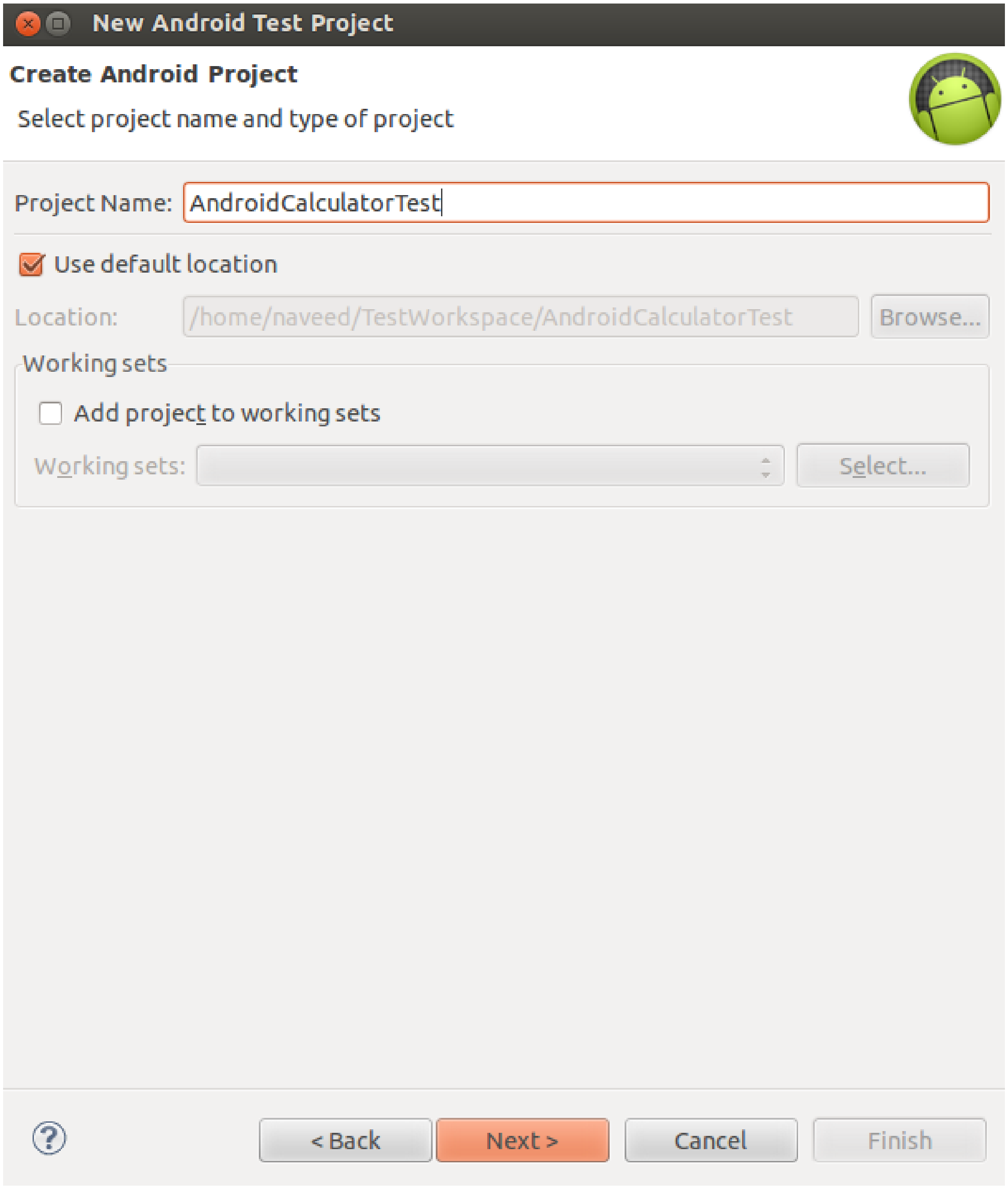
Test Android Application with Robotium:

1. Create Test Project

To test an Android application using Robotium, we need to create a test project within the package *(com.calculator)* of Simple Calculator Project Source Code.

