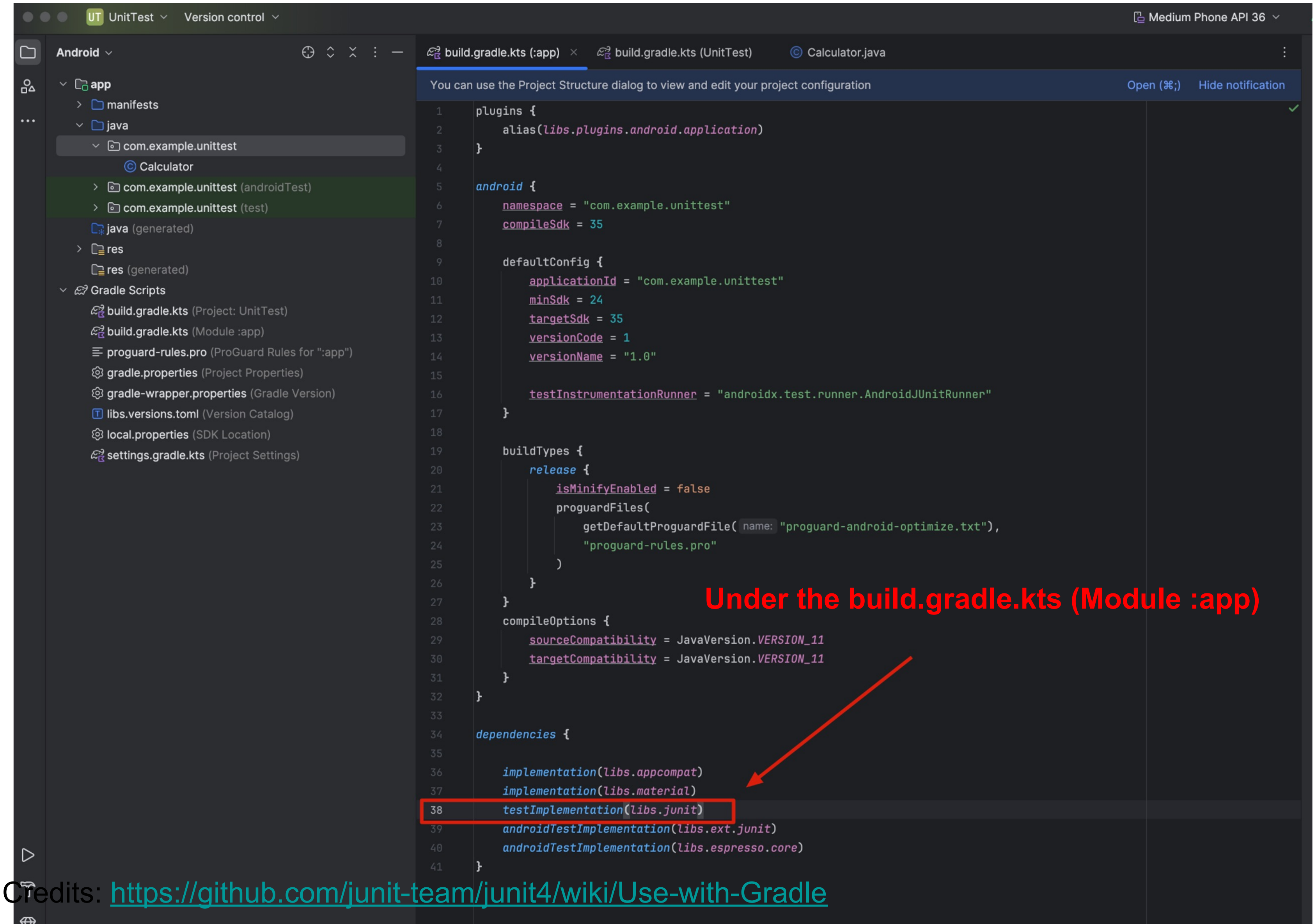
Minimum SDK: API 24 ("Nougat"; Android 7.0)

Your app will run on approximately **97.4%** of devices.
Help me choose

Build configuration language: Kotlin DSL (build.gradle.kts) [Recommended]

Credits: https://github.com/junit-team/junit4/wiki/Use-with-Gradle

Part 2 — Today: Unit Testing in Android


UnitTest
Version control
Medium Phone API 36
Android
build.gradle.kts (:app)
build.gradle.kts (UnitTest)
Calculator.java
app
manifests
java
com.example.unittest
Calculator
com.example.unittest (androidTest)
com.example.unittest (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: UnitTest)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
Open (⌘;)
Hide notification

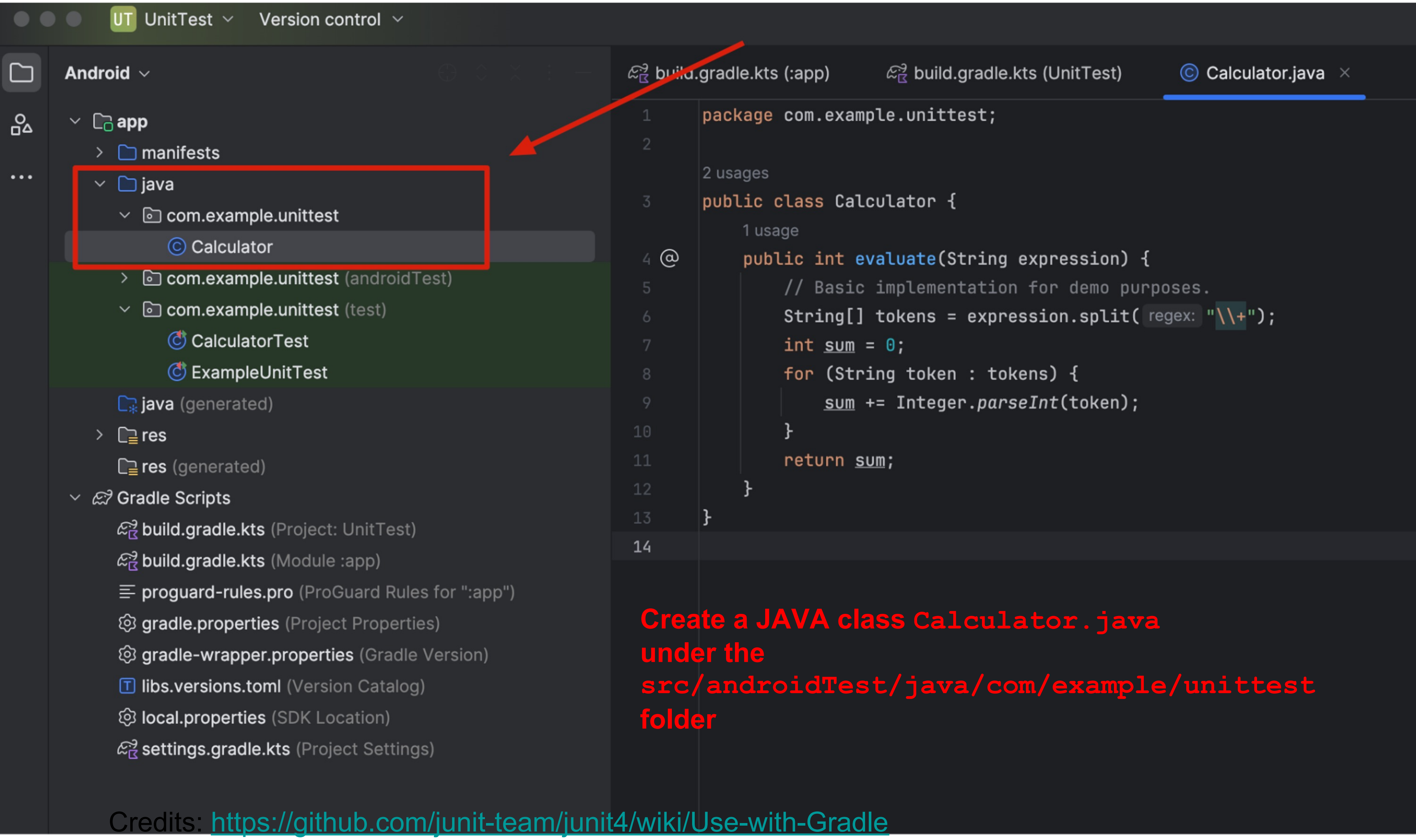
You can use the Project Structure dialog to view and edit your project configuration

```
plugins {
    alias(libs.plugins.android.application)
}

android {
    namespace = "com.example.unittest"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.example.unittest"
        minSdk = 24
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile( name: "proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
}

dependencies {

    implementation(libs.appcompat)
    implementation(libs.material)
    testImplementation(libs.junit)
    androidTestImplementation(libs.ext.junit)
    androidTestImplementation(libs.espresso.core)
}
```

Under the build.gradle.kts (Module :app)

Credits: https://github.com/junit-team/junit4/wiki/Use-with-Gradle

Part 2 — Today: Unit Testing in Android


UT UnitTest
Version control
Android
app
manifests
java
com.example.unittest
Calculator
com.example.unittest (androidTest)
com.example.unittest (test)
CalculatorTest
ExampleUnitTest
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: UnitTest)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
build.gradle.kts (:app)
build.gradle.kts (UnitTest)
Calculator.java

```java
package com.example.unittest;

2 usages
public class Calculator {
    1 usage
    public int evaluate(String expression) {
        // Basic implementation for demo purposes.
        String[] tokens = expression.split( regex: "\\+");
        int sum = 0;
        for (String token : tokens) {
            sum += Integer.parseInt(token);
        }
        return sum;
    }
}
```

Create a JAVA class `Calculator.java` under the `src/androidTest/java/com/example/unittest` folder

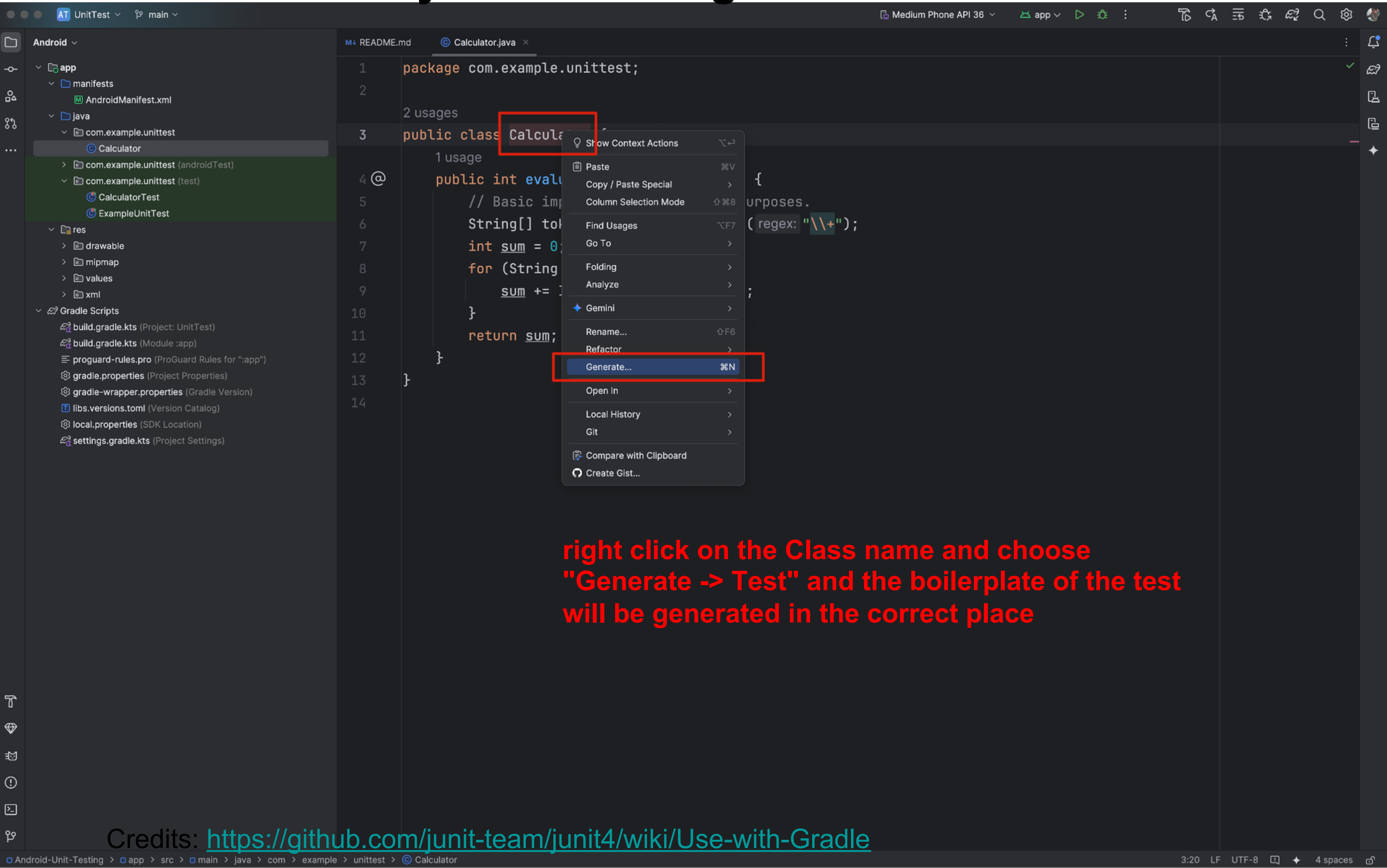
Credits: https://github.com/junit-team/junit4/wiki/Use-with-Gradle

Part 2 — Today: Unit Testing in Android

```java
package com.example.unittest;

public class Calculator {
    public int evaluate(String expression) {
        // Basic implementation for demo purposes.
        String[] tokens = expression.split("\\+");
        int sum = 0;
        for (String token : tokens) {
            sum += Integer.parseInt(token);
        }
        return sum;
    }
}
```

Credits: https://github.com/junit-team/junit4/wiki/Use-with-Gradle

Part 2 — Today: Unit Testing in Android



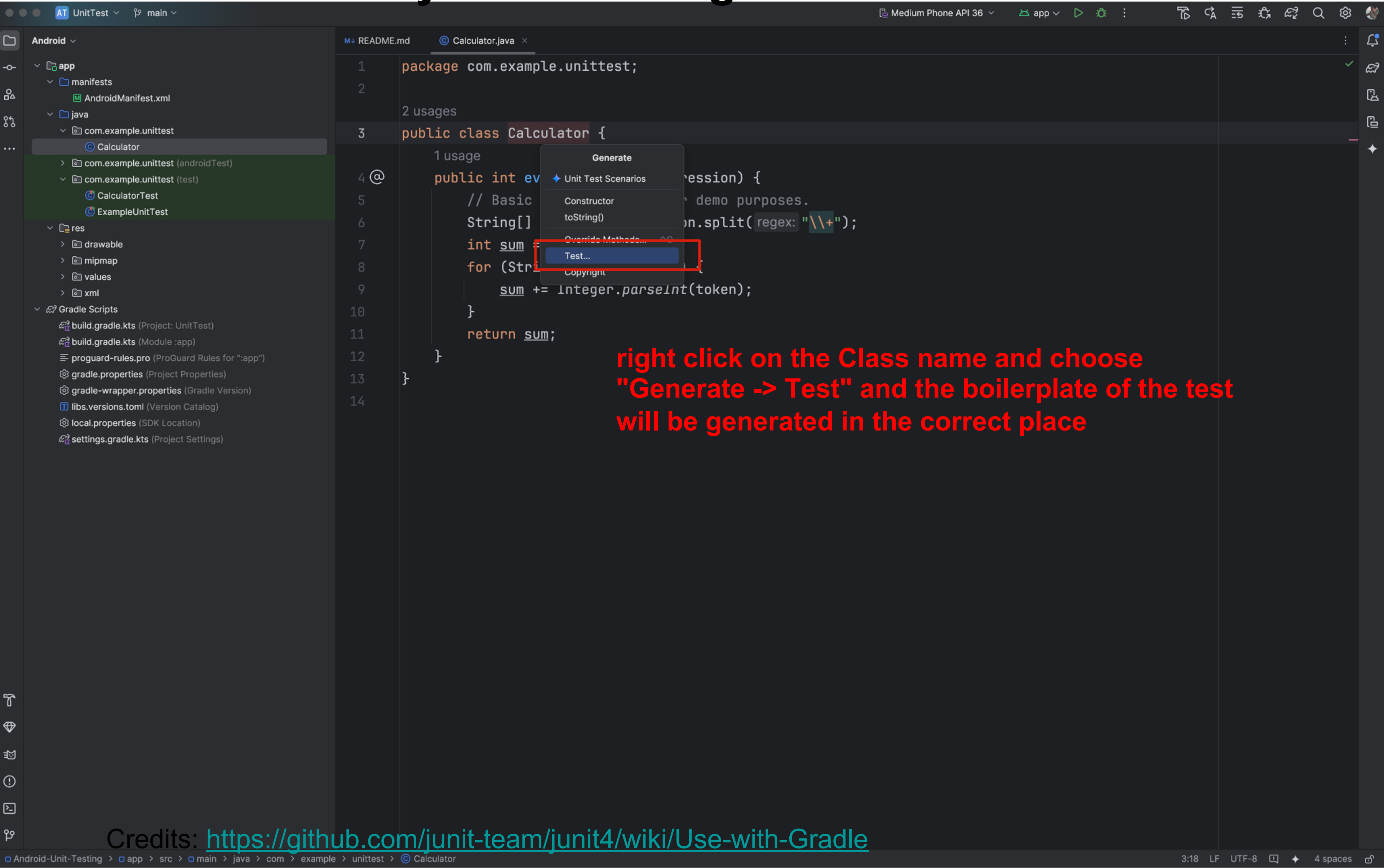
```
package com.example.unittest;

2 usages
public class Calcula
    1 usage
    public int evalu                    {
        // Basic imp                    urposes.
        String[] tok                    ( regex: "\\+");
        int sum = 0;
        for (String
            sum += I                    ;
        }
        return sum;
    }
}
```

