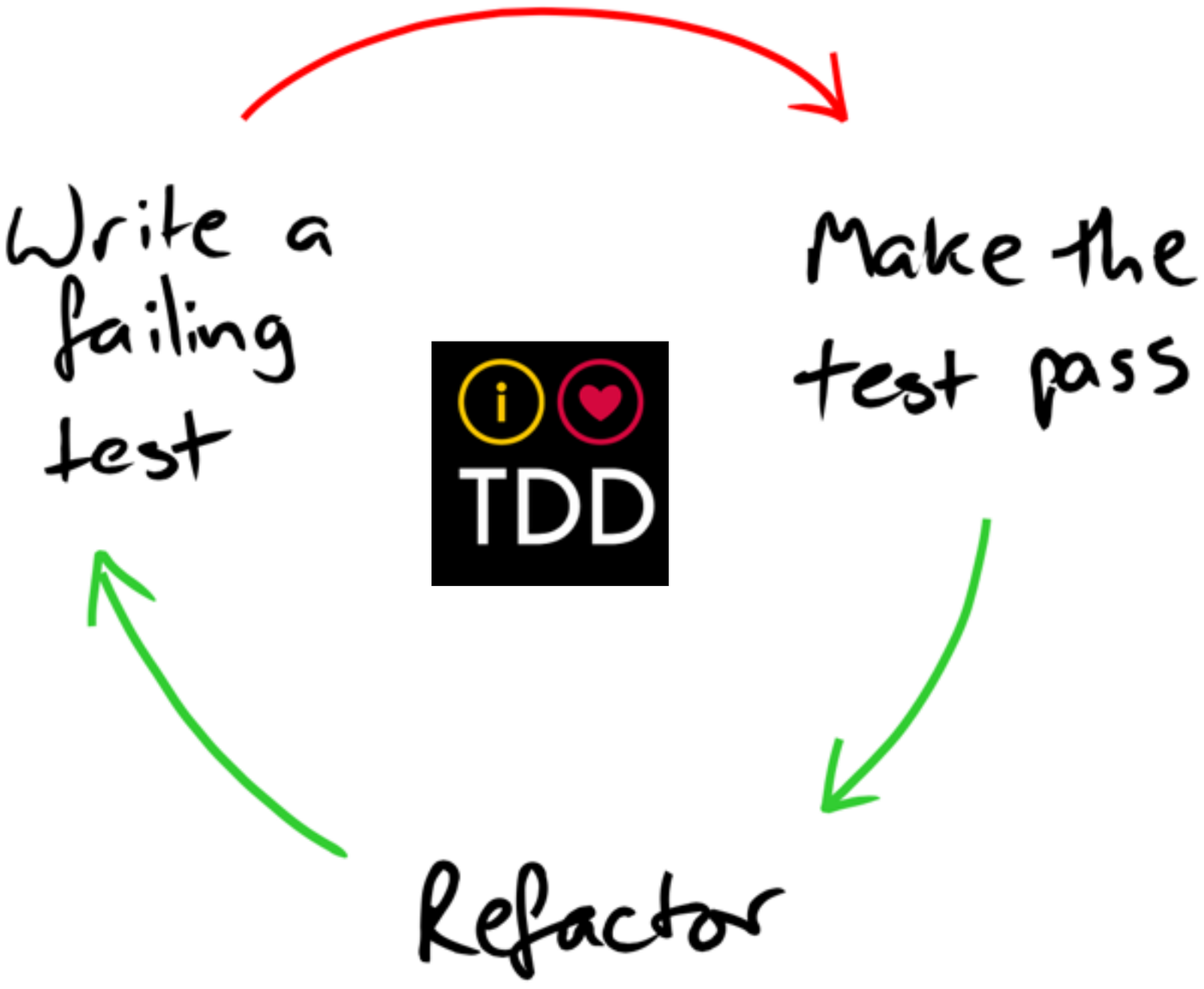
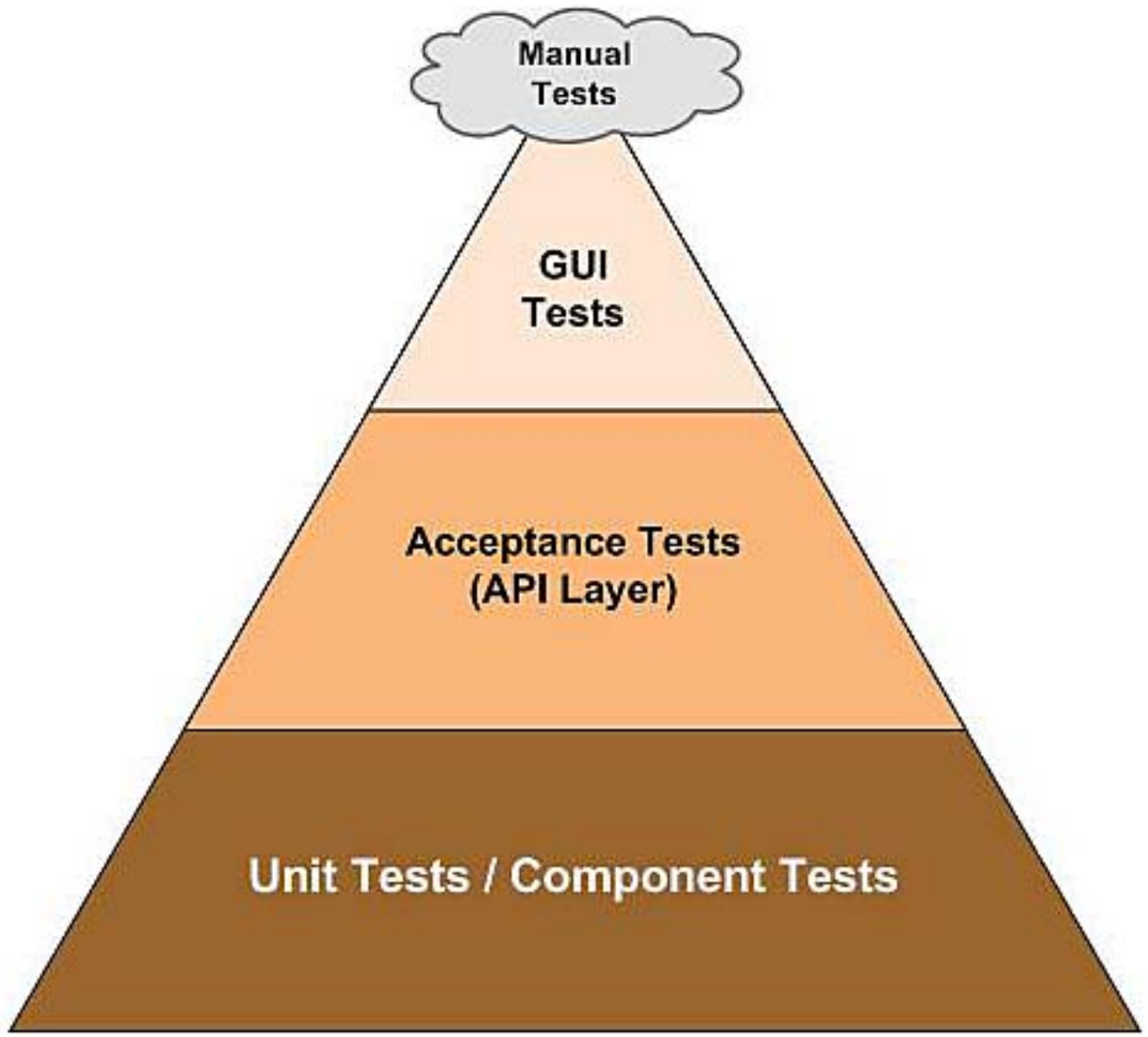
WORKSHOP :: Espresso testing