Put project name



New Android Test Project

Create Android Project

Select project name and type of project

Project Name: AndroidCalculatorTest

☑ Use default location

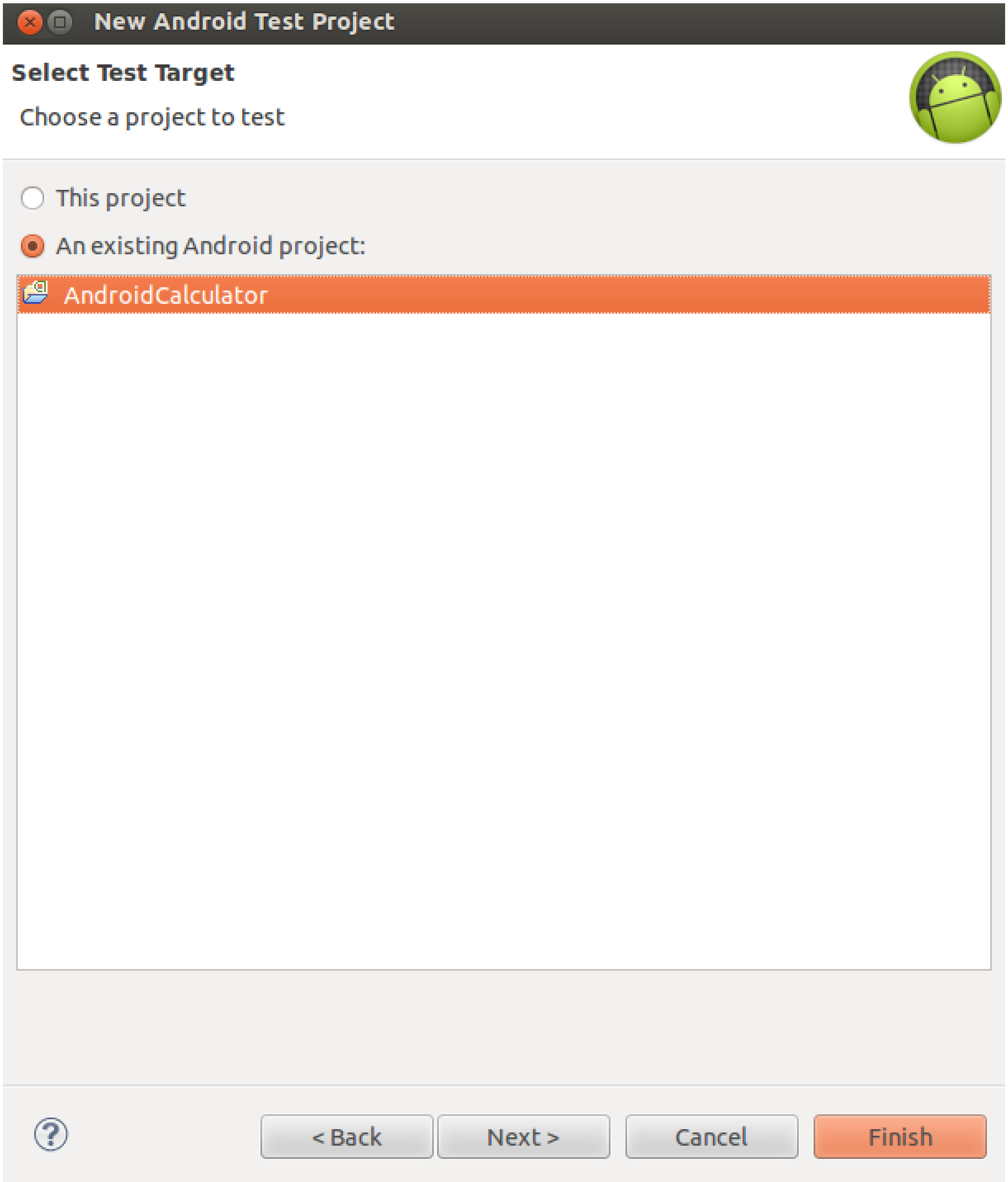
Location: /home/naveed/TestWorkspace/AndroidCalculatorTest