UnitTest
main
Medium Phone API 36
app
Android
README.md
Calculator.java
app
manifests
AndroidManifest.xml
java
com.example.unittest
Calculator
com.example.unittest (androidTest)
com.example.unittest (test)
CalculatorTest
ExampleUnitTest
res
drawable
mipmap
values
xml
Gradle Scripts
build.gradle.kts (Project: UnitTest)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
Show Context Actions
Paste ⌘V
Copy / Paste Special
Column Selection Mode ⇧⌘8
Find Usages ⌥F7
Go To
Folding
Analyze
Gemini
Rename... ⇧F6
Refactor
Generate... ⌘N
Open In
Local History
Git
Compare with Clipboard
Create Gist...
Android-Unit-Testing › app › src › main › java › com › example › unittest › Calculator
3:20 LF UTF-8 4 spaces

right click on the Class name and choose "Generate -> Test" and the boilerplate of the test will be generated in the correct place

Credits: https://github.com/junit-team/junit4/wiki/Use-with-Gradle

Part 2 — Today: Unit Testing in Android

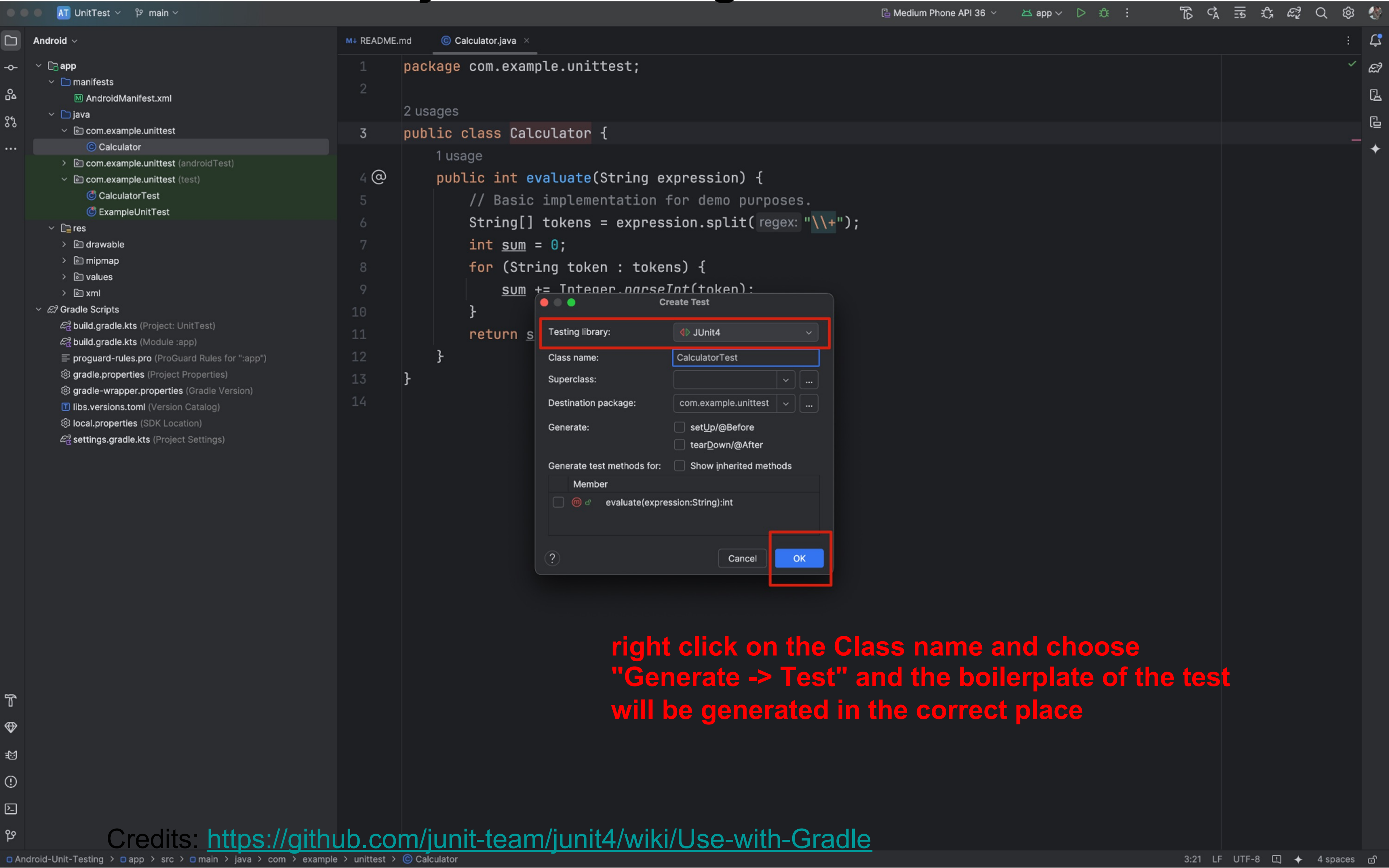

UnitTest
main
Medium Phone API 36
app
Android
app
manifests
AndroidManifest.xml
java
com.example.unittest
Calculator
com.example.unittest (androidTest)
com.example.unittest (test)
CalculatorTest
ExampleUnitTest
res
drawable
mipmap
values
xml
Gradle Scripts
build.gradle.kts (Project: UnitTest)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
README.md
Calculator.java
Generate
Unit Test Scenarios
Constructor
toString()
Override Methods...
Test...
Copyright
Android-Unit-Testing > app > src > main > java > com > example > unittest > Calculator
3:18 LF UTF-8 4 spaces

```
package com.example.unittest;

2 usages
public class Calculator {
    1 usage
    public int ev                              ession) {
        // Basic                             demo purposes.
        String[]                         n.split( regex: "\\+");
        int sum =
        for (Str                              ) {
            sum += Integer.parseInt(token);
        }
        return sum;
    }
}
```

right click on the Class name and choose "Generate -> Test" and the boilerplate of the test will be generated in the correct place

Credits: https://github.com/junit-team/junit4/wiki/Use-with-Gradle

Part 2 — Today: Unit Testing in Android



```
package com.example.unittest;

2 usages
public class Calculator {
    1 usage
    public int evaluate(String expression) {
        // Basic implementation for demo purposes.
        String[] tokens = expression.split( regex: "\\+");
        int sum = 0;
        for (String token : tokens) {
            sum += Integer.parseInt(token);
        }
        return s
    }
}
```

Create Test

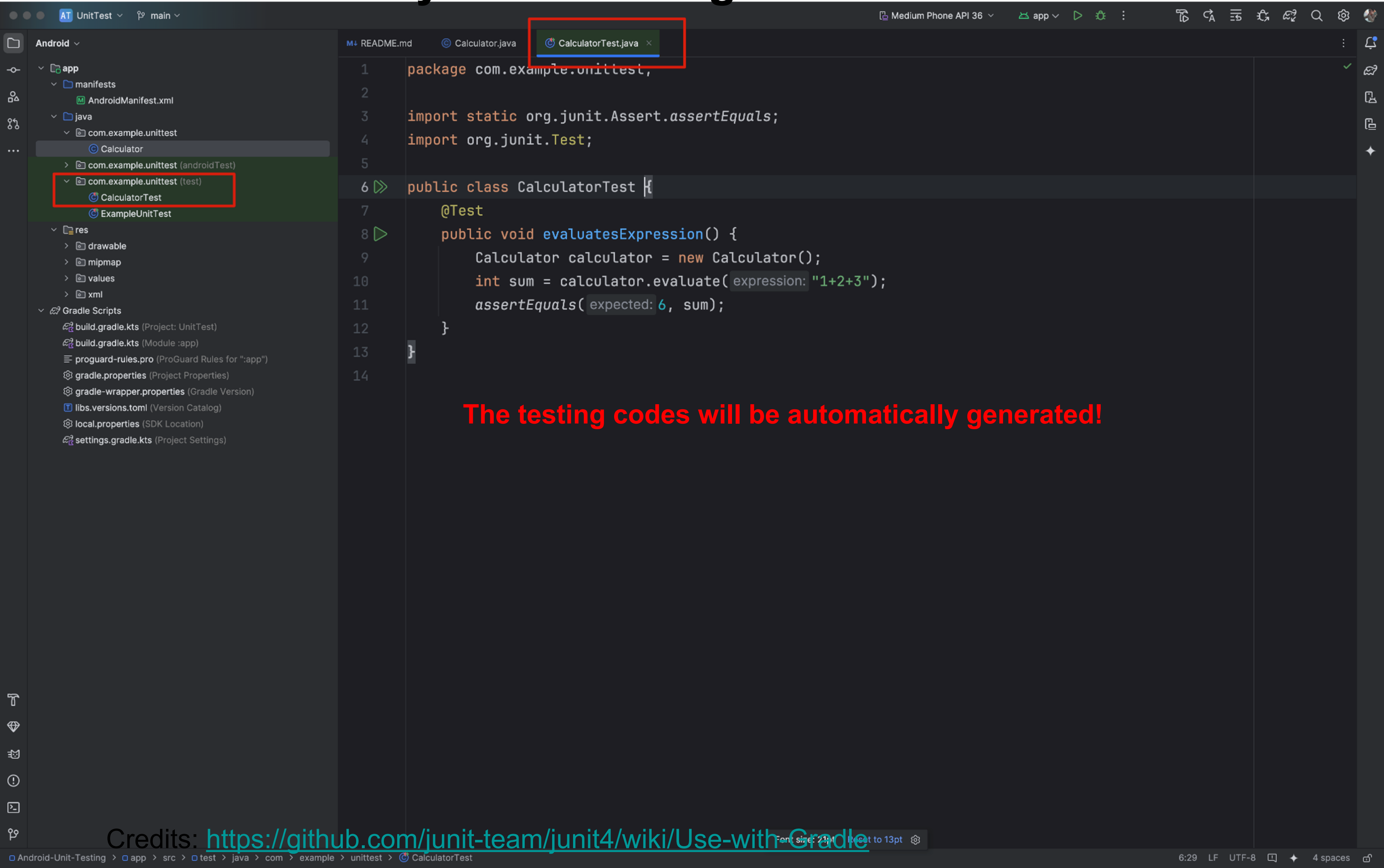
Testing library: JUnit4
Class name: CalculatorTest
Superclass:
Destination package: com.example.unittest
Generate: setUp/@Before, tearDown/@After
Generate test methods for: Show inherited methods
Member: evaluate(expression:String):int

UnitTest
main
Medium Phone API 36
app
Android
app
manifests
AndroidManifest.xml
java
com.example.unittest
Calculator
com.example.unittest (androidTest)
com.example.unittest (test)
CalculatorTest
ExampleUnitTest
res
drawable
mipmap
values
xml
Gradle Scripts
build.gradle.kts (Project: UnitTest)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
README.md
Calculator.java
Cancel
OK
Android-Unit-Testing > app > src > main > java > com > example > unittest > Calculator
3:21 LF UTF-8 4 spaces

right click on the Class name and choose "Generate -> Test" and the boilerplate of the test will be generated in the correct place

Credits: https://github.com/junit-team/junit4/wiki/Use-with-Gradle

Part 2 — Today: Unit Testing in Android


UnitTest
main
Medium Phone API 36
app
Android
README.md
Calculator.java
CalculatorTest.java
app
manifests
AndroidManifest.xml
java
com.example.unittest
Calculator
com.example.unittest (androidTest)
com.example.unittest (test)
CalculatorTest
ExampleUnitTest
res
drawable
mipmap
values
xml
Gradle Scripts
build.gradle.kts (Project: UnitTest)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
Android-Unit-Testing > app > src > test > java > com > example > unittest > CalculatorTest
6:29 LF UTF-8 4 spaces

```java
package com.example.unittest;

import static org.junit.Assert.assertEquals;
import org.junit.Test;

public class CalculatorTest {
    @Test
    public void evaluatesExpression() {
        Calculator calculator = new Calculator();
        int sum = calculator.evaluate( expression: "1+2+3");
        assertEquals( expected: 6, sum);
    }
}
```

The testing codes will be automatically generated!

Credits: https://github.com/junit-team/junit4/wiki/Use-with-Gradle

Part 2 — Today: Unit Testing in Android

Open your terminal and go to the project folder

```
$ cd Desktop/UnitTest/
```

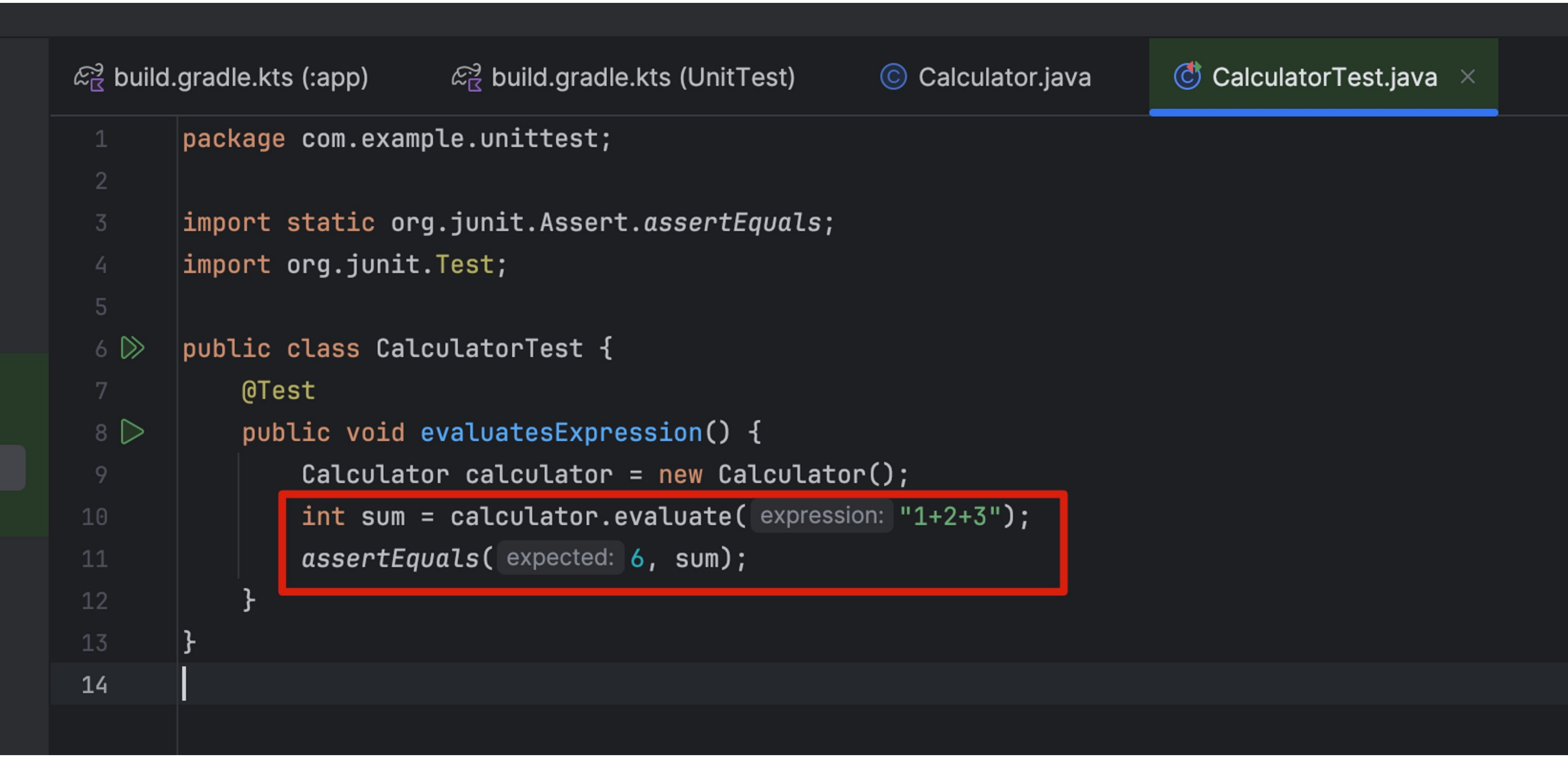
Then run the unit testing

```
$ ./gradlew test
```