Android Testing



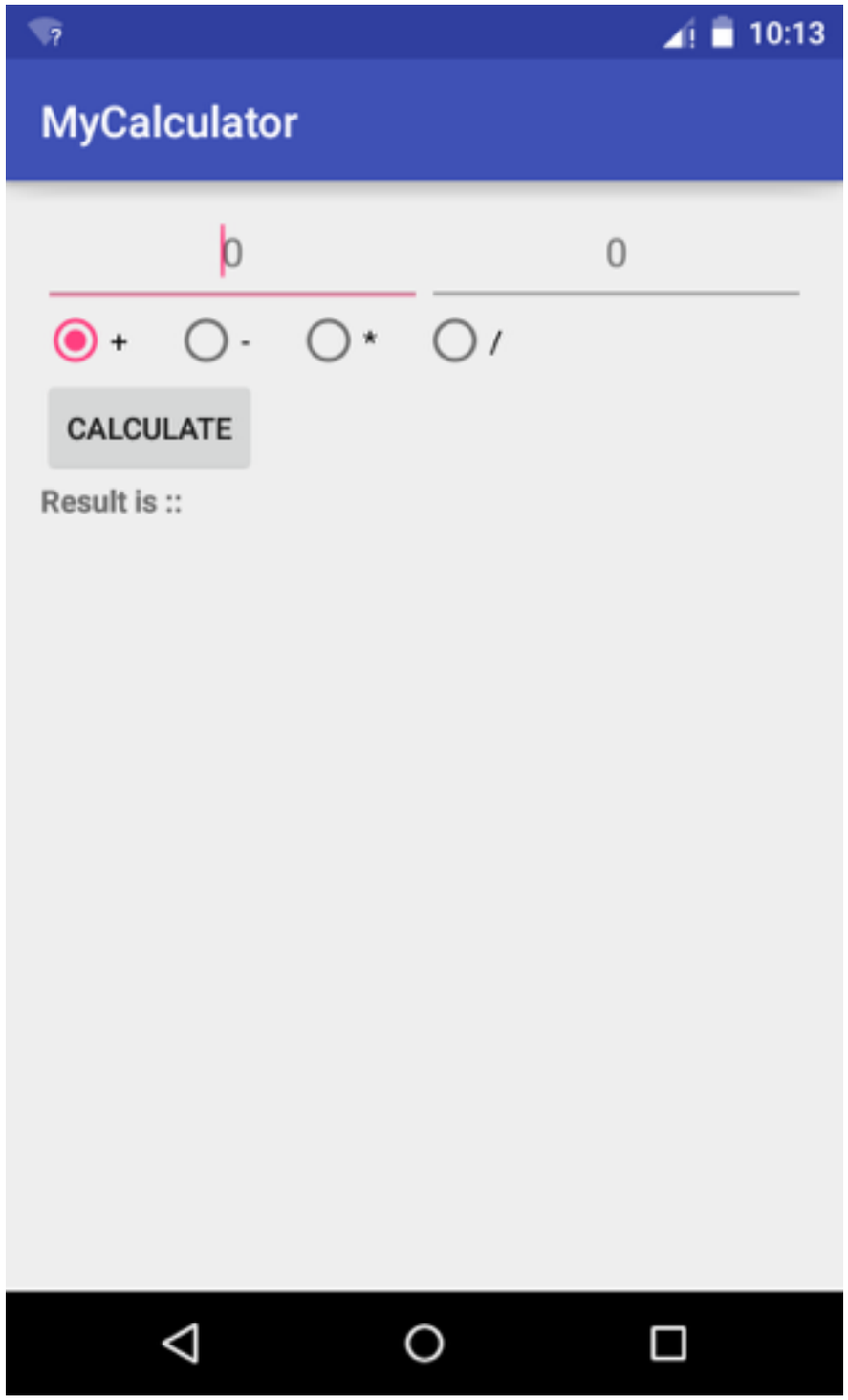
Monkey

Espresso

Android unit test