Working sets

□ Add project to working sets

Working sets:

Select test target or AUT(application under test)


New Android Test Project

Select Test Target

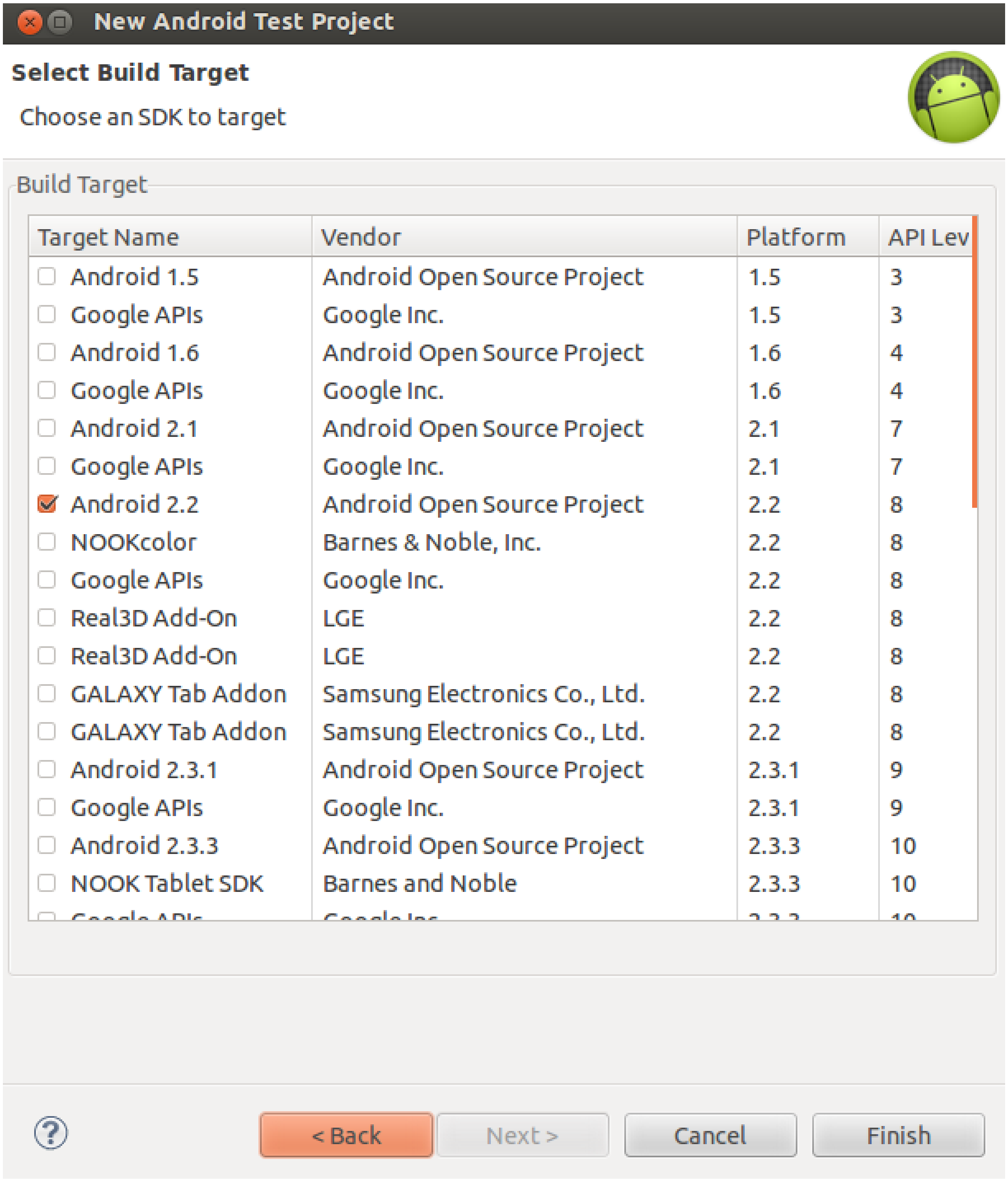
Choose a project to test

○ This project

● An existing Android project:

AndroidCalculator

Select Min SDK version



Select Build Target

Choose an SDK to target

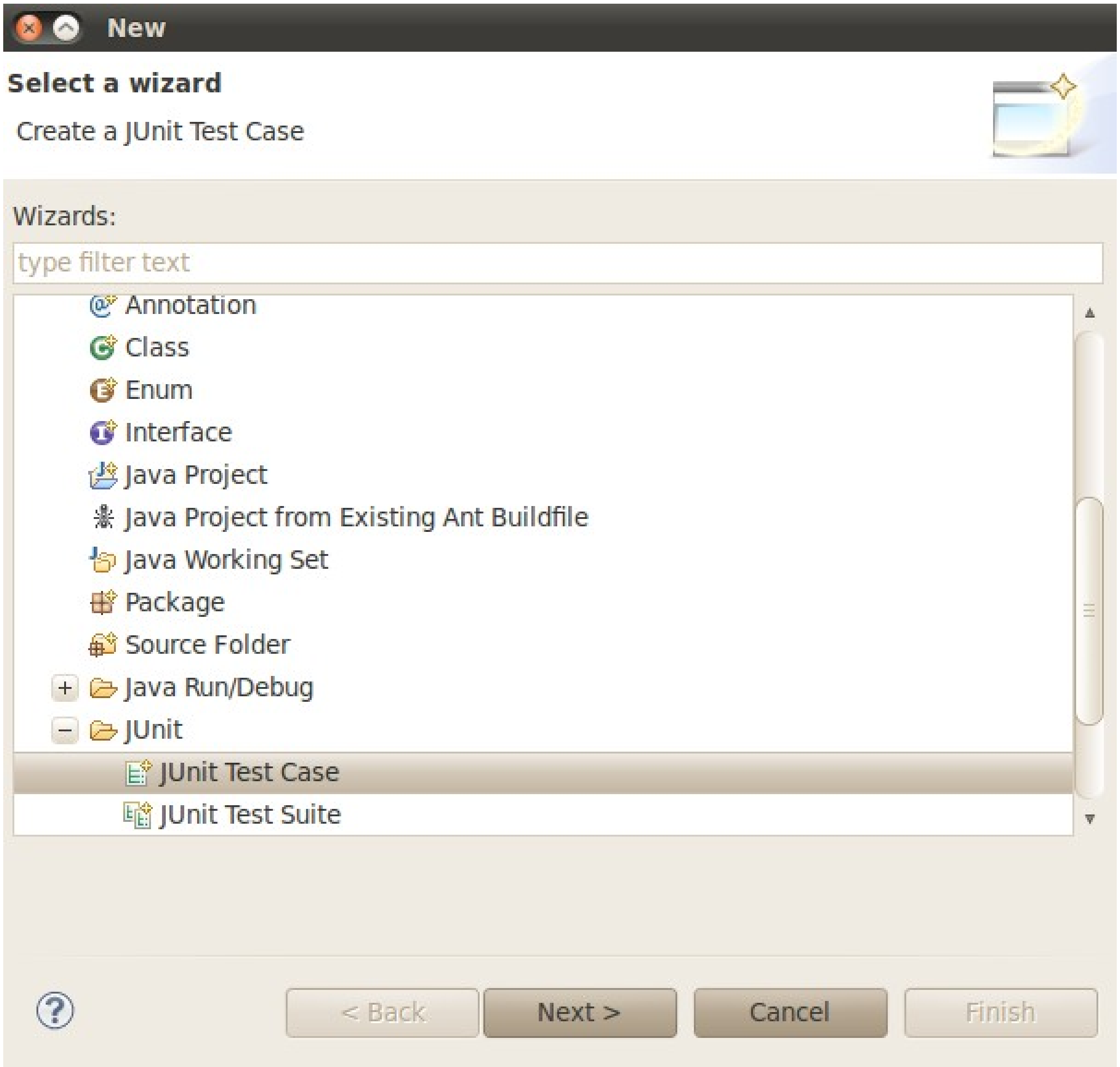
Build Target

Target Name	Vendor	Platform	API Lev
☐ Android 1.5	Android Open Source Project	1.5	3
☐ Google APIs	Google Inc.	1.5	3
☐ Android 1.6	Android Open Source Project	1.6	4
☐ Google APIs	Google Inc.	1.6	4
☐ Android 2.1	Android Open Source Project	2.1	7
☐ Google APIs	Google Inc.	2.1	7
☑ Android 2.2	Android Open Source Project	2.2	8
☐ NOOKcolor	Barnes & Noble, Inc.	2.2	8
☐ Google APIs	Google Inc.	2.2	8
☐ Real3D Add-On	LGE	2.2	8