Check the code if you need it: `Android-Unit-Testing`

Credits: https://github.com/junit-team/junit4/wiki/Use-with-Gradle

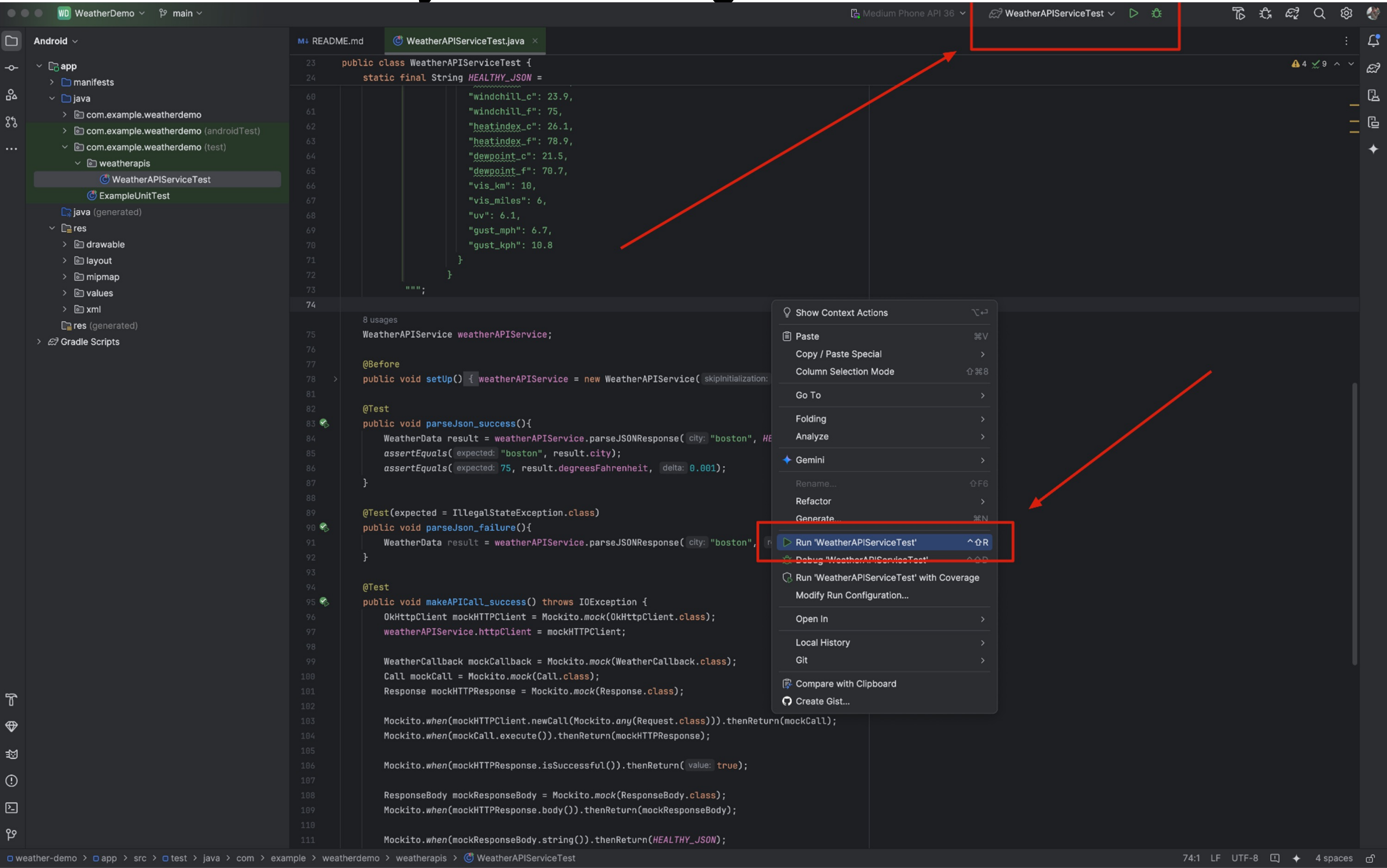
Part 2 — Today: Unit Testing in Android



```java
package com.example.unittest;

import static org.junit.Assert.assertEquals;
import org.junit.Test;

public class CalculatorTest {
    @Test
    public void evaluatesExpression() {
        Calculator calculator = new Calculator();
        int sum = calculator.evaluate( expression: "1+2+3");
        assertEquals( expected: 6, sum);
    }
}
```

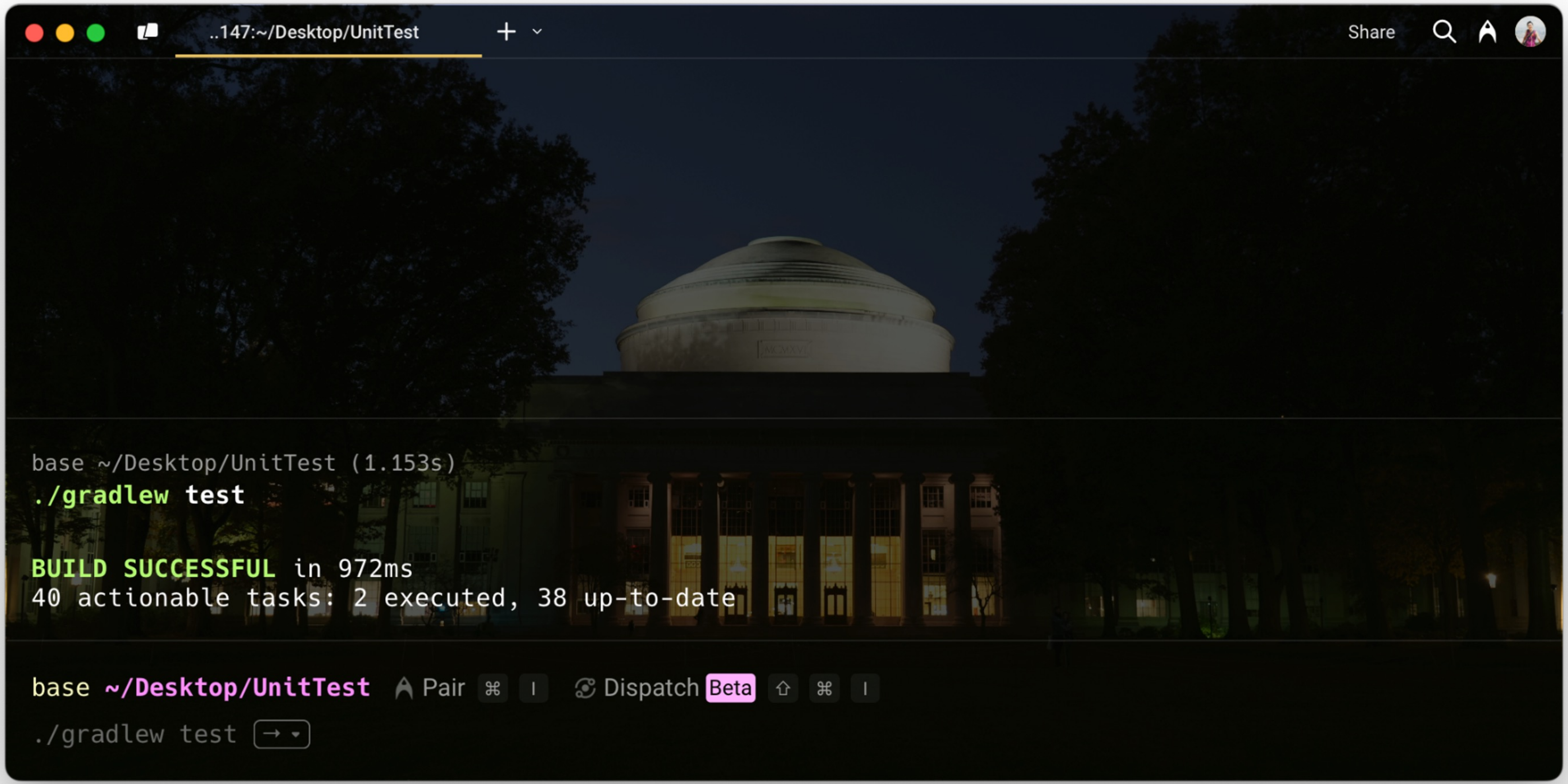
Credits: https://github.com/junit-team/junit4/wiki/Use-with-Gradle

Part 2 — Today: Unit Testing in Android



```
public class WeatherAPIServiceTest {
    static final String HEALTHY_JSON =
                    "windchill_c": 23.9,
                    "windchill_f": 75,
                    "heatindex_c": 26.1,
                    "heatindex_f": 78.9,
                    "dewpoint_c": 21.5,
                    "dewpoint_f": 70.7,
                    "vis_km": 10,
                    "vis_miles": 6,
                    "uv": 6.1,
                    "gust_mph": 6.7,
                    "gust_kph": 10.8
                }
            }
        """;

    8 usages
    WeatherAPIService weatherAPIService;

    @Before
    public void setUp() { weatherAPIService = new WeatherAPIService( skipInitialization:

    @Test
    public void parseJson_success(){
        WeatherData result = weatherAPIService.parseJSONResponse( city: "boston", HE
        assertEquals( expected: "boston", result.city);
        assertEquals( expected: 75, result.degreesFahrenheit, delta: 0.001);
    }

    @Test(expected = IllegalStateException.class)
    public void parseJson_failure(){
        WeatherData result = weatherAPIService.parseJSONResponse( city: "boston",
    }

    @Test
    public void makeAPICall_success() throws IOException {
        OkHttpClient mockHTTPClient = Mockito.mock(OkHttpClient.class);
        weatherAPIService.httpClient = mockHTTPClient;

        WeatherCallback mockCallback = Mockito.mock(WeatherCallback.class);
        Call mockCall = Mockito.mock(Call.class);
        Response mockHTTPResponse = Mockito.mock(Response.class);

        Mockito.when(mockHTTPClient.newCall(Mockito.any(Request.class))).thenReturn(mockCall);
        Mockito.when(mockCall.execute()).thenReturn(mockHTTPResponse);

        Mockito.when(mockHTTPResponse.isSuccessful()).thenReturn( value: true);

        ResponseBody mockResponseBody = Mockito.mock(ResponseBody.class);
        Mockito.when(mockHTTPResponse.body()).thenReturn(mockResponseBody);

        Mockito.when(mockResponseBody.string()).thenReturn(HEALTHY_JSON);
```

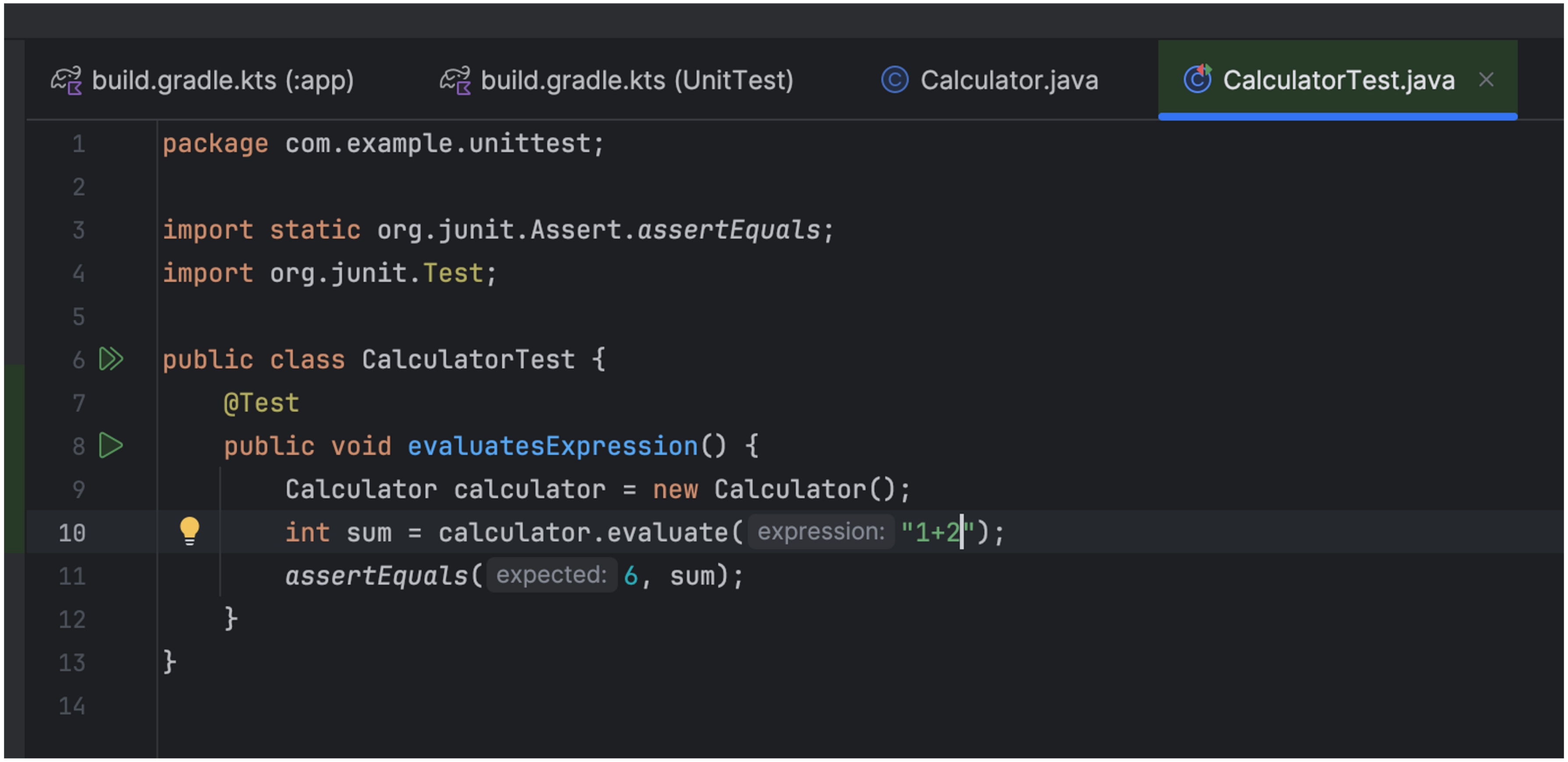
Part 2 — Today: Unit Testing in Android



```
base ~/Desktop/UnitTest (1.153s)
./gradlew test

BUILD SUCCESSFUL in 972ms
40 actionable tasks: 2 executed, 38 up-to-date

base ~/Desktop/UnitTest
./gradlew test
```

Credits: https://github.com/junit-team/junit4/wiki/Use-with-Gradle

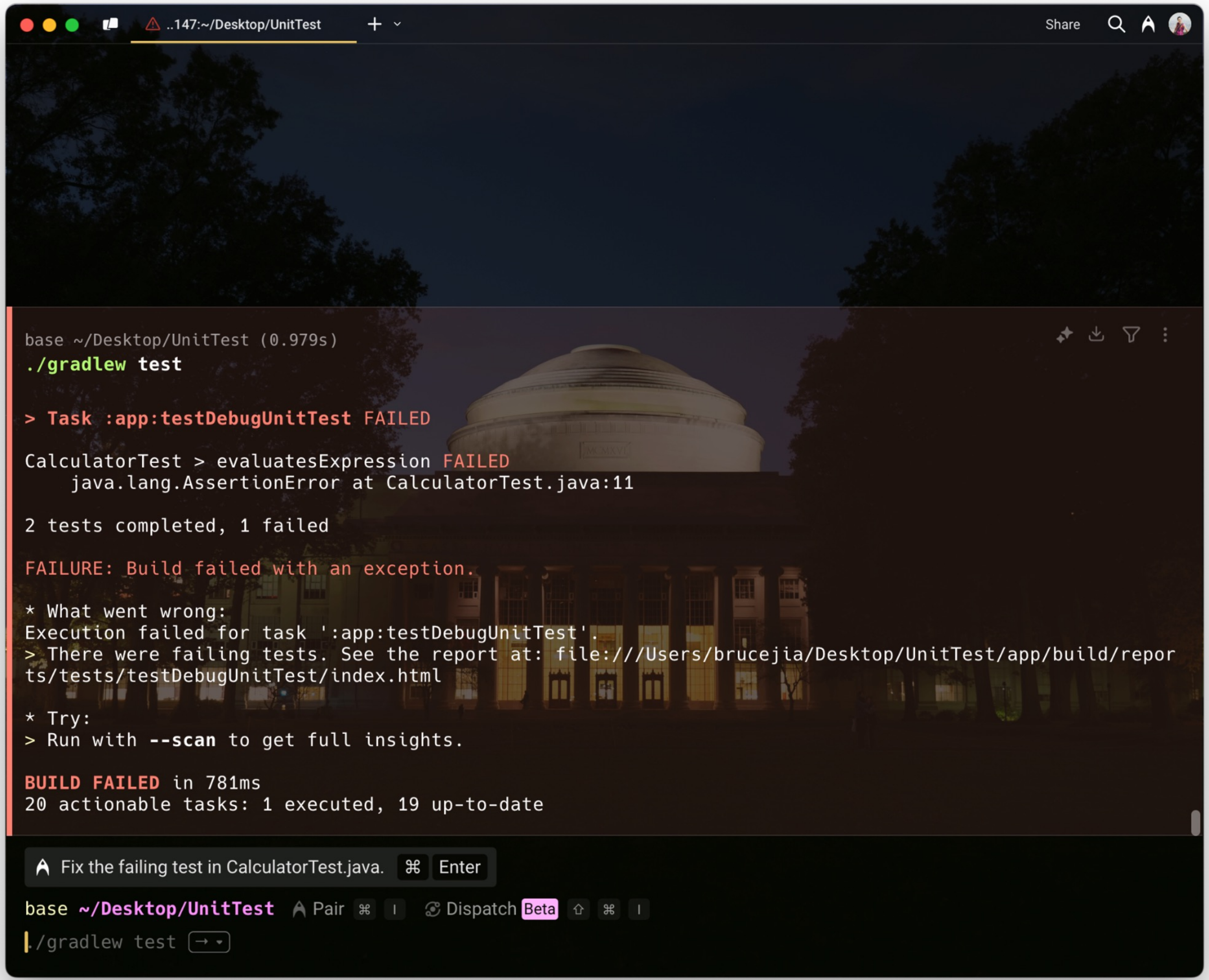
Part 2 — Today: Unit Testing in Android



```java
package com.example.unittest;

import static org.junit.Assert.assertEquals;
import org.junit.Test;

public class CalculatorTest {
    @Test
    public void evaluatesExpression() {
        Calculator calculator = new Calculator();
        int sum = calculator.evaluate( expression: "1+2");
        assertEquals( expected: 6, sum);
    }
}
```