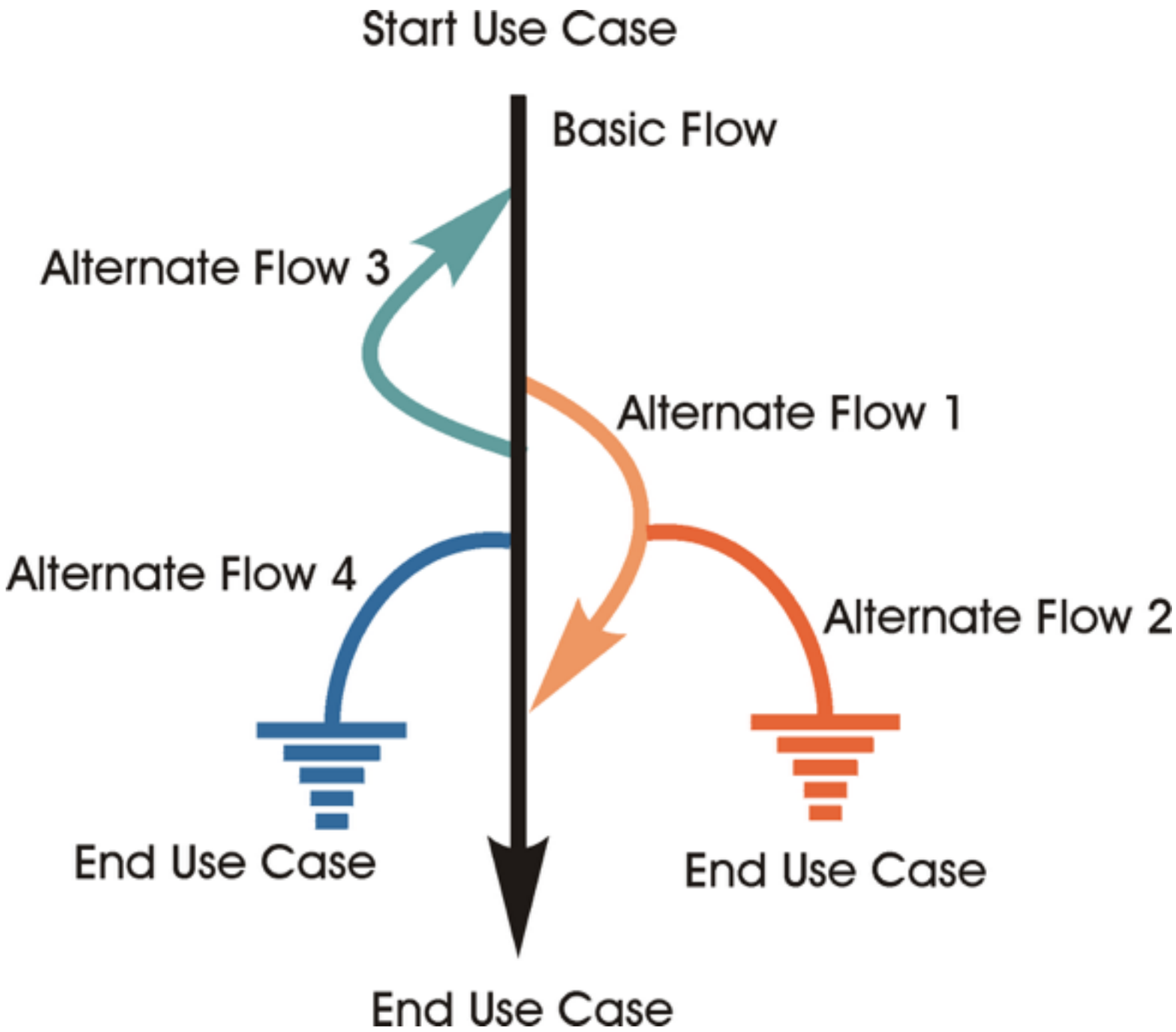
jUnit

My calculator



How to test ?

Test cases ?