☐ Real3D Add-On	LGE	2.2	8
☐ GALAXY Tab Addon	Samsung Electronics Co., Ltd.	2.2	8
☐ GALAXY Tab Addon	Samsung Electronics Co., Ltd.	2.2	8
☐ Android 2.3.1	Android Open Source Project	2.3.1	9
☐ Google APIs	Google Inc.	2.3.1	9
☐ Android 2.3.3	Android Open Source Project	2.3.3	10
☐ NOOK Tablet SDK	Barnes and Noble	2.3.3	10
☐ Google APIs	Google Inc.	2.3.3	10

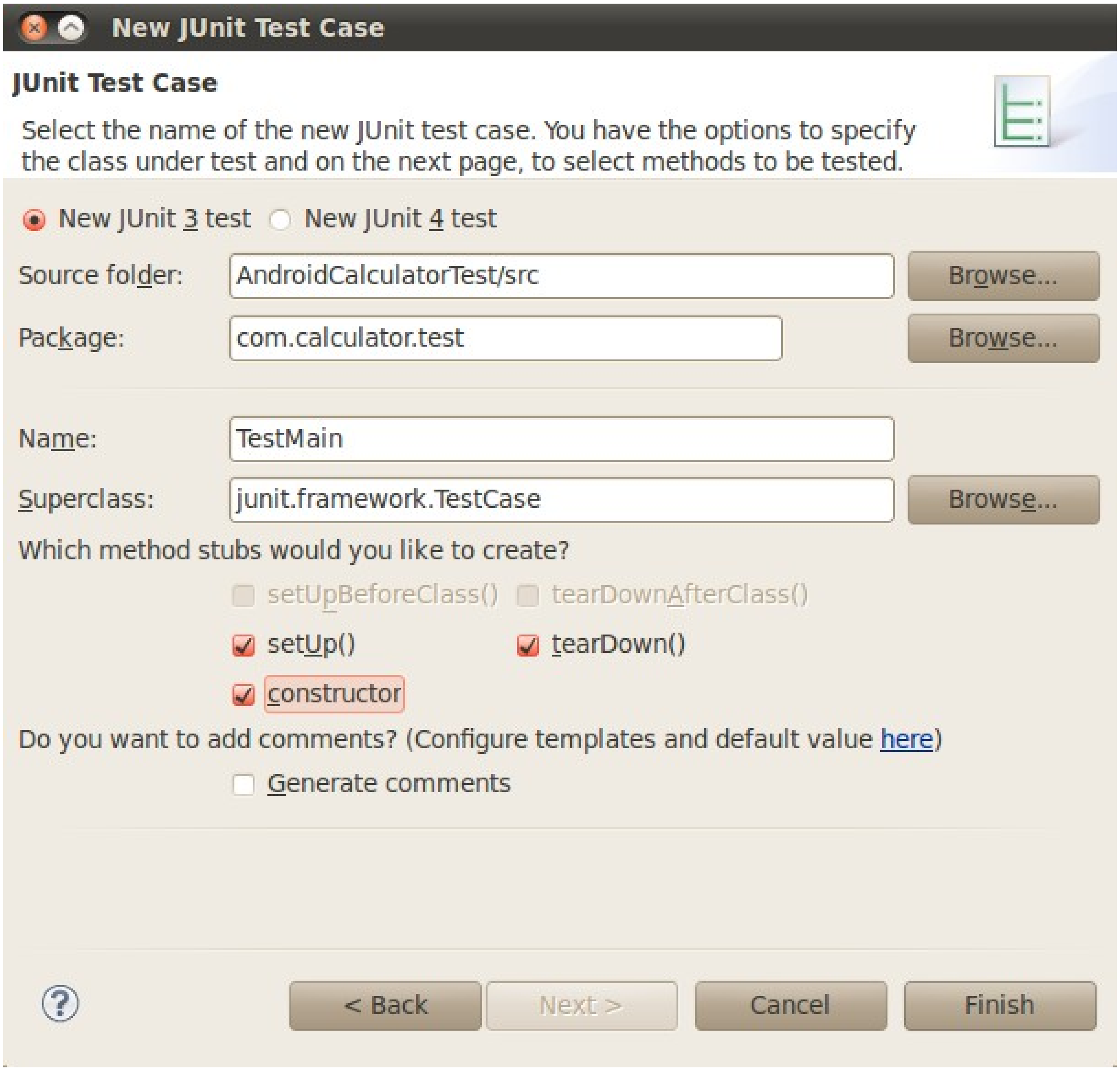
We will move on to design our logic to test AndroidCaculator. We need to create test case class where we will write code to test AndroidCalculator's main class (Main.java).