Credits: https://github.com/junit-team/junit4/wiki/Use-with-Gradle

Part 2 — Today: Unit Testing in Android

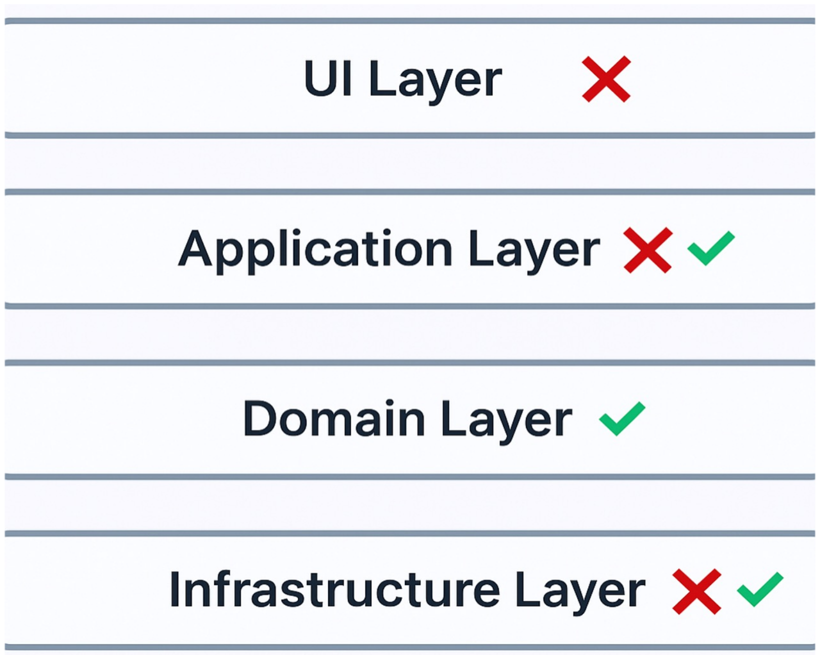


```
base ~/Desktop/UnitTest (0.979s)
./gradlew test

> Task :app:testDebugUnitTest FAILED

CalculatorTest > evaluatesExpression FAILED
    java.lang.AssertionError at CalculatorTest.java:11

2 tests completed, 1 failed

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':app:testDebugUnitTest'.
> There were failing tests. See the report at: file:///Users/brucejia/Desktop/UnitTest/app/build/repor
ts/tests/testDebugUnitTest/index.html

* Try:
> Run with --scan to get full insights.

BUILD FAILED in 781ms
20 actionable tasks: 1 executed, 19 up-to-date
```

Fix the failing test in CalculatorTest.java. ⌘ Enter
base ~/Desktop/UnitTest Pair Dispatch Beta
./gradlew test

Credits: https://github.com/junit-team/junit4/wiki/Use-with-Gradle

Part 3 — Unit Testing Constraints



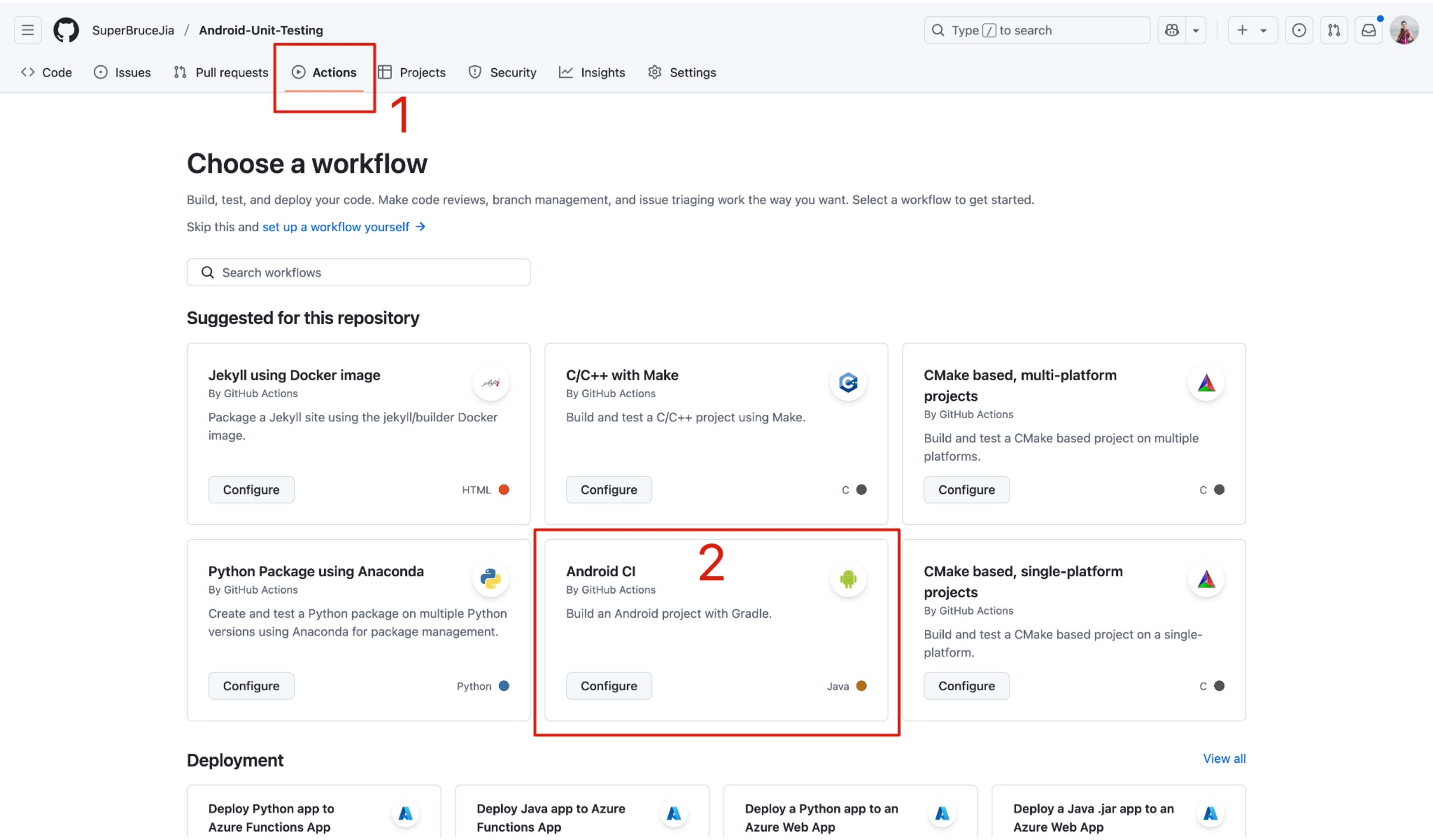
Unit Test is not aware of **Nullability**

Don't use **Singleton**

Static Methods has hidden dependencies, so can not be substituted

Credits: https://www.slideshare.net/slideshow/unit-testing-in-android-156437598/156437598#15

Part 4 — Test Suits with GitHub Actions (`Android CI`)



Choose a workflow

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

Skip this and set up a workflow yourself →

Search workflows

Suggested for this repository

Jekyll using Docker image
By GitHub Actions
Package a Jekyll site using the jekyll/builder Docker image.
Configure — HTML

C/C++ with Make
By GitHub Actions
Build and test a C/C++ project using Make.
Configure — C

CMake based, multi-platform projects
By GitHub Actions
Build and test a CMake based project on multiple platforms.
Configure — C

Python Package using Anaconda
By GitHub Actions
Create and test a Python package on multiple Python versions using Anaconda for package management.
Configure — Python

Android CI
By GitHub Actions
Build an Android project with Gradle.
Configure — Java

CMake based, single-platform projects
By GitHub Actions
Build and test a CMake based project on a single-platform.
Configure — C

Deployment — View all

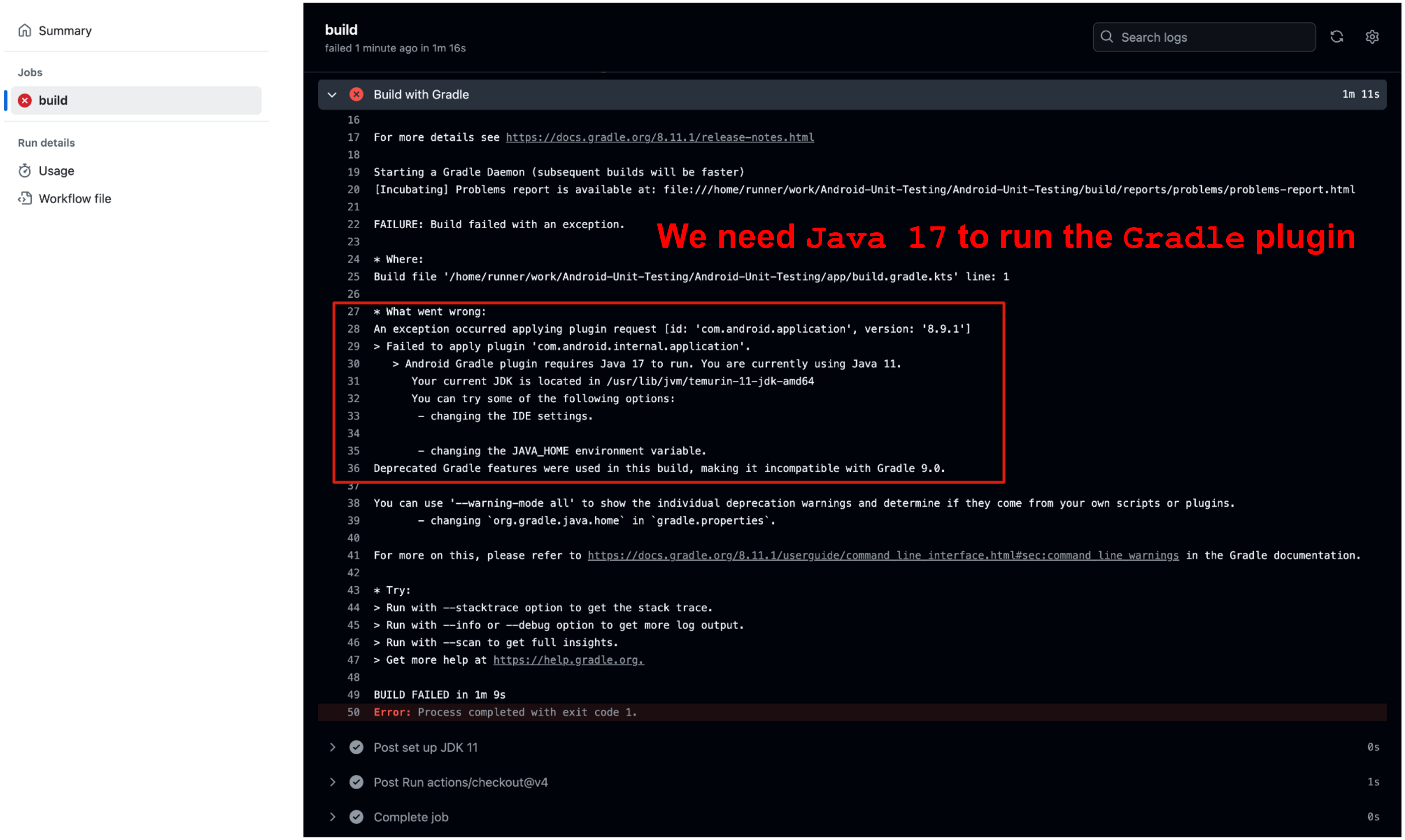
Deploy Python app to Azure Functions App

Deploy Java app to Azure Functions App

Deploy a Python app to an Azure Web App

Deploy a Java .jar app to an Azure Web App

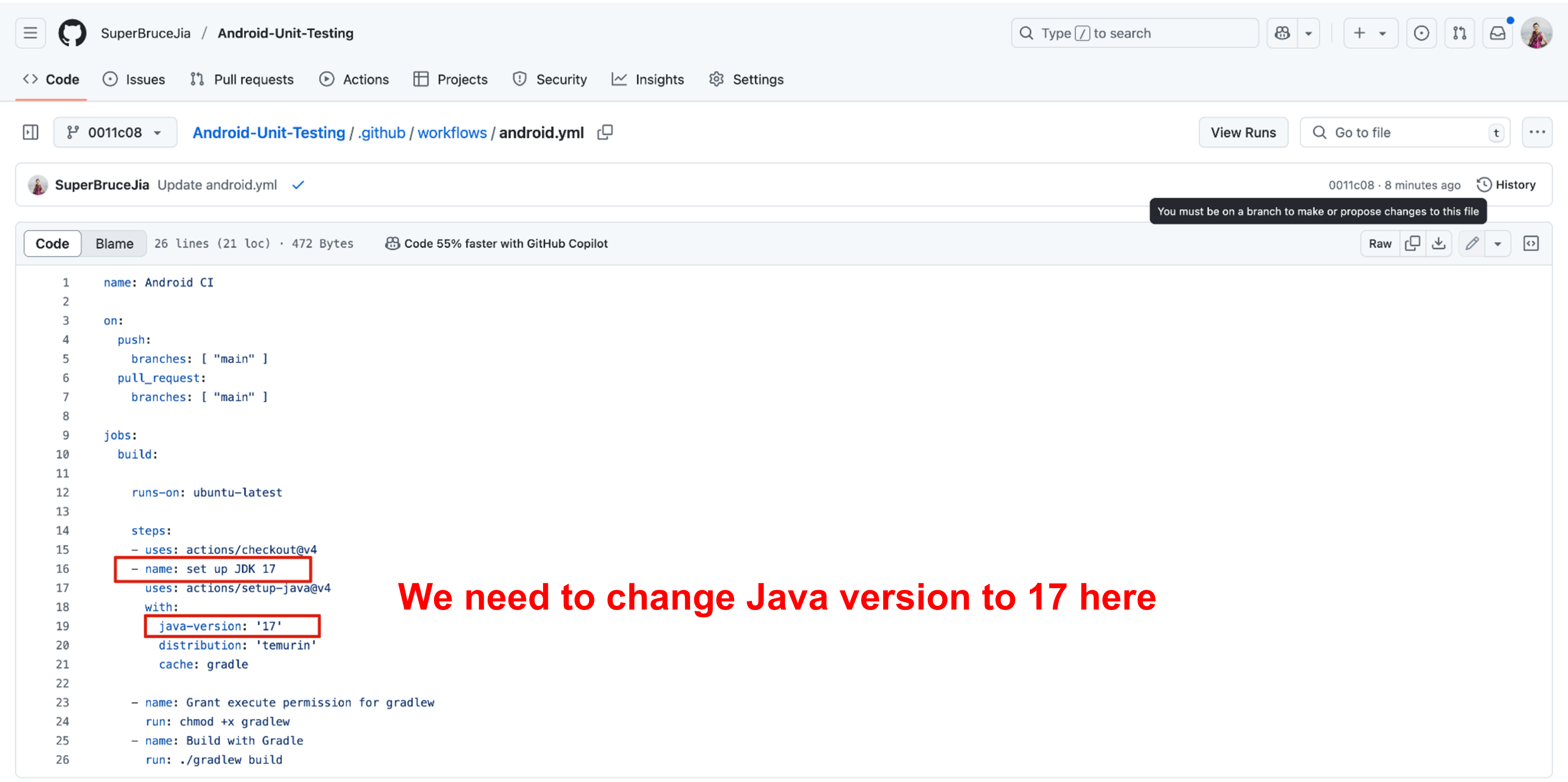
Part 4 — Test Suits with GitHub Actions (`Android CI`)



```
For more details see https://docs.gradle.org/8.11.1/release-notes.html

Starting a Gradle Daemon (subsequent builds will be faster)
[Incubating] Problems report is available at: file:///home/runner/work/Android-Unit-Testing/Android-Unit-Testing/build/reports/problems/problems-report.html

FAILURE: Build failed with an exception.

* Where:
Build file '/home/runner/work/Android-Unit-Testing/Android-Unit-Testing/app/build.gradle.kts' line: 1

* What went wrong:
An exception occurred applying plugin request [id: 'com.android.application', version: '8.9.1']
> Failed to apply plugin 'com.android.internal.application'.
   > Android Gradle plugin requires Java 17 to run. You are currently using Java 11.
      Your current JDK is located in /usr/lib/jvm/temurin-11-jdk-amd64
      You can try some of the following options:
       - changing the IDE settings.

       - changing the JAVA_HOME environment variable.
Deprecated Gradle features were used in this build, making it incompatible with Gradle 9.0.

You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own scripts or plugins.
       - changing `org.gradle.java.home` in `gradle.properties`.

For more on this, please refer to https://docs.gradle.org/8.11.1/userguide/command_line_interface.html#sec:command_line_warnings in the Gradle documentation.

* Try:
> Run with --stacktrace option to get the stack trace.
> Run with --info or --debug option to get more log output.
> Run with --scan to get full insights.
> Get more help at https://help.gradle.org.

BUILD FAILED in 1m 9s
Error: Process completed with exit code 1.
```

We need `Java` 17 to run the `Gradle` plugin

Part 4 — Test Suits with GitHub Actions (`Android CI`)


SuperBruceJia / Android-Unit-Testing
Type / to search
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
0011c08
Android-Unit-Testing / .github / workflows / android.yml
View Runs
Go to file
SuperBruceJia Update android.yml
0011c08 · 8 minutes ago
History
You must be on a branch to make or propose changes to this file
Code
Blame
26 lines (21 loc) · 472 Bytes
Code 55% faster with GitHub Copilot
Raw