Start Use Case
Basic Flow
Alternate Flow 3
Alternate Flow 1
Alternate Flow 4
Alternate Flow 2
End Use Case
End Use Case
End Use Case

UI TESTING FOR ANDROID

espresso

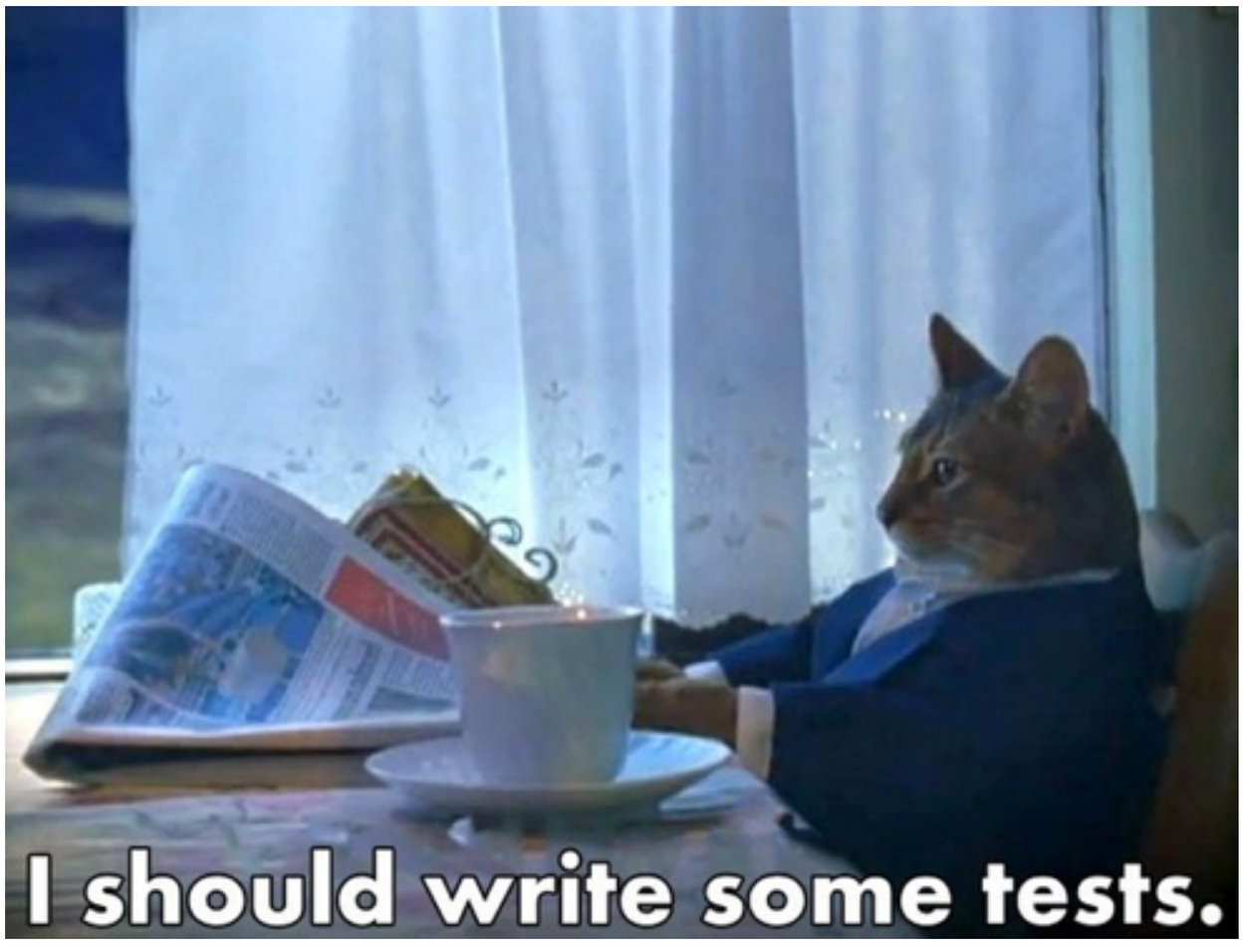
https://google.github.io/android-testing-support-library/

Espresso ?

- A funny little Android UI test APIs
- Created by Google
- Easy APIs

Why Espresso ?

“Developer Developer Developer”



Why Espresso ?

- Developer need ...
  - easy
  - reliable
  - durable

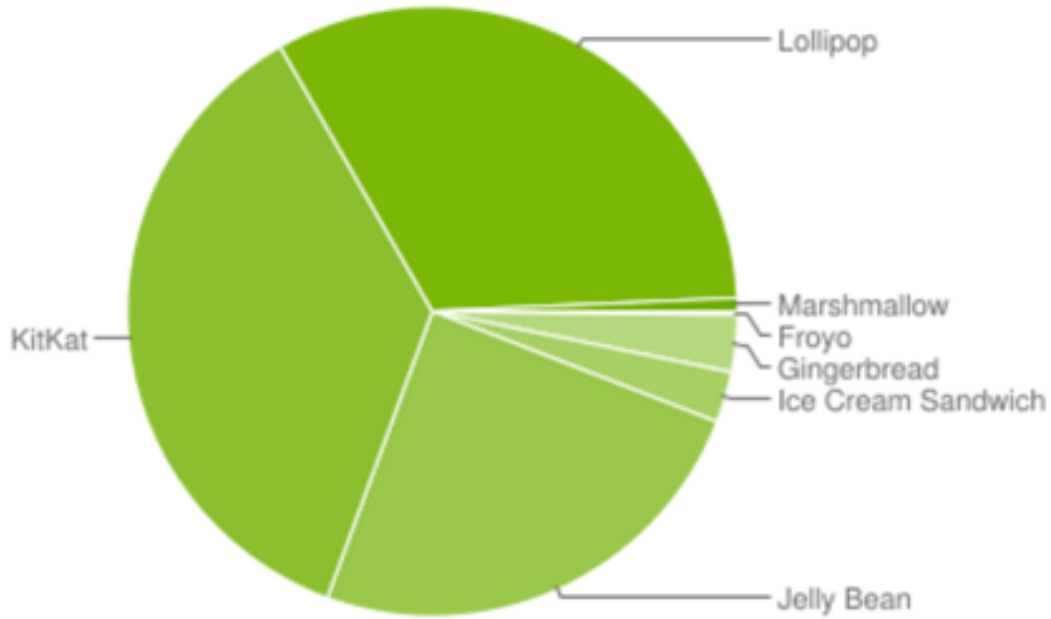
Why Espresso ?