2. Create Test Case

In test project from project explorer window right click on *com.calculator.test* select *New* then *others*. On *New* window expand *Java* and then expand *Junit* category and select *Junit Test Case* and click on *Next*.



On *New Junit Test Case* screen, most of the options will be automatically filled as we have already created test project (AndroidCalculatorTest) with project (AndroidCalculator). We need to enter the Name of Test case, which I will enter TestMain, as I am going to test (main.java) of AndroidCalculator project. On next section check Setup(), tearDown() & Constructor options and click on Finish.



JUnit Test Case

Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.

● New JUnit 3 test ○ New JUnit 4 test

Source folder: AndroidCalculatorTest/src

Package: com.calculator.test

Name: TestMain

Superclass: junit.framework.TestCase

Which method stubs would you like to create?

□ setUpBeforeClass() □ tearDownAfterClass()

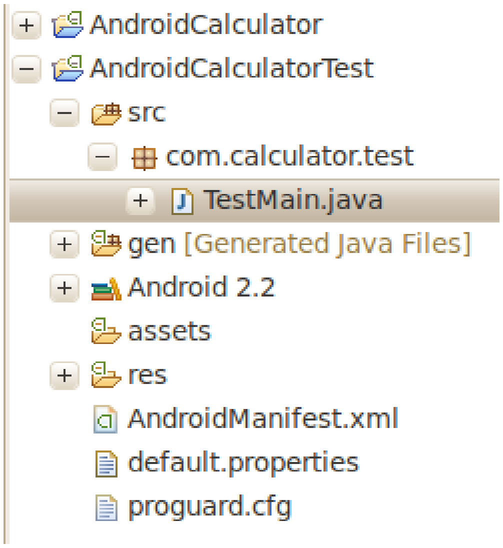
☑ setUp() ☑ tearDown()

☑ constructor

Do you want to add comments? (Configure templates and default value here)

□ Generate comments

A new test case by the name of TestMain will be created into com.calculator.test package of my test project (AndroidCaculatorTest)