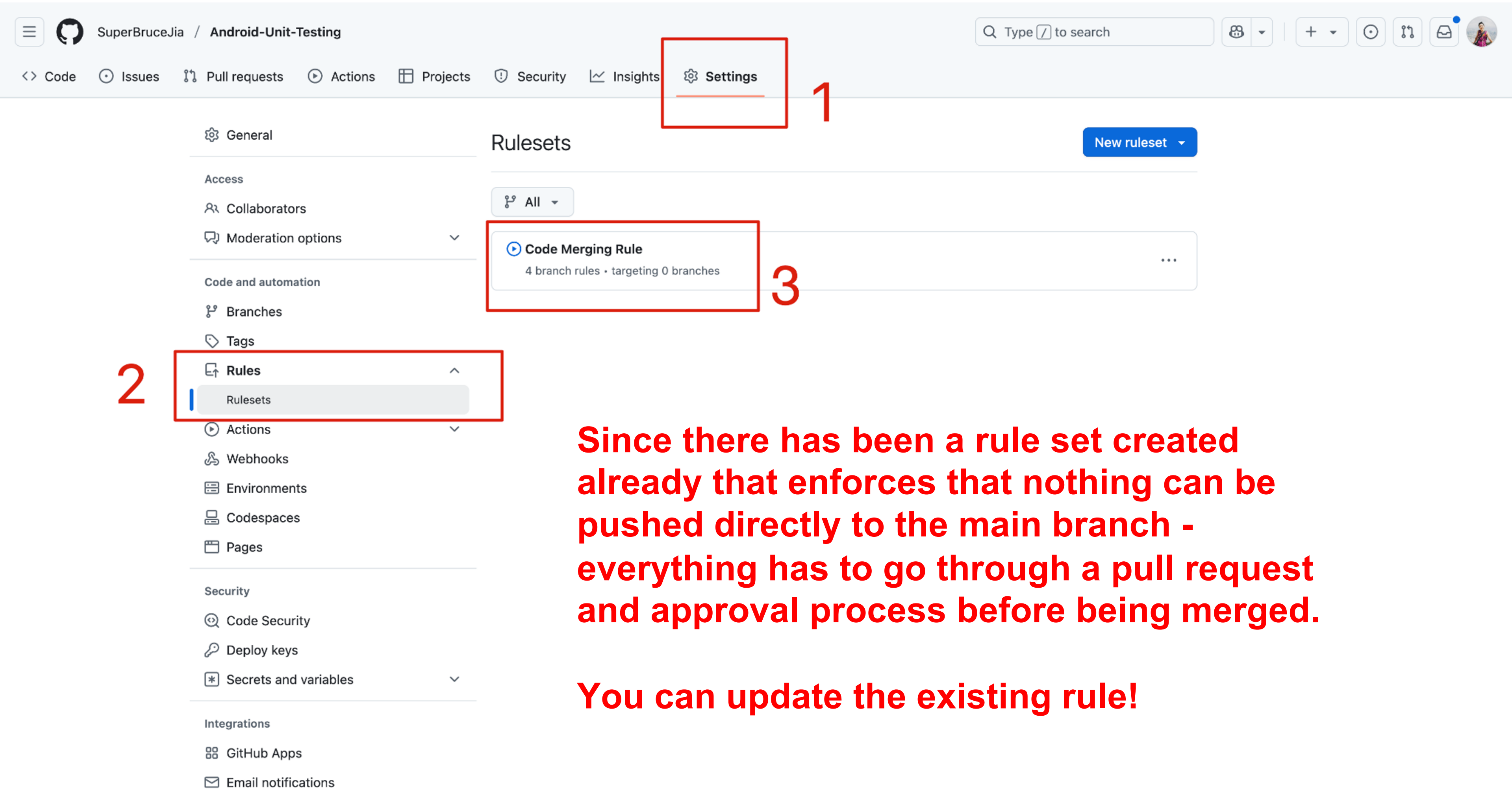
```yaml
name: Android CI

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v4
    - name: set up JDK 17
      uses: actions/setup-java@v4
      with:
        java-version: '17'
        distribution: 'temurin'
        cache: gradle

    - name: Grant execute permission for gradlew
      run: chmod +x gradlew
    - name: Build with Gradle
      run: ./gradlew build
```

We need to change Java version to 17 here

GitHub

"Take care of everything for you"

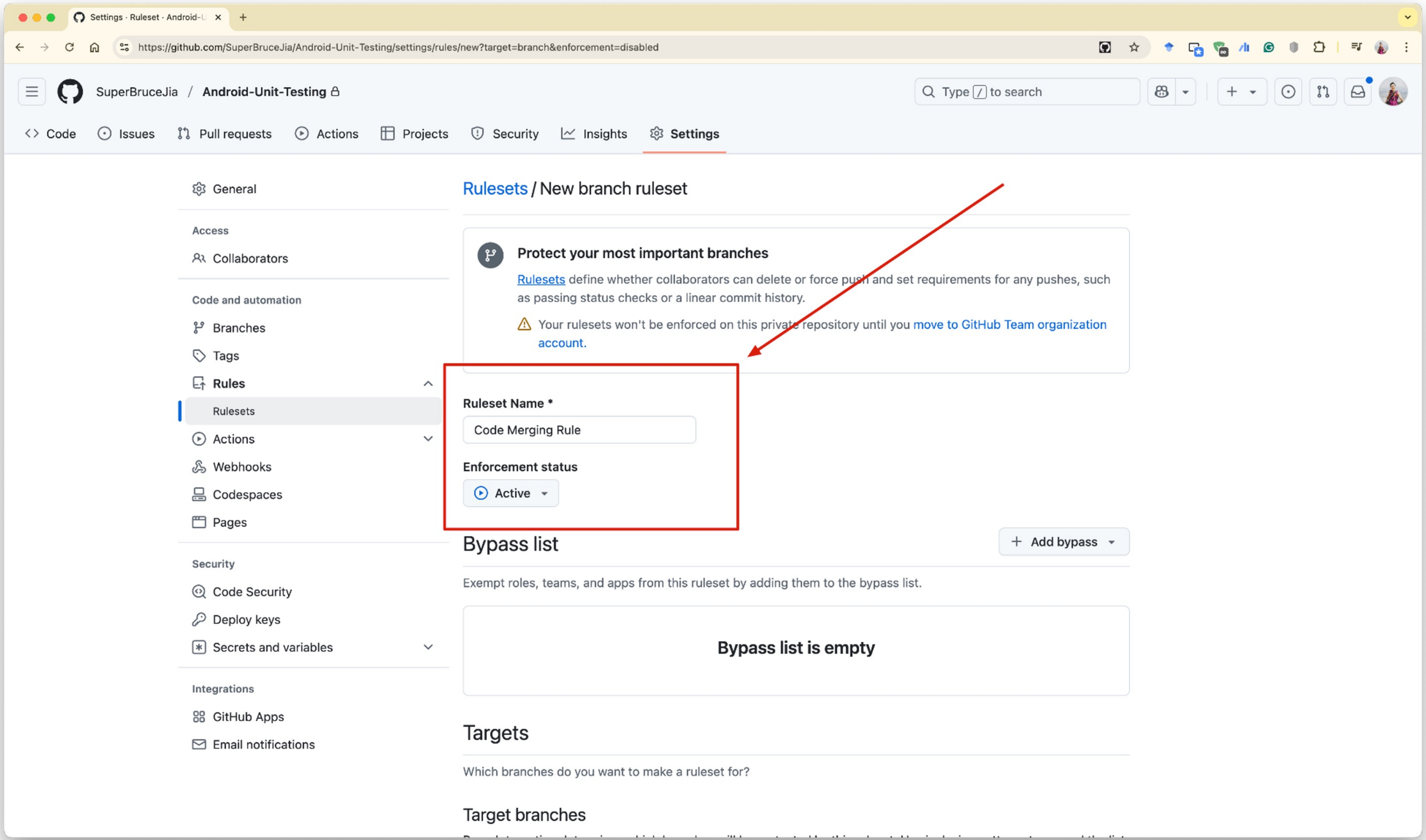
Part 5 — Code Merging after Passing Unit Testing



Since there has been a rule set created already that enforces that nothing can be pushed directly to the main branch - everything has to go through a pull request and approval process before being merged.

You can update the existing rule!

Part 5 — Code Merging after Passing Unit Testing



Settings · Ruleset · Android-U

https://github.com/SuperBruceJia/Android-Unit-Testing/settings/rules/new?target=branch&enforcement=disabled

SuperBruceJia / Android-Unit-Testing

Code | Issues | Pull requests | Actions | Projects | Security | Insights | Settings

General

Access
- Collaborators

Code and automation
- Branches
- Tags
- Rules
  - Rulesets
- Actions
- Webhooks
- Codespaces
- Pages

Security
- Code Security
- Deploy keys
- Secrets and variables

Integrations
- GitHub Apps
- Email notifications

Rulesets / New branch ruleset

Protect your most important branches

Rulesets define whether collaborators can delete or force push and set requirements for any pushes, such as passing status checks or a linear commit history.

⚠ Your rulesets won't be enforced on this private repository until you move to GitHub Team organization account.

Ruleset Name *

Code Merging Rule

Enforcement status

Active

Bypass list — Add bypass

Exempt roles, teams, and apps from this ruleset by adding them to the bypass list.

Bypass list is empty

Targets

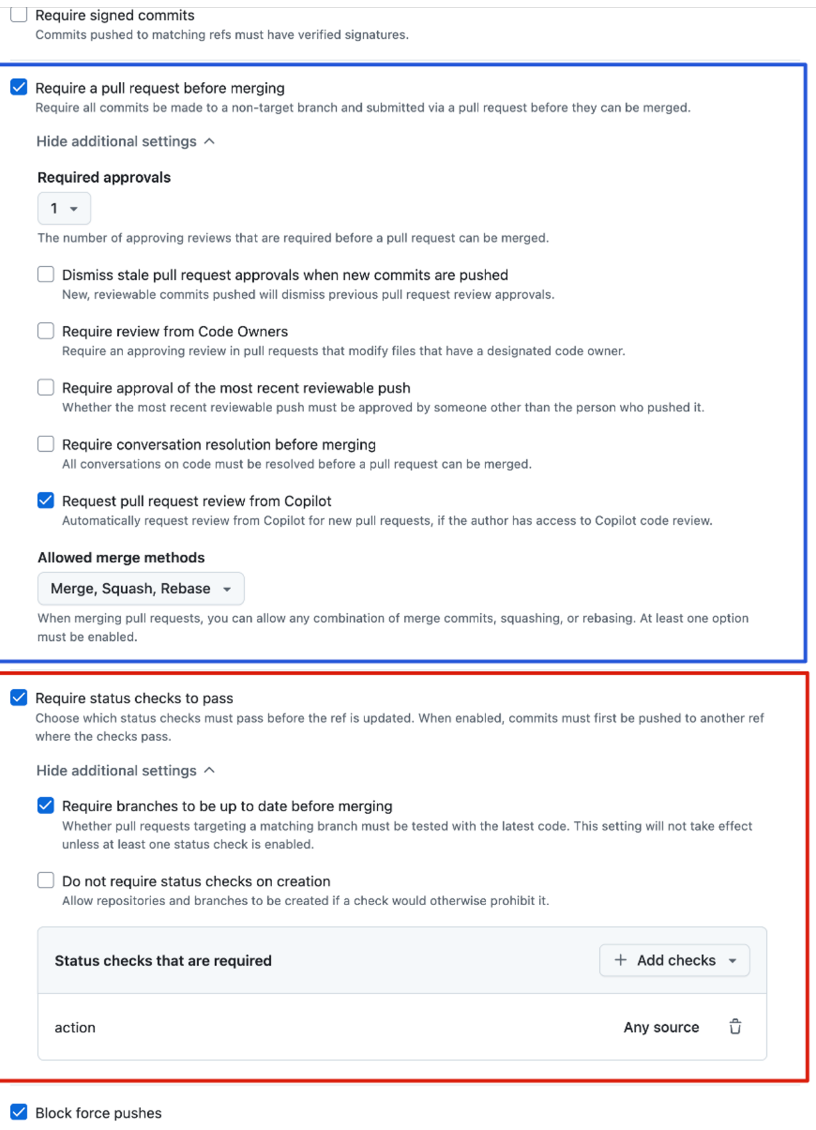
Which branches do you want to make a ruleset for?

Target branches

Part 5 — Code Merging after Passing Unit Testing

This has been set for all of you

pull requests cannot be merged until all the testing has passed



- [ ] Require signed commits
  Commits pushed to matching refs must have verified signatures.
- [✓] Require a pull request before merging
  Require all commits be made to a non-target branch and submitted via a pull request before they can be merged.
  Hide additional settings
  Required approvals
  1
  The number of approving reviews that are required before a pull request can be merged.
  - [ ] Dismiss stale pull request approvals when new commits are pushed
    New, reviewable commits pushed will dismiss previous pull request review approvals.
  - [ ] Require review from Code Owners
    Require an approving review in pull requests that modify files that have a designated code owner.
  - [ ] Require approval of the most recent reviewable push
    Whether the most recent reviewable push must be approved by someone other than the person who pushed it.
  - [ ] Require conversation resolution before merging
    All conversations on code must be resolved before a pull request can be merged.
  - [✓] Request pull request review from Copilot
    Automatically request review from Copilot for new pull requests, if the author has access to Copilot code review.

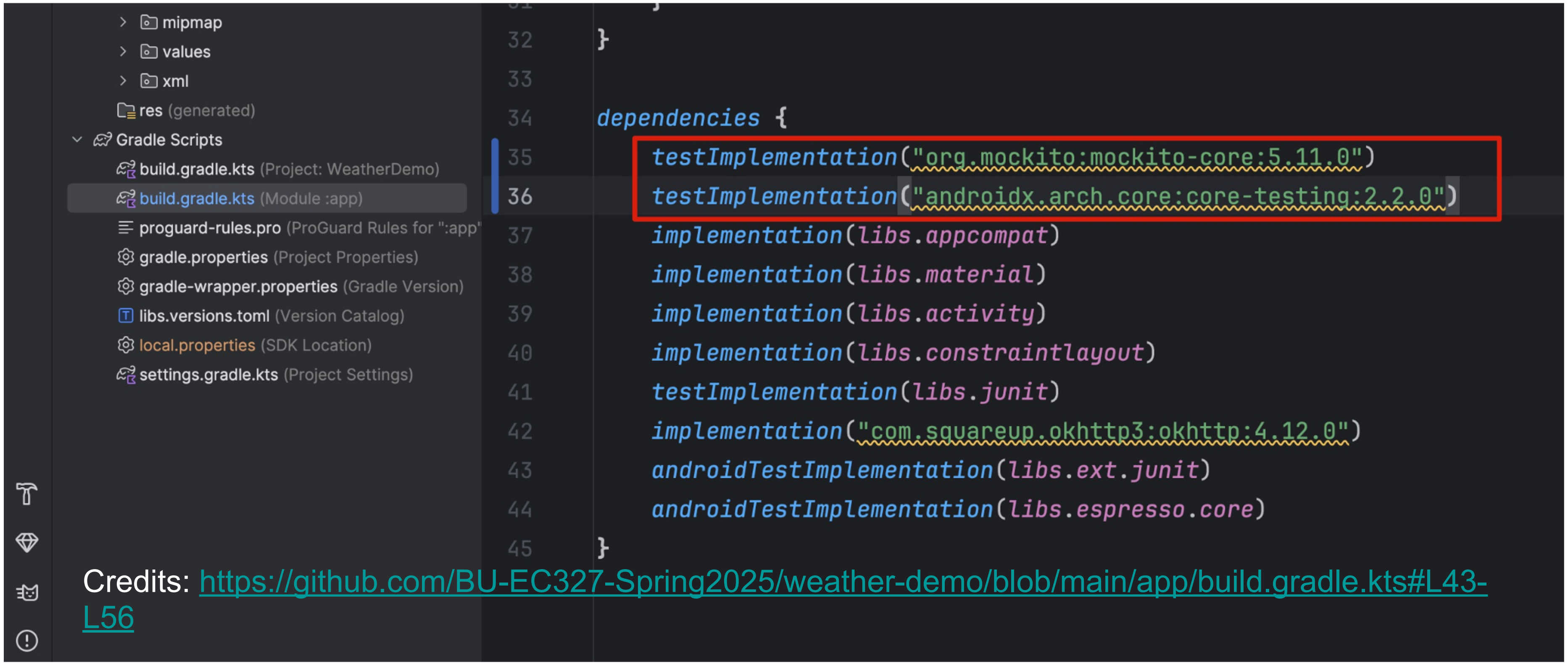
  Allowed merge methods
  Merge, Squash, Rebase
  When merging pull requests, you can allow any combination of merge commits, squashing, or rebasing. At least one option must be enabled.
- [✓] Require status checks to pass
  Choose which status checks must pass before the ref is updated. When enabled, commits must first be pushed to another ref where the checks pass.
  Hide additional settings
  - [✓] Require branches to be up to date before merging
    Whether pull requests targeting a matching branch must be tested with the latest code. This setting will not take effect unless at least one status check is enabled.
  - [ ] Do not require status checks on creation
    Allow repositories and branches to be created if a check would otherwise prohibit it.