- On all API Levels

Codename	API
Froyo	8
Gingerbread	10
Ice Cream Sandwich	15
Jelly Bean	16,17,18
KitKat	19
Lollipop	21

Platform version

Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	3.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.7%
4.1.x	Jelly Bean	16	9.0%
4.2.x		17	12.2%
4.3		18	3.5%
4.4	KitKat	19	36.1%
5.0	Lollipop	21	16.9%
5.1		22	15.7%
6.0	Marshmallow	23	0.7%

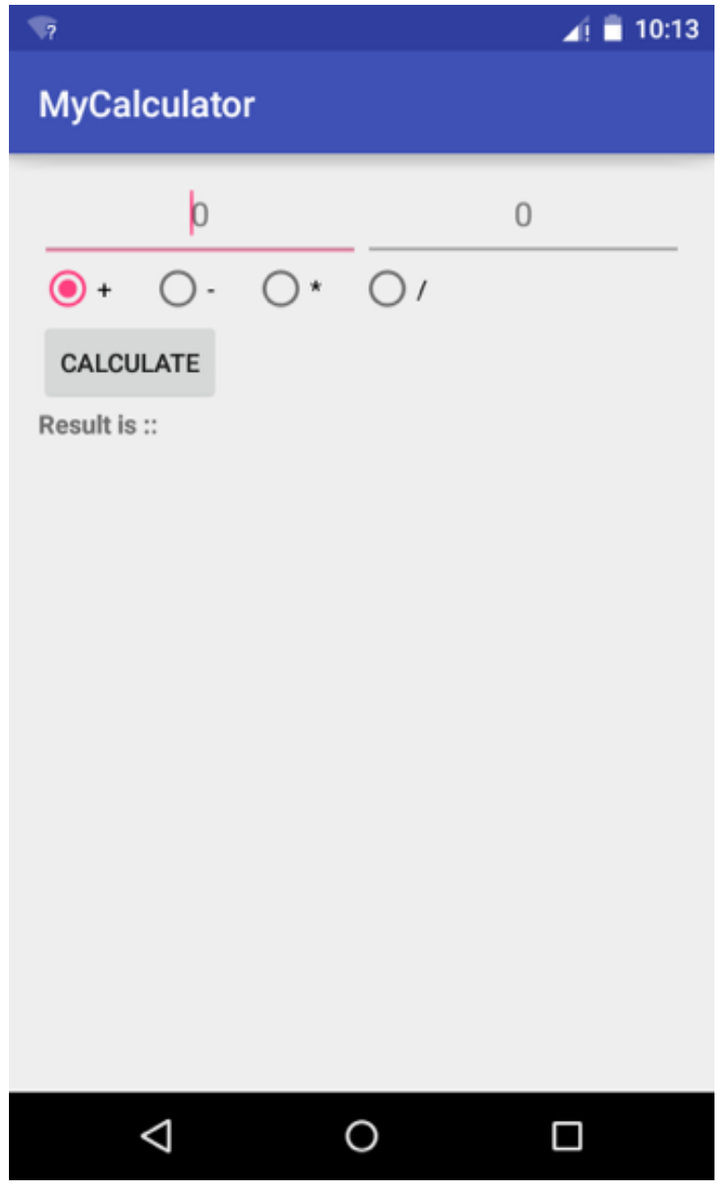


http://developer.android.com/about/dashboards/index.html#Platform

Easy :: API for testing

What would a device user do ?

- Find a view
- Do something with it
- Check some state



Find view & Do

```
onView(withId(R.id.greet_button))
        .perform(click());
```

Check something

```
onView(withText("Hello Steve!"))
        .check(matches(isDisplayed()));
```

Easy :: Framework APIs

onView(Matcher<View>)

perform(ViewAction)

check(ViewAssertion)

Easy :: Framework APIs

onView(**Matcher<View>**)

- withId
- withText
- withContentDescription
- isDisplay
- hasFocus
- hasSibling
- custom

perform(ViewAction)

check(ViewAssertion)

Easy :: Framework APIs

onView(Matcher<View>)

perform(**ViewAction**)

- click
- longClick
- doubleClick
- typeText
- scrollTo
- custom

check(ViewAssertion)

Easy :: Framework APIs

onView(Matcher<View>)

perform(ViewAction)

check(**ViewAssertion**)