Status checks that are required	+ Add checks
action	Any source
- [✓] Block force pushes

Part 6 — Weather Demo Unit Testing

We need these two dependencies in the `build.gradle` file:

```
testImplementation("org.mockito:mockito-core:5.11.0")
testImplementation("androidx.arch.core:core-testing:2.2.0")
```



mipmap
values
xml
res (generated)
Gradle Scripts
build.gradle.kts (Project: WeatherDemo)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app"
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)

```
}

dependencies {
    testImplementation("org.mockito:mockito-core:5.11.0")
    testImplementation("androidx.arch.core:core-testing:2.2.0")
    implementation(libs.appcompat)
    implementation(libs.material)
    implementation(libs.activity)
    implementation(libs.constraintlayout)
    testImplementation(libs.junit)
    implementation("com.squareup.okhttp3:okhttp:4.12.0")
    androidTestImplementation(libs.ext.junit)
    androidTestImplementation(libs.espresso.core)
}
```

Credits: https://github.com/BU-EC327-Spring2025/weather-demo/blob/main/app/build.gradle.kts#L43-L56

Part 6 — Weather Demo Unit Testing

import necessary packages and dependencies:

For `Mockito` framework:

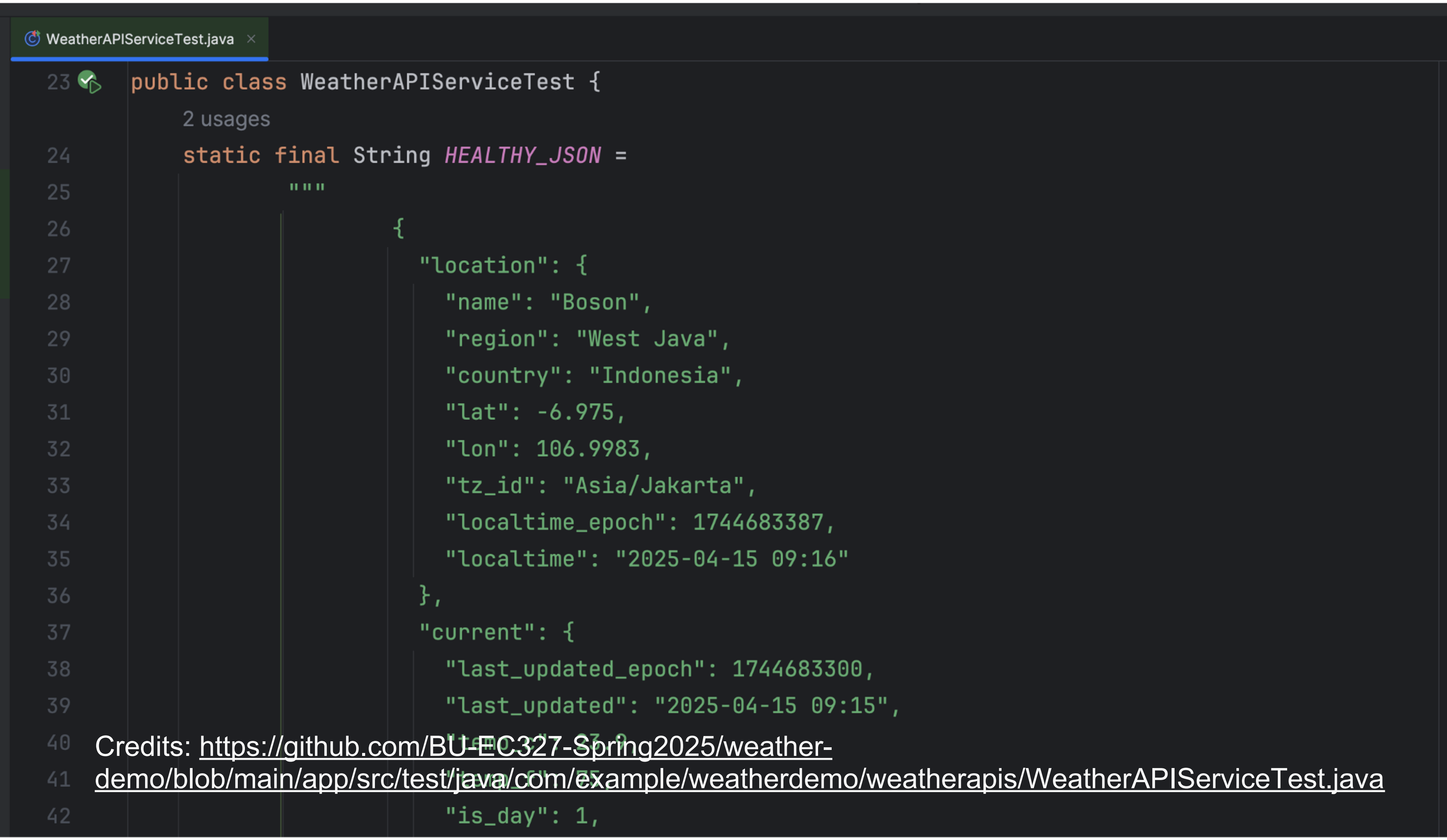
```
import org.junit.Before;
import org.junit.Test;
import org.mockito.Mockito;
```

For `@RunWith(AndroidJUnit4.class)` framework:

```
import androidx.test.ext.junit.runners.AndroidJUnit4;
import org.junit.Test;
import org.junit.runner.RunWith;
```

Credits:
[1]https://github.com/BU-EC327-Spring2025/weather-demo/blob/main/app/src/test/java/com/example/weatherdemo/weatherapis/WeatherAPIServiceTest.java
[2]https://github.com/BU-EC327-Spring2025/weather-demo/blob/main/app/src/androidTest/java/com/example/weatherdemo/DependencyManagerTest.java

Part 6 — Weather Demo Unit Testing

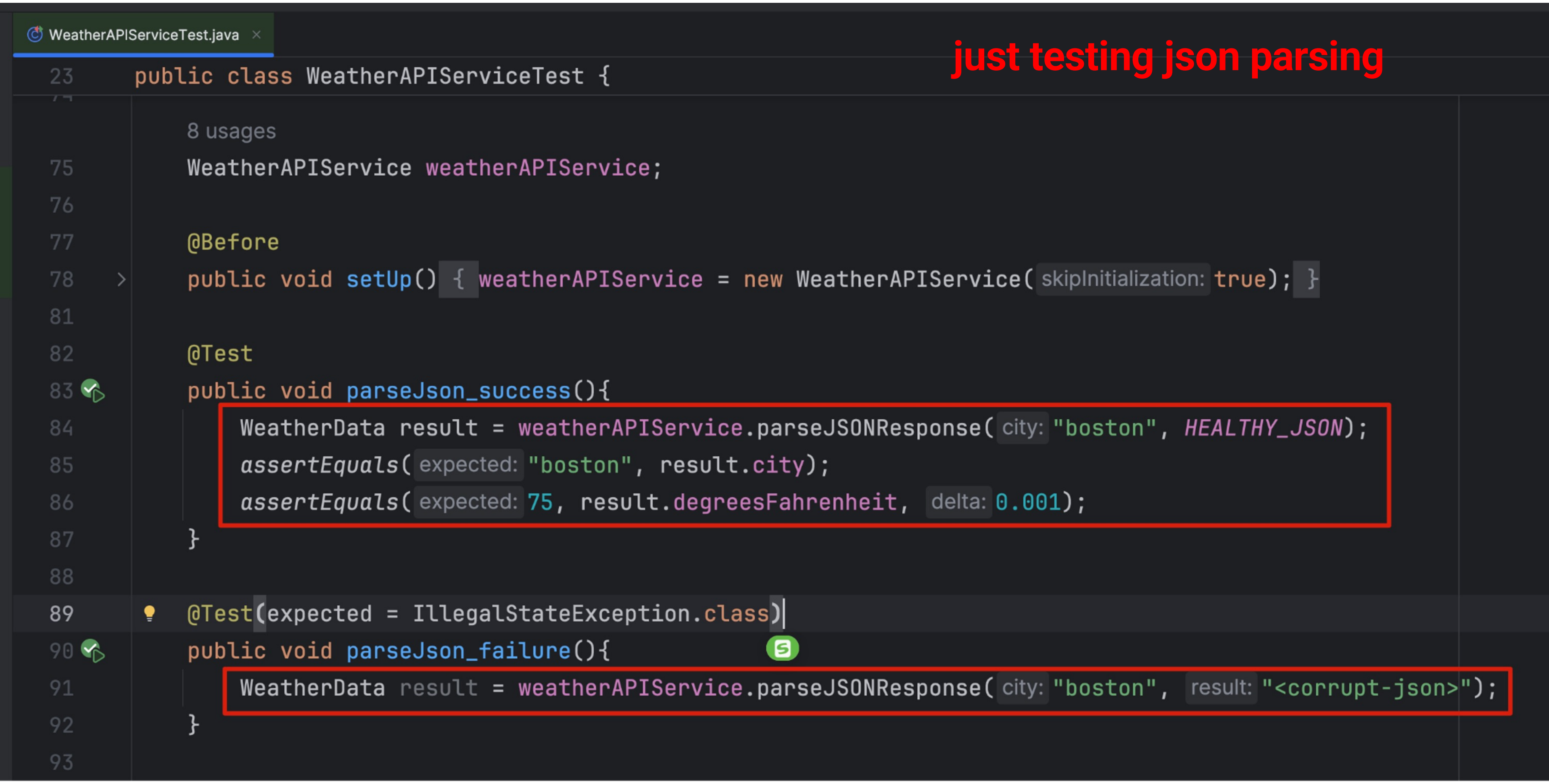


WeatherAPIServiceTest.java

```
public class WeatherAPIServiceTest {
    2 usages
    static final String HEALTHY_JSON =
            """
                    {
                      "location": {
                        "name": "Boson",
                        "region": "West Java",
                        "country": "Indonesia",
                        "lat": -6.975,
                        "lon": 106.9983,
                        "tz_id": "Asia/Jakarta",
                        "localtime_epoch": 1744683387,
                        "localtime": "2025-04-15 09:16"
                      },
                      "current": {
                        "last_updated_epoch": 1744683300,
                        "last_updated": "2025-04-15 09:15",
                        "temp_c": 23.9,
                        "temp_f": 75.[illegible]
                        "is_day": 1,
```

Credits: https://github.com/BU-EC327-Spring2025/weather-demo/blob/main/app/src/test/java/com/example/weatherdemo/weatherapis/WeatherAPIServiceTest.java

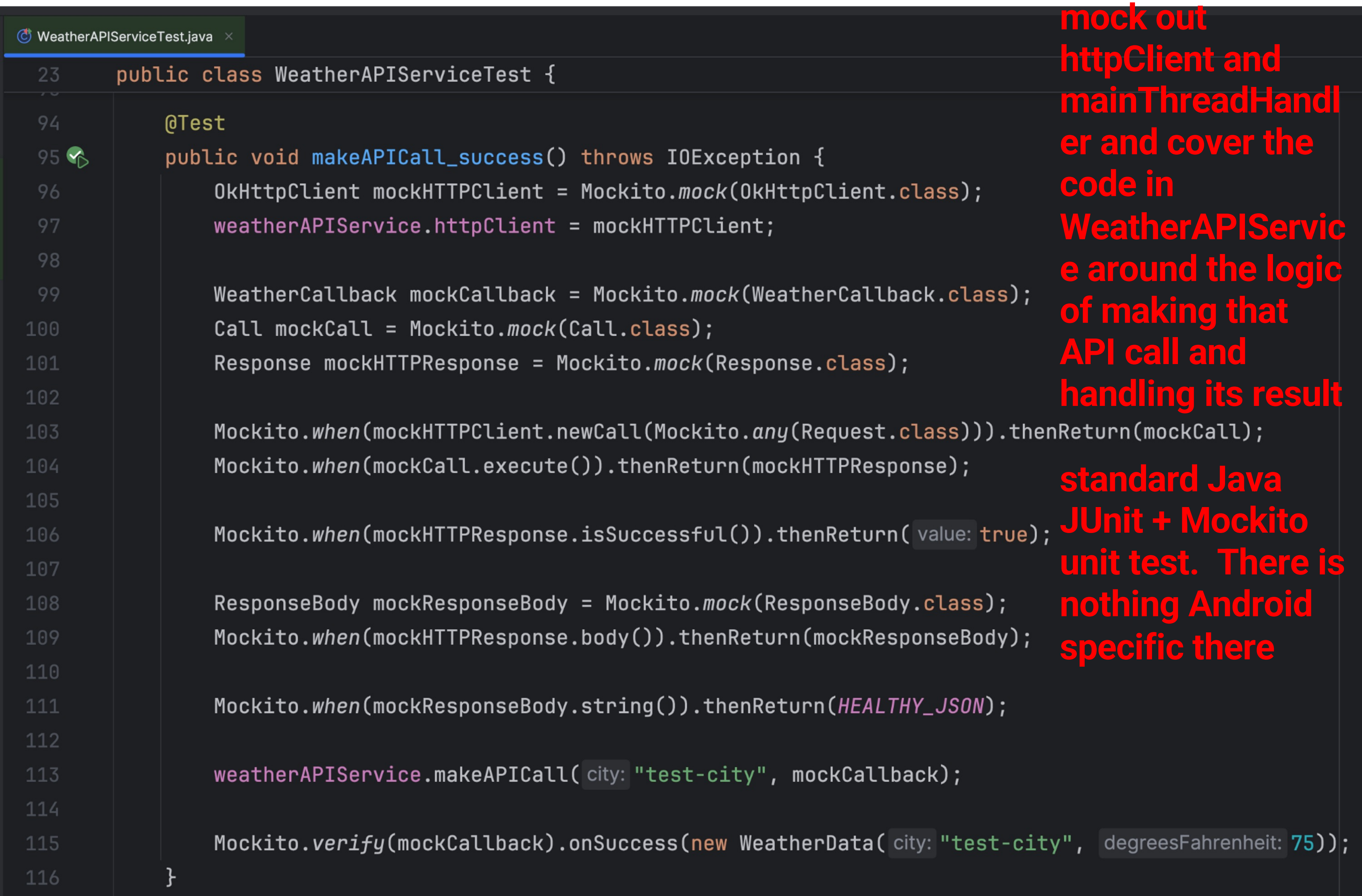
Part 6 — Weather Demo Unit Testing


WeatherAPIServiceTest.java

just testing json parsing

```
public class WeatherAPIServiceTest {

    8 usages
    WeatherAPIService weatherAPIService;

    @Before
    public void setUp() { weatherAPIService = new WeatherAPIService( skipInitialization: true); }

    @Test
    public void parseJson_success(){
        WeatherData result = weatherAPIService.parseJSONResponse( city: "boston", HEALTHY_JSON);
        assertEquals( expected: "boston", result.city);
        assertEquals( expected: 75, result.degreesFahrenheit,  delta: 0.001);
    }

    @Test(expected = IllegalStateException.class)
    public void parseJson_failure(){
        WeatherData result = weatherAPIService.parseJSONResponse( city: "boston",  result: "<corrupt-json>");
    }
```

Part 6 — Weather Demo Unit Testing



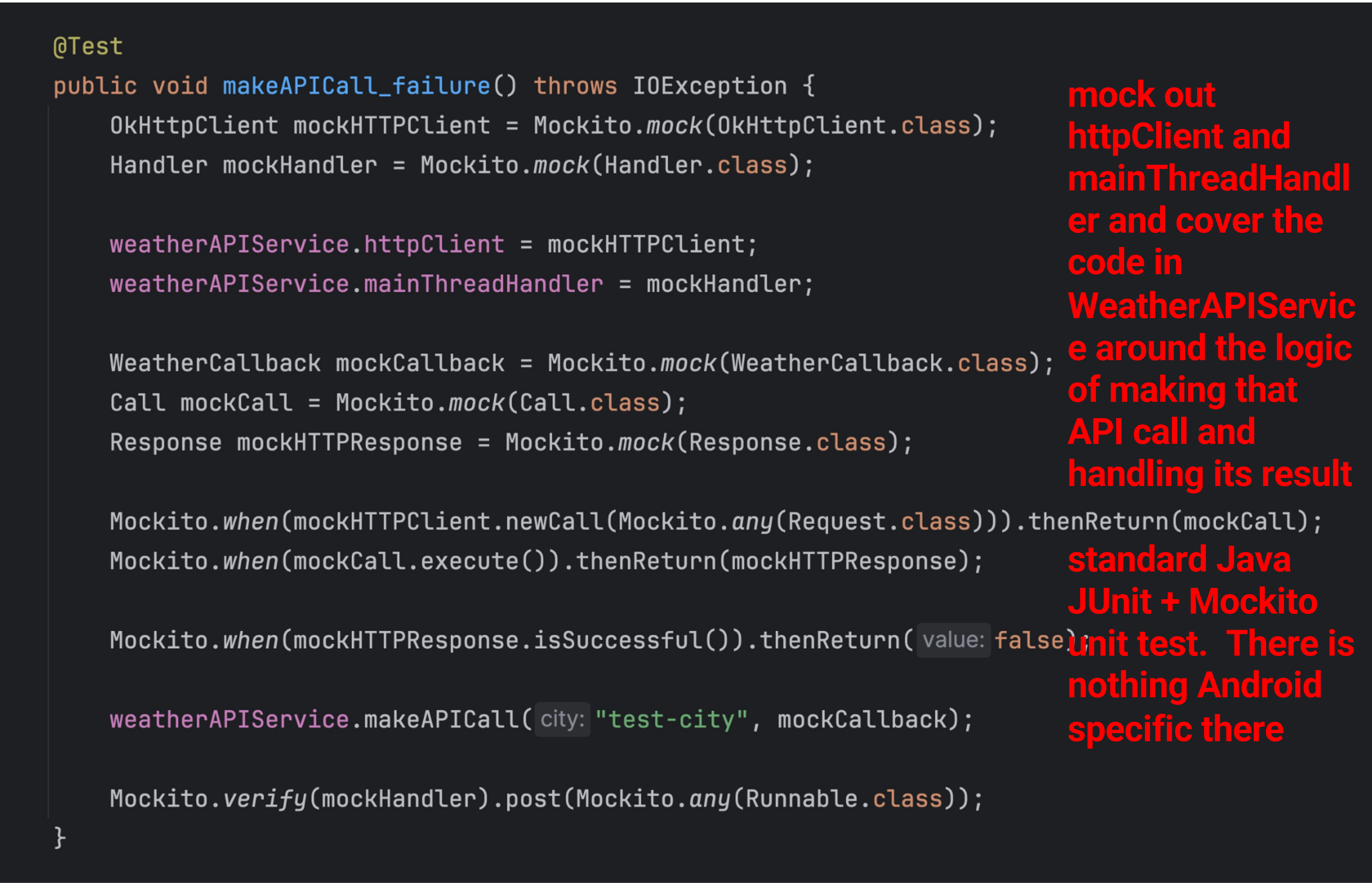
WeatherAPIServiceTest.java

```
public class WeatherAPIServiceTest {

    @Test
    public void makeAPICall_success() throws IOException {
        OkHttpClient mockHTTPClient = Mockito.mock(OkHttpClient.class);
        weatherAPIService.httpClient = mockHTTPClient;

        WeatherCallback mockCallback = Mockito.mock(WeatherCallback.class);
        Call mockCall = Mockito.mock(Call.class);
        Response mockHTTPResponse = Mockito.mock(Response.class);

        Mockito.when(mockHTTPClient.newCall(Mockito.any(Request.class))).thenReturn(mockCall);
        Mockito.when(mockCall.execute()).thenReturn(mockHTTPResponse);

        Mockito.when(mockHTTPResponse.isSuccessful()).thenReturn( value: true);

        ResponseBody mockResponseBody = Mockito.mock(ResponseBody.class);
        Mockito.when(mockHTTPResponse.body()).thenReturn(mockResponseBody);

        Mockito.when(mockResponseBody.string()).thenReturn(HEALTHY_JSON);

        weatherAPIService.makeAPICall( city: "test-city", mockCallback);

        Mockito.verify(mockCallback).onSuccess(new WeatherData( city: "test-city", degreesFahrenheit: 75));
    }
```

mock out httpClient and mainThreadHandler and cover the code in WeatherAPIService around the logic of making that API call and handling its result

standard Java JUnit + Mockito unit test. There is nothing Android specific there

Part 6 — Weather Demo Unit Testing



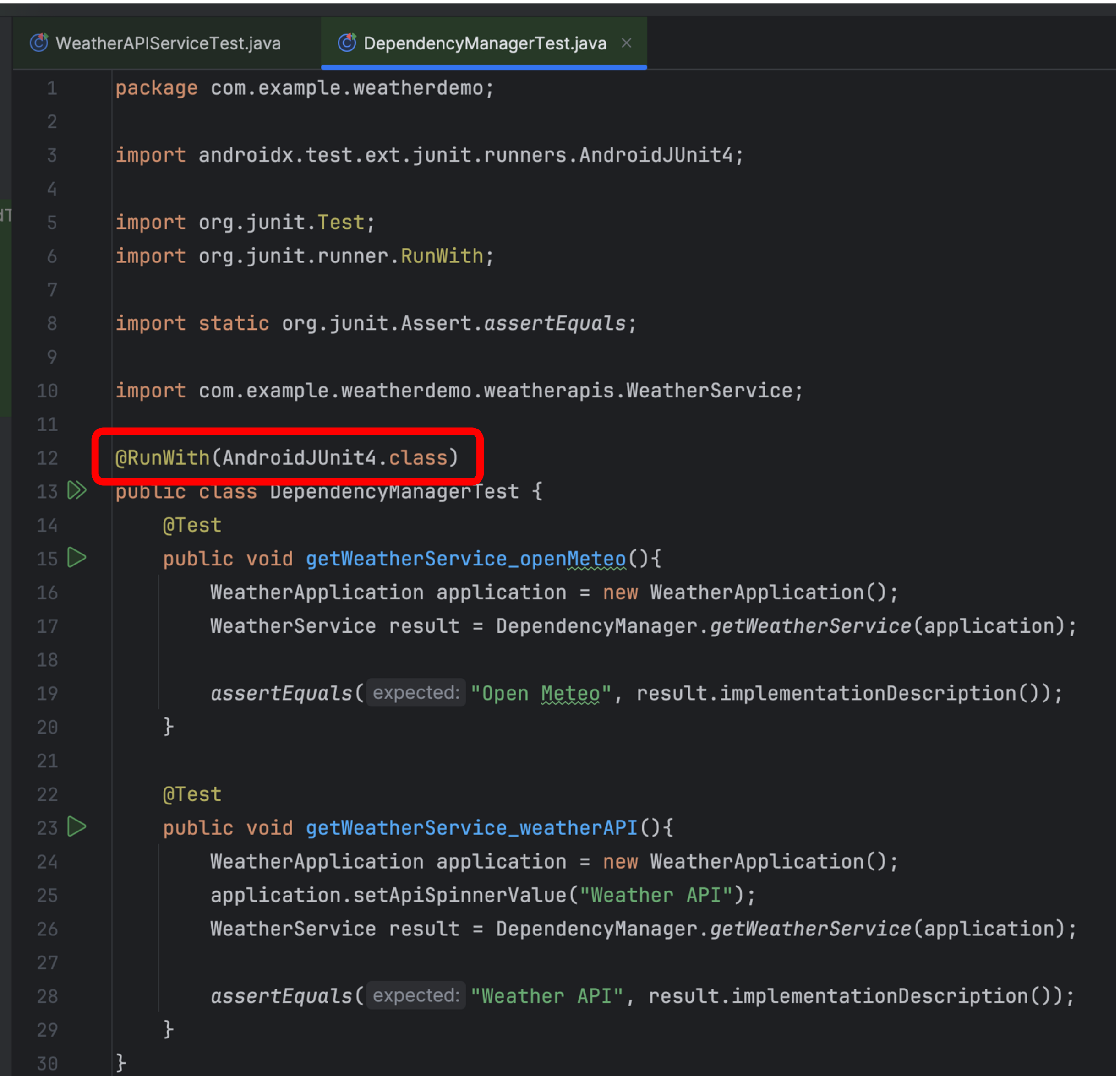
```
@Test
public void makeAPICall_failure() throws IOException {
    OkHttpClient mockHTTPClient = Mockito.mock(OkHttpClient.class);
    Handler mockHandler = Mockito.mock(Handler.class);

    weatherAPIService.httpClient = mockHTTPClient;
    weatherAPIService.mainThreadHandler = mockHandler;

    WeatherCallback mockCallback = Mockito.mock(WeatherCallback.class);
    Call mockCall = Mockito.mock(Call.class);
    Response mockHTTPResponse = Mockito.mock(Response.class);

    Mockito.when(mockHTTPClient.newCall(Mockito.any(Request.class))).thenReturn(mockCall);
    Mockito.when(mockCall.execute()).thenReturn(mockHTTPResponse);

    Mockito.when(mockHTTPResponse.isSuccessful()).thenReturn( value: false);

    weatherAPIService.makeAPICall( city: "test-city", mockCallback);

    Mockito.verify(mockHandler).post(Mockito.any(Runnable.class));
}
```

mock out httpClient and mainThreadHandler and cover the code in WeatherAPIService around the logic of making that API call and handling its result

standard Java JUnit + Mockito unit test. There is nothing Android specific there

Credits: https://github.com/BU-EC327-Spring2025/weather-demo/blob/main/app/src/androidTest/java/com/example/weatherdemo/DependencyManagerTest.java

Part 6 — Weather Demo Unit Testing


WeatherAPIServiceTest.java | DependencyManagerTest.java

```
package com.example.weatherdemo;

import androidx.test.ext.junit.runners.AndroidJUnit4;

import org.junit.Test;
import org.junit.runner.RunWith;

import static org.junit.Assert.assertEquals;

import com.example.weatherdemo.weatherapis.WeatherService;

@RunWith(AndroidJUnit4.class)
public class DependencyManagerTest {
    @Test
    public void getWeatherService_openMeteo(){
        WeatherApplication application = new WeatherApplication();
        WeatherService result = DependencyManager.getWeatherService(application);

        assertEquals( expected: "Open Meteo", result.implementationDescription());
    }

    @Test
    public void getWeatherService_weatherAPI(){
        WeatherApplication application = new WeatherApplication();
        application.setApiSpinnerValue("Weather API");
        WeatherService result = DependencyManager.getWeatherService(application);

        assertEquals( expected: "Weather API", result.implementationDescription());
    }
}
```

- ❑ for anything that touches an Android UI component, or an Android activity, or an Android main thread, we need to use `@RunWith(AndroidJUnit4.class)`
- ❑ these tests will be sigificantly slower because they will need to start the emulator and will get executed on the emnulator instead of just in the JVM

Part 6 — Weather Demo Unit Testing

- ❑ Run a **specific** Instrumented Test Class

```
$ ./gradlew connectedAndroidTest -Pandroid.testInstrumentationRunnerArguments.class=com.example.weatherdemo.DependencyManagerTest
```

- ❑ Run a **specific** Instrumented Test Class

```
$ ./gradlew connectedAndroidTest -Pandroid.testInstrumentationRunnerArguments.class=com.example.weatherdemo.DependencyManagerTest#methodName
```

- ❑ Run **All** Instrumented Tests

```
$ ./gradlew connectedAndroidTest test
```

- ❑ **Clean** all build artifacts and test results

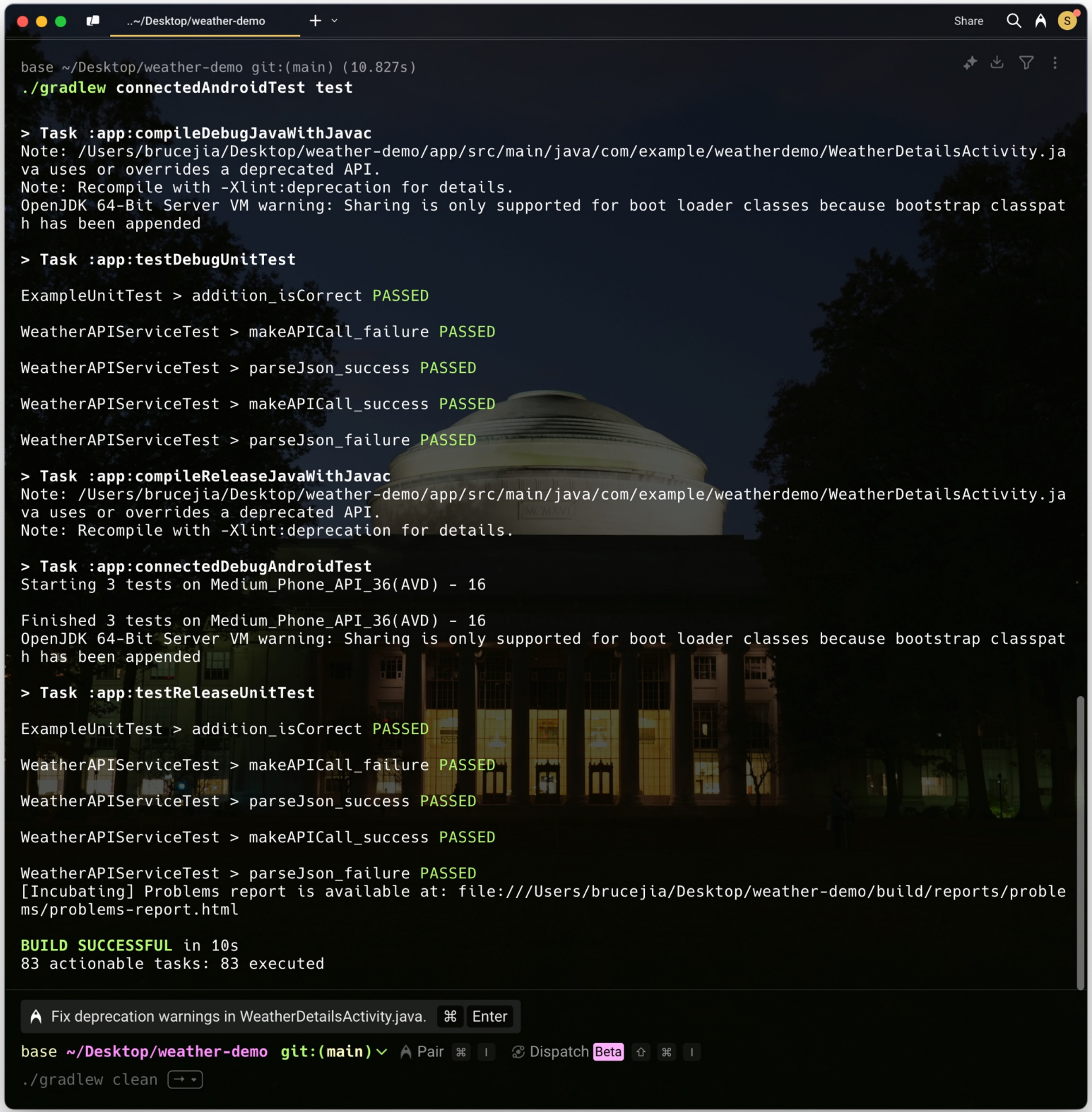
```
$ ./gradlew clean
```

- ❑ **Clean** and then **run** all tests

```
$ ./gradlew clean test connectedAndroidTest
```

Unit Testing in Android with GitHub Action

Instrumented Test Class

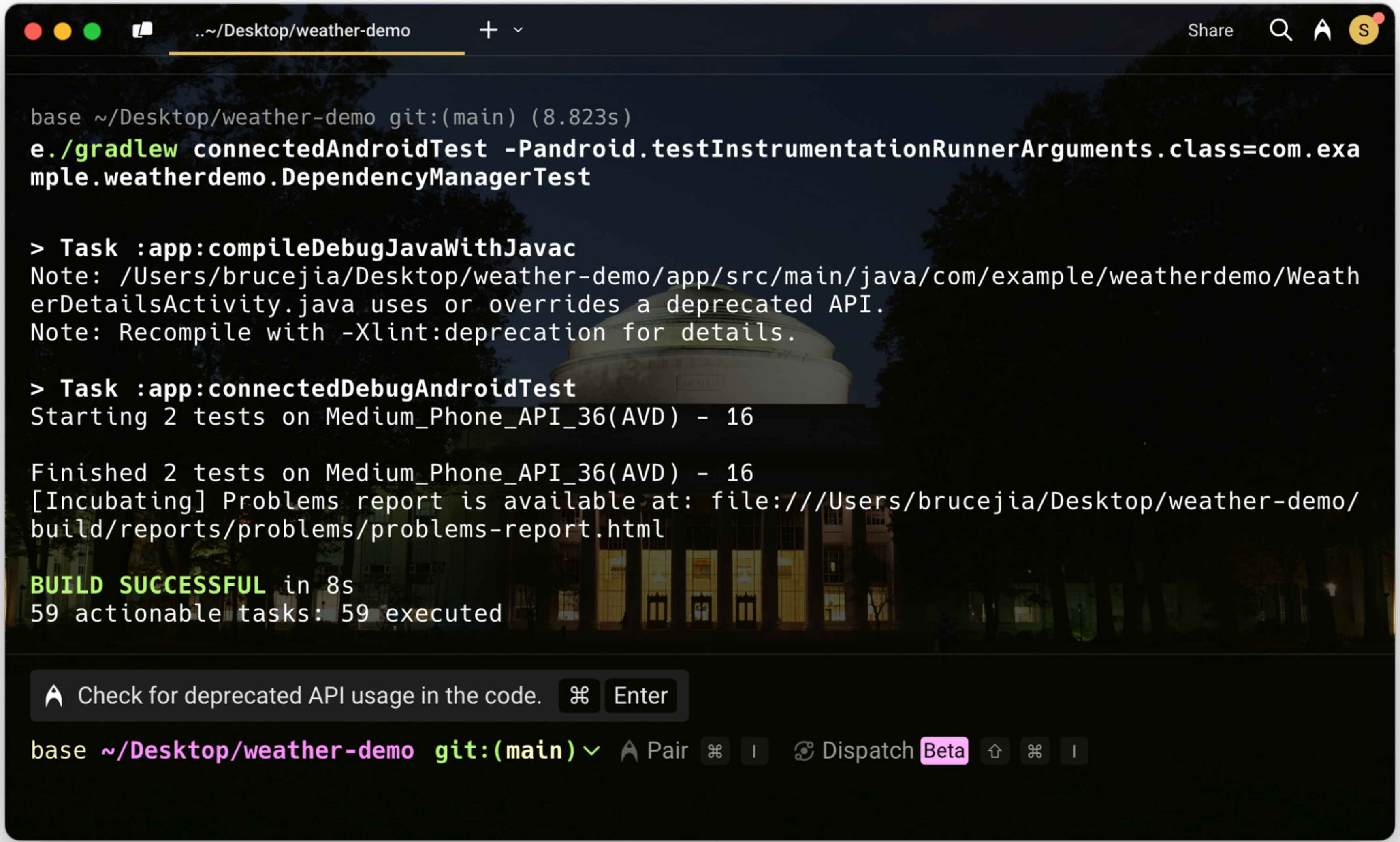


```
base ~/Desktop/weather-demo git:(main) (10.827s)
./gradlew connectedAndroidTest test

> Task :app:compileDebugJavaWithJavac
Note: /Users/brucejia/Desktop/weather-demo/app/src/main/java/com/example/weatherdemo/WeatherDetailsActivity.ja
va uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpat
h has been appended

> Task :app:testDebugUnitTest

ExampleUnitTest > addition_isCorrect PASSED

WeatherAPIServiceTest > makeAPICall_failure PASSED

WeatherAPIServiceTest > parseJson_success PASSED

WeatherAPIServiceTest > makeAPICall_success PASSED

WeatherAPIServiceTest > parseJson_failure PASSED

> Task :app:compileReleaseJavaWithJavac
Note: /Users/brucejia/Desktop/weather-demo/app/src/main/java/com/example/weatherdemo/WeatherDetailsActivity.ja
va uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

> Task :app:connectedDebugAndroidTest
Starting 3 tests on Medium_Phone_API_36(AVD) - 16

Finished 3 tests on Medium_Phone_API_36(AVD) - 16
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpat
h has been appended

> Task :app:testReleaseUnitTest

ExampleUnitTest > addition_isCorrect PASSED

WeatherAPIServiceTest > makeAPICall_failure PASSED

WeatherAPIServiceTest > parseJson_success PASSED

WeatherAPIServiceTest > makeAPICall_success PASSED

WeatherAPIServiceTest > parseJson_failure PASSED
[Incubating] Problems report is available at: file:///Users/brucejia/Desktop/weather-demo/build/reports/proble
ms/problems-report.html

BUILD SUCCESSFUL in 10s
83 actionable tasks: 83 executed
```