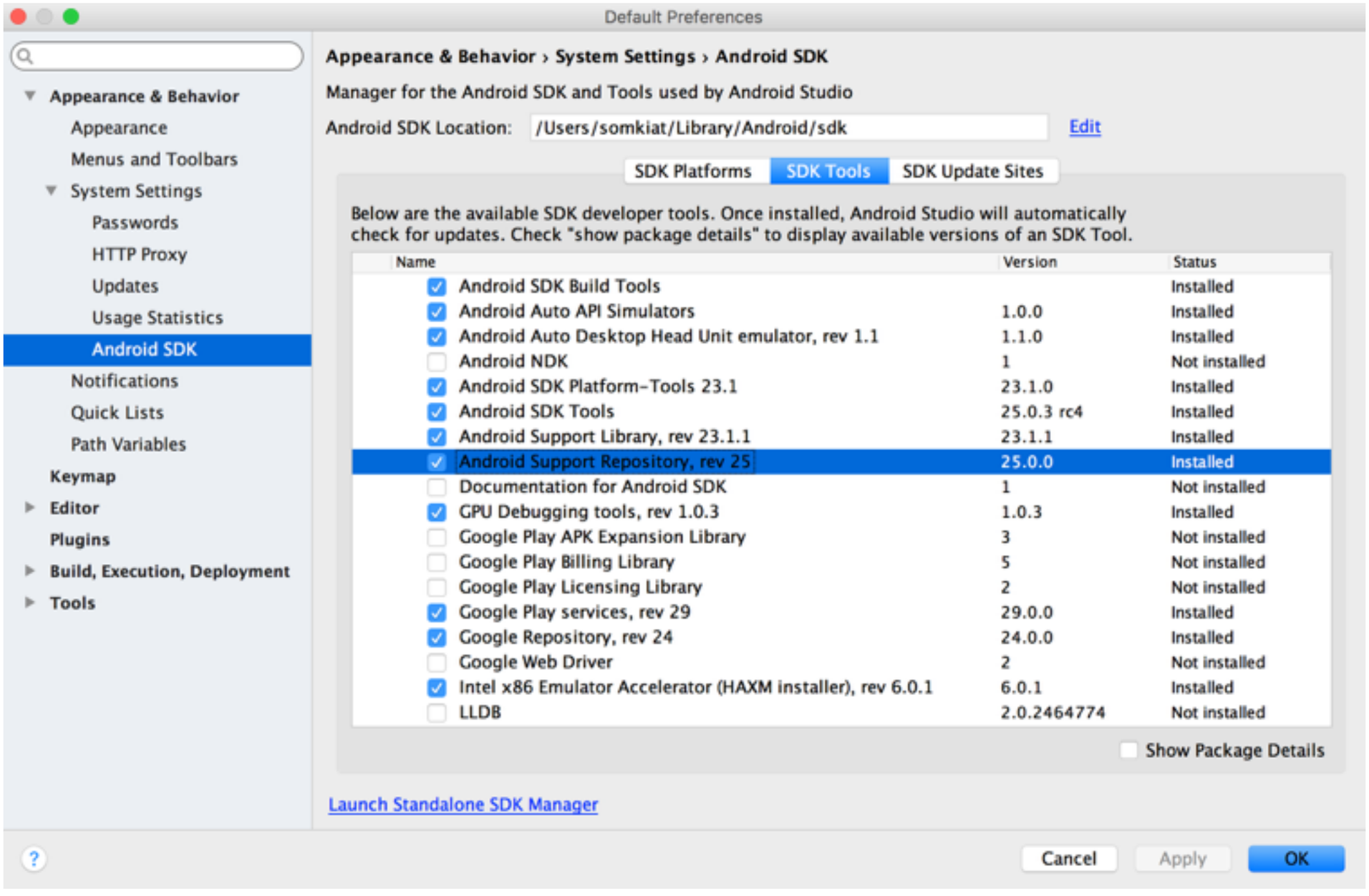
- matches
- doesNotExist
- custom

Easy :: To install

How to install espresso ?

Install Android suport repository

Tools -> Android -> SDK Manager



Config app/build.gradle

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile "com.android.support:appcompat-v7:$rootProject.supportLibraryVersion"

    // Android Testing Support Library's runner and rules
    androidTestCompile "com.android.support.test:runner:$rootProject.ext.runnerVersion"
    androidTestCompile "com.android.support.test:rules:$rootProject.ext.runnerVersion"

    // Espresso UI Testing dependencies.
    androidTestCompile "com.android.support.test.espresso:espresso-core:$rootProject.ext.espressoVersion"
    androidTestCompile "com.android.support.test.espresso:espresso-contrib:$rootProject.ext.espressoVersion"
    androidTestCompile "com.android.support.test.espresso:espresso-intents:$rootProject.ext.espressoVersion"
}

configurations.all {
    resolutionStrategy.force "com.android.support:support-annotations:$rootProject.supportLibraryVersion"
}

configurations.compile.dependencies.each { compileDependency ->
    println "Excluding compile dependency: ${compileDependency.getName()}"
    configurations.androidTestCompile.dependencies.each { androidTestCompileDependency ->
        configurations.androidTestCompile.exclude module: "${compileDependency.getName()}"
    }
}
```

Config build.gradle

```
ext {
    // Sdk and tools
    minSdkVersion = 10
    targetSdkVersion = 22
    compileSdkVersion = 23
    buildToolsVersion = '23.0.2'

    // App dependencies
    supportLibraryVersion = '23.1.1'
    runnerVersion = '0.4.1'
    rulesVersion = '0.4.1'
    espressoVersion = '2.2.1'
}
```

Config app/build.gradle

```
android {
    compileSdkVersion rootProject.ext.compileSdkVersion
    buildToolsVersion rootProject.ext.buildToolsVersion

    defaultConfig {
        applicationId "up1.mycalculator"
        minSdkVersion rootProject.ext.minSdkVersion
        targetSdkVersion rootProject.ext.targetSdkVersion
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner 'android.support.test.runner.AndroidJUnitRunner'