Fix deprecation warnings in WeatherDetailsActivity.java. ⌘ Enter

```
base ~/Desktop/weather-demo git:(main)
./gradlew clean
```

Instrumented Test Class

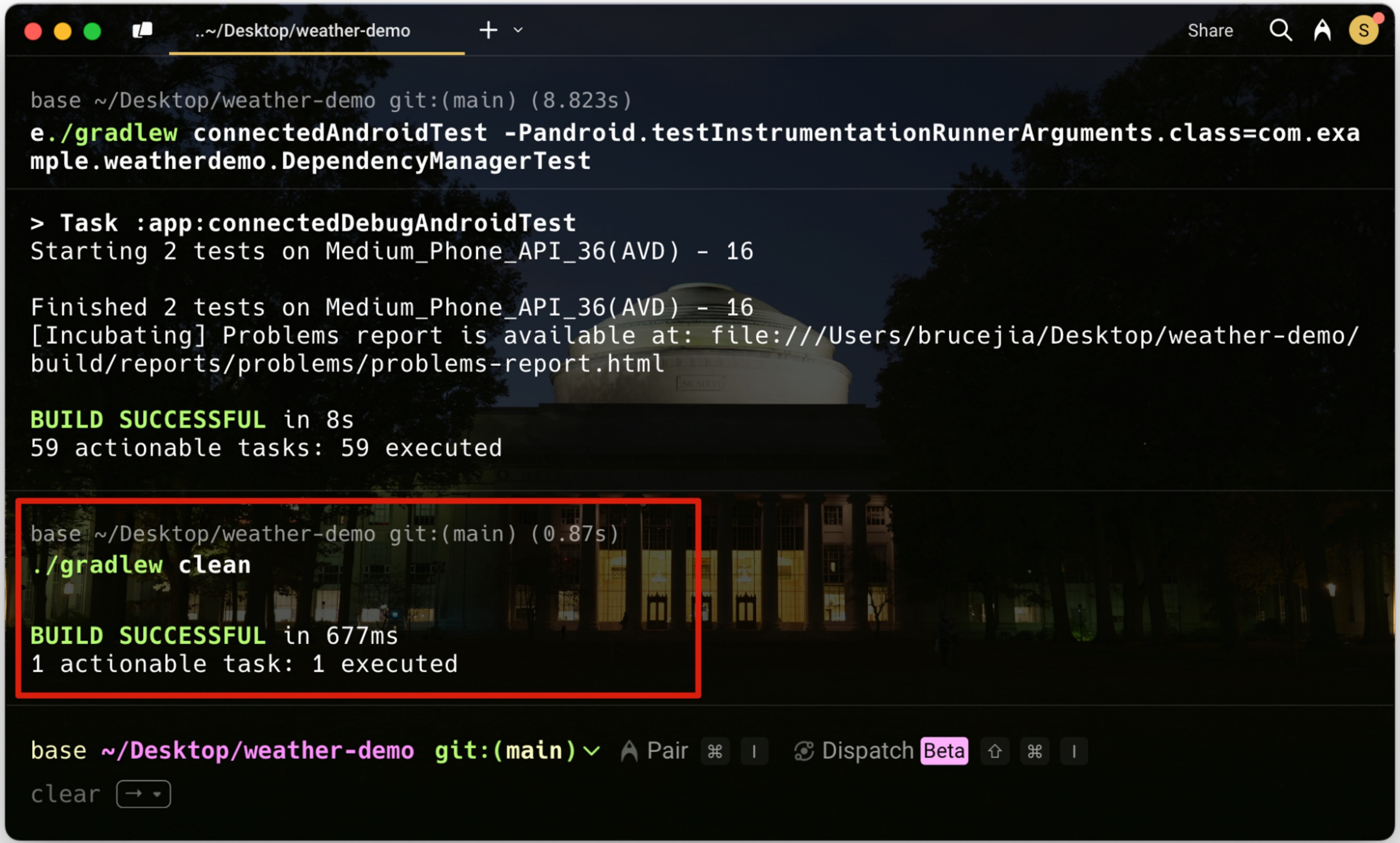


```
base ~/Desktop/weather-demo git:(main) (8.823s)
e./gradlew connectedAndroidTest -Pandroid.testInstrumentationRunnerArguments.class=com.exa
mple.weatherdemo.DependencyManagerTest

> Task :app:compileDebugJavaWithJavac
Note: /Users/brucejia/Desktop/weather-demo/app/src/main/java/com/example/weatherdemo/Weath
erDetailsActivity.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

> Task :app:connectedDebugAndroidTest
Starting 2 tests on Medium_Phone_API_36(AVD) - 16

Finished 2 tests on Medium_Phone_API_36(AVD) - 16
[Incubating] Problems report is available at: file:///Users/brucejia/Desktop/weather-demo/
build/reports/problems/problems-report.html

BUILD SUCCESSFUL in 8s
59 actionable tasks: 59 executed
```

Check for deprecated API usage in the code.

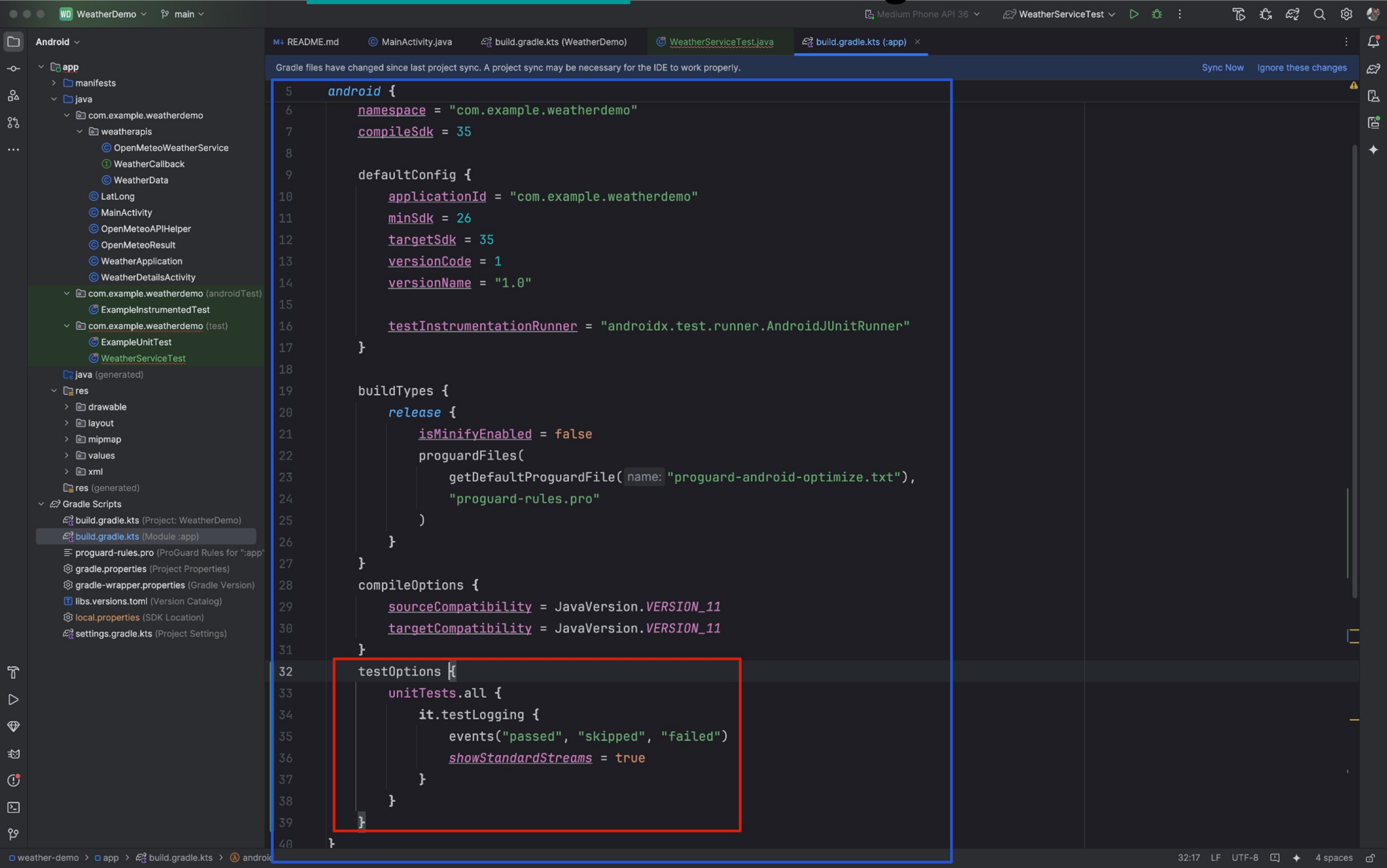
```
base ~/Desktop/weather-demo git:(main)
```

Instrumented Test Class



```
base ~/Desktop/weather-demo git:(main) (8.823s)
e./gradlew connectedAndroidTest -Pandroid.testInstrumentationRunnerArguments.class=com.exa
mple.weatherdemo.DependencyManagerTest

> Task :app:connectedDebugAndroidTest
Starting 2 tests on Medium_Phone_API_36(AVD) - 16

Finished 2 tests on Medium_Phone_API_36(AVD) - 16
[Incubating] Problems report is available at: file:///Users/brucejia/Desktop/weather-demo/
build/reports/problems/problems-report.html

BUILD SUCCESSFUL in 8s
59 actionable tasks: 59 executed

base ~/Desktop/weather-demo git:(main) (0.87s)
./gradlew clean

BUILD SUCCESSFUL in 677ms
1 actionable task: 1 executed

base ~/Desktop/weather-demo git:(main)
clear
```

Part 6 — Weather Demo Unit Testing



Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

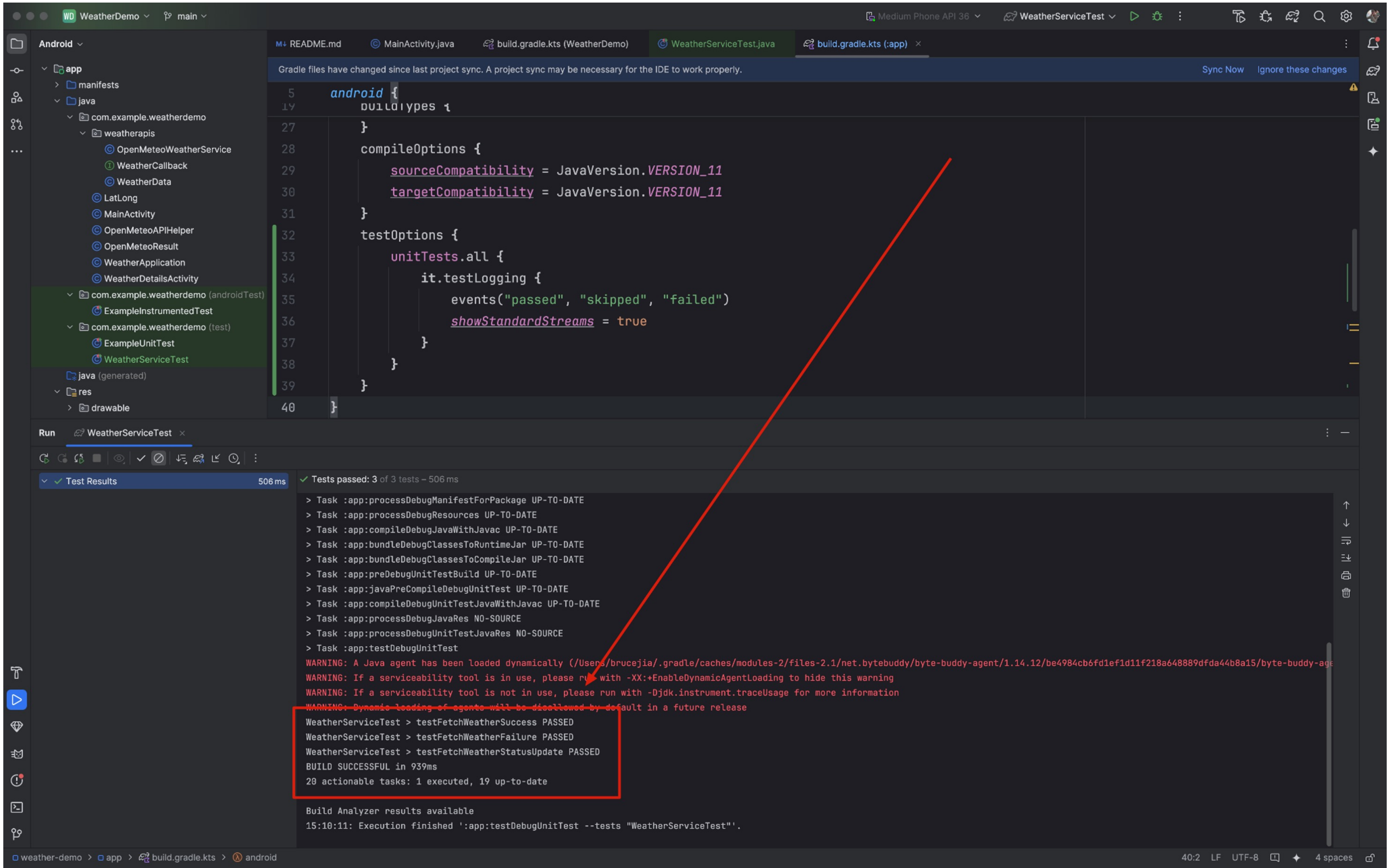
```
android {
    namespace = "com.example.weatherdemo"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.example.weatherdemo"
        minSdk = 26
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile( name: "proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
    testOptions {
        unitTests.all {
            it.testLogging {
                events("passed", "skipped", "failed")
                showStandardStreams = true
            }
        }
    }
}
```

Part 6 — Weather Demo Unit Testing

You can add your `testOptions {}` inside the `android {}`

```
android {
    …

    testOptions {
        unitTests.all {
            it.testLogging {
                events("passed", "skipped", "failed")
                showStandardStreams = true
            }
        }
    }
}
```

Part 6 — Weather Demo Unit Testing



Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

```
android {
    buildTypes {
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
    testOptions {
        unitTests.all {
            it.testLogging {
                events("passed", "skipped", "failed")
                showStandardStreams = true
            }
        }
    }
}
```

Tests passed: 3 of 3 tests – 506 ms

```
> Task :app:processDebugManifestForPackage UP-TO-DATE
> Task :app:processDebugResources UP-TO-DATE
> Task :app:compileDebugJavaWithJavac UP-TO-DATE
> Task :app:bundleDebugClassesToRuntimeJar UP-TO-DATE
> Task :app:bundleDebugClassesToCompileJar UP-TO-DATE
> Task :app:preDebugUnitTestBuild UP-TO-DATE
> Task :app:javaPreCompileDebugUnitTest UP-TO-DATE
> Task :app:compileDebugUnitTestJavaWithJavac UP-TO-DATE
> Task :app:processDebugJavaRes NO-SOURCE
> Task :app:processDebugUnitTestJavaRes NO-SOURCE
> Task :app:testDebugUnitTest
WARNING: A Java agent has been loaded dynamically (/Users/brucejia/.gradle/caches/modules-2/files-2.1/net.bytebuddy/byte-buddy-agent/1.14.12/be4984cb6fd1ef1d11f218a648889dfda44b8a15/byte-buddy-ag
WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a future release
WeatherServiceTest > testFetchWeatherSuccess PASSED
WeatherServiceTest > testFetchWeatherFailure PASSED
WeatherServiceTest > testFetchWeatherStatusUpdate PASSED
BUILD SUCCESSFUL in 939ms
20 actionable tasks: 1 executed, 19 up-to-date

Build Analyzer results available
15:10:11: Execution finished ':app:testDebugUnitTest --tests "WeatherServiceTest"'.
```

Part 7 — Takeaway

Use **Google Test** for **C++** codes

Use **JUnit with Gradle** for **Android Java/Kotlin** codes

Use **`Android CI`** with **GitHub Actions** for Android Testing

Setting **Rules** on GitHub for **code merging**

Use **`Mockito`** to replace any **network**, **file system**, or **async behavior with canned responses**

Use **`@RunWith(AndroidJUnit4.class)`** for **Android UI components**, **Android activity**, or **Android main thread**

Use `testOptions` for **testing event report**

Part 7 — Takeaway (Credits and Links)

Google Test for **C++** codes

Android-Unit-Testing/tree/main/GoogleTest/tests

Use **JUnit with Gradle** for **Android Java/Kotlin** codes

[1] Android-Unit-Testing

[2] weatherdemo/weatherapis/WeatherAPIServiceTest.java

Use `Mockito` to replace **any network**, **file system**, or **async behavior with canned responses**

weatherdemo/weatherapis/WeatherAPIServiceTest.java

Use `@RunWith(AndroidJUnit4.class)` for **Android UI components**, **Android activity**, or **Android main thread**

weatherdemo/DependencyManagerTest.java

Use `testOptions` for **testing event report**

weather-demo/blob/main/app/build.gradle.kts

Thank you very much for your attention!