    }
```

Create first test !!

Default in **src/androidTest/java/<your package>**

- MainActivityTest.java

Step 1 :: Run with

```java
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;

@RunWith(AndroidJUnit4.class)
public class MainActivityTest {

    @Rule
    public ActivityTestRule<MainActivity> mMainActivityTestRule =
            new ActivityTestRule<>(MainActivity.class);


    @Test
    public void first_test_case() {

    }


}
```

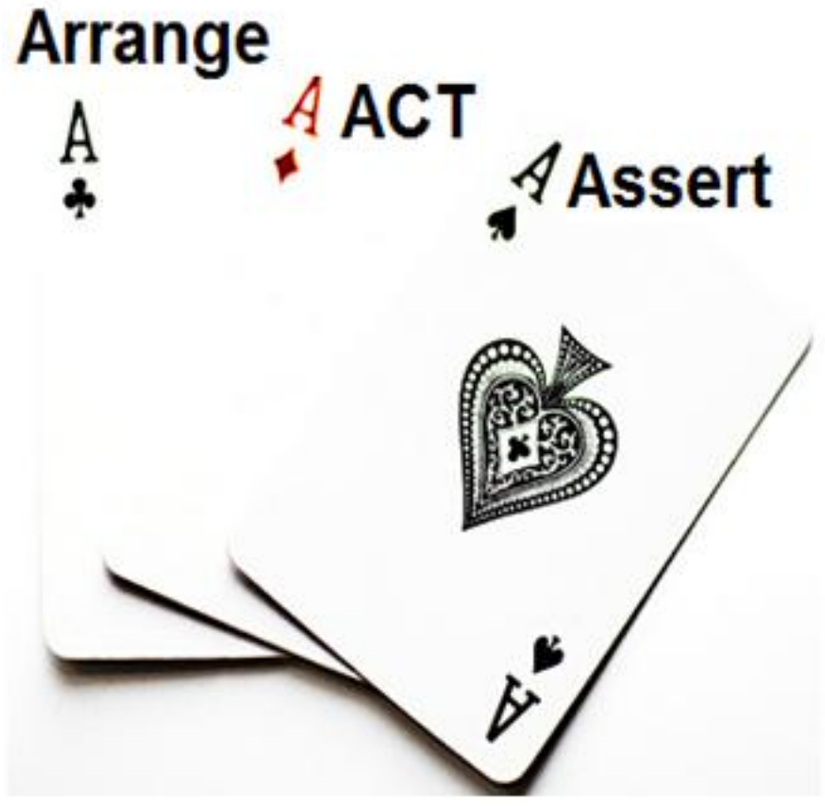
Step 2 :: Add rule to start activity

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

@RunWith(AndroidJUnit4.class)
public class MainActivityTest {

    @Rule
    public ActivityTestRule<MainActivity> mMainActivityTestRule =
            new ActivityTestRule<>(MainActivity.class);

    @Test
    public void first_test_case() {

    }


}
```

Step 3 :: Create a first test case

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

@RunWith(AndroidJUnit4.class)
public class MainActivityTest {

    @Rule
    public ActivityTestRule<MainActivity> mMainActivityTestRule =
            new ActivityTestRule<>(MainActivity.class);


    @Test
    public void first_test_case() {

    }


}
```

Structure of good test case



AAA

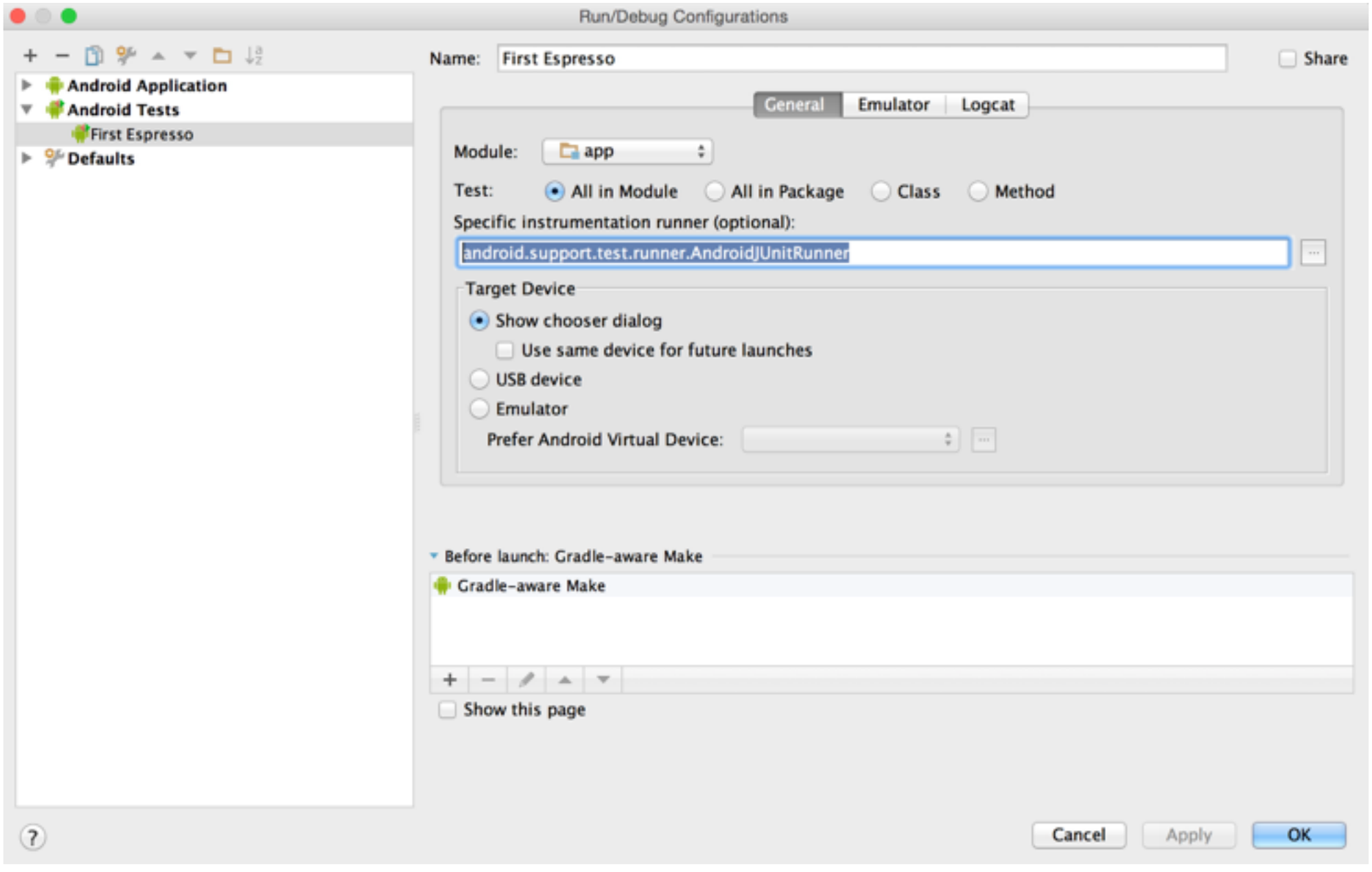
```
@Test
public void first_test_case() {

    //Arrange
    onView(withId(R.id.firstNumber)).perform(typeText("5"));
    onView(withId(R.id.secondNumber)).perform(typeText("4"));

    //Act
    onView(withId(R.id.calculateButton)).perform(click());

    //Assert
    onView(withId(R.id.result)).check(matches(withText("9")));

}
```

How to run test ?

How to run test ?

Run -> Edit configuration -> Add android tests

Runner = **android.support.test.runner.AndroidJUnitRunner**



How to run test ?

On command line or terminal

$./gradlew :App:connectedAndroidTest

See result

Test Summary

2	0	4.578s
tests	failures	duration

100%
successful

Packages Classes

Package	Tests	Failures	Duration	Success rate
up1.mycalculator	2	0	4.578s	100%

Run all devices

Class up1.mycalculator.MainActivityTest

all > up1.mycalculator > MainActivityTest

2	0	4.578s
tests	failures	duration

100%
successful

Tests **Devices**

Devices	Tests	Failures	Duration	Success rate
Google Nexus 4 - 5.0.0 - API 21 - 768x1280 - 5.0	1	0	2.173s	100%
SM-G360HU - 4.4.4	1	0	2.405s	100%

Try by yourself

Many data test ?

Use Parameterized

```java
@RunWith(Parameterized.class)
public class MainActivityWithParamtersTest {

    private final String mFirst;
    private final String mSecond;
    private final String mOperator;
    private final String mResult;
    @Rule
    public ActivityTestRule<MainActivity> mMainActivityTestRule =
            new ActivityTestRule<>(MainActivity.class);

    @Parameterized.Parameters
    public static Iterable<Object[]> setupData() {
        return Arrays.asList(
                new Object[][]{
                        {"5", "4", "+", "9"},
                        {"5", "1", "+", "6"}
                }
        );
    }
```

Setup data

```
@RunWith(Parameterized.class)
public class MainActivityWithParamtersTest {

    private final String mFirst;
    private final String mSecond;
    private final String mOperator;
    private final String mResult;
    @Rule
    public ActivityTestRule<MainActivity> mMainActivityTestRule =
            new ActivityTestRule<>(MainActivity.class);

    @Parameterized.Parameters
    public static Iterable<Object[]> setupData() {
        return Arrays.asList(
                new Object[][]{
                        {"5", "4", "+", "9"},
                        {"5", "1", "+", "6"}
                }
        );
    }
```

Create constructer

```
public MainActivityWithParamtersTest(String first, String second,
                                     String operator, String result) {
    mFirst = first;
    mSecond = second;
    mOperator = operator;
    mResult = result;
}
```

```
@Test
public void first_test_case() {

    //Arrange
    onView(withId(R.id.firstNumber)).perform(typeText(this.mFirst));
    onView(withId(R.id.secondNumber)).perform(typeText(this.mSecond));

    //Act
    onView(withId(R.id.calculateButton)).perform(click());

    //Assert
    onView(withId(R.id.result)).check(matches(withText(this.mResult)));

}
```

Create test case with data

```
public MainActivityWithParamtersTest(String first, String second,
                                     String operator, String result) {
    mFirst = first;
    mSecond = second;
    mOperator = operator;
    mResult = result;
}
```

```
@Test
public void first_test_case() {

    //Arrange
    onView(withId(R.id.firstNumber)).perform(typeText(this.mFirst));
    onView(withId(R.id.secondNumber)).perform(typeText(this.mSecond));

    //Act
    onView(withId(R.id.calculateButton)).perform(click());

    //Assert
    onView(withId(R.id.result)).check(matches(withText(this.mResult)));

}
```

Try by yourself