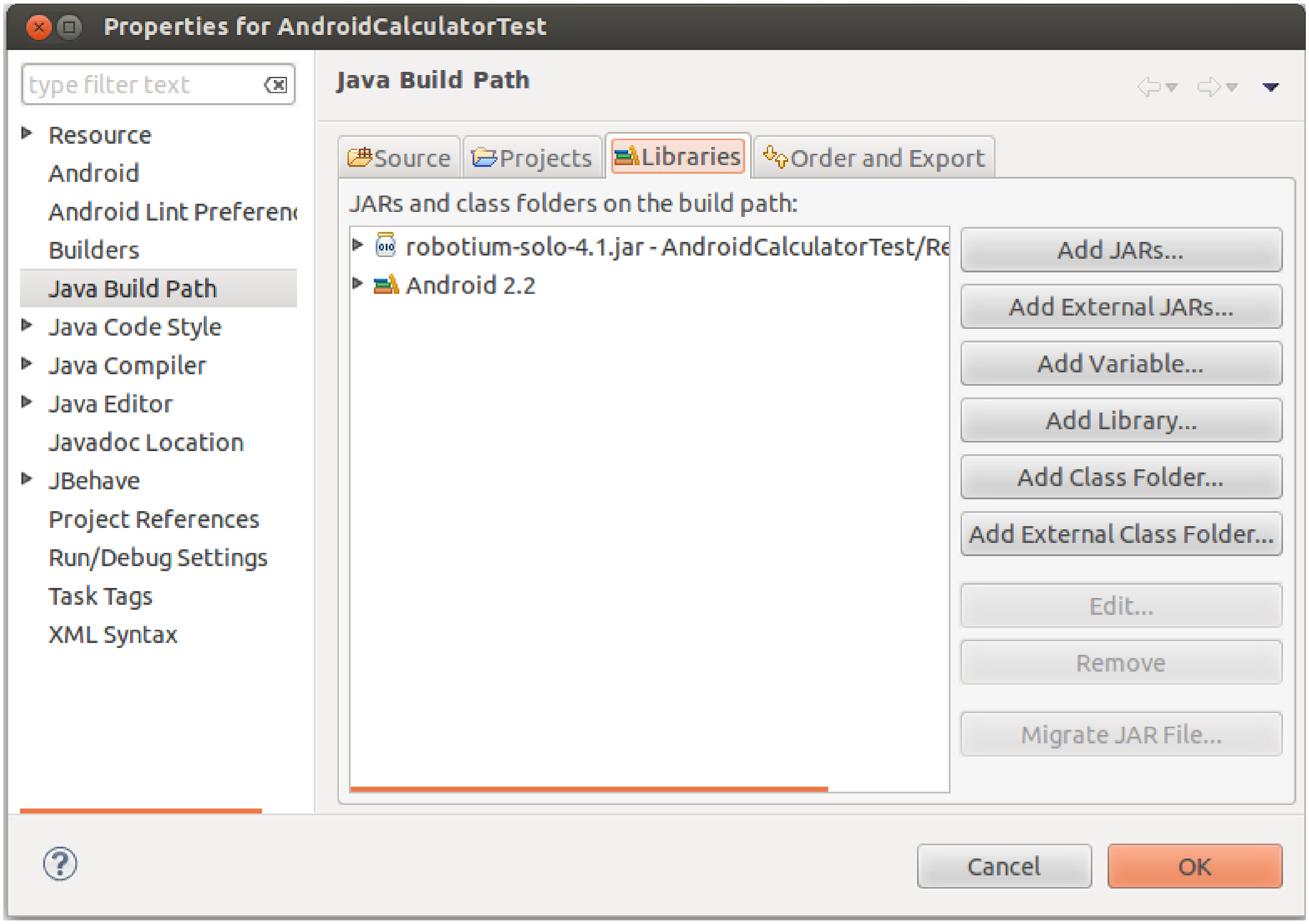


```
+ AndroidCalculator
− AndroidCalculatorTest
    − src
        − com.calculator.test
            + TestMain.java
    + gen [Generated Java Files]
    + Android 2.2
      assets
    + res
      AndroidManifest.xml
      default.properties
      proguard.cfg
```

3. Add Robotium jar

We need to reference the Robotium jar to our project.

Right click on project select *Build Path*, and then click on *Configure Build Path* option. On *Properties* window click on *Libraries* tab and add *Robotium latest* jar into project. we can download Robotium jar from http://code.google.com/p/robotium/downloads/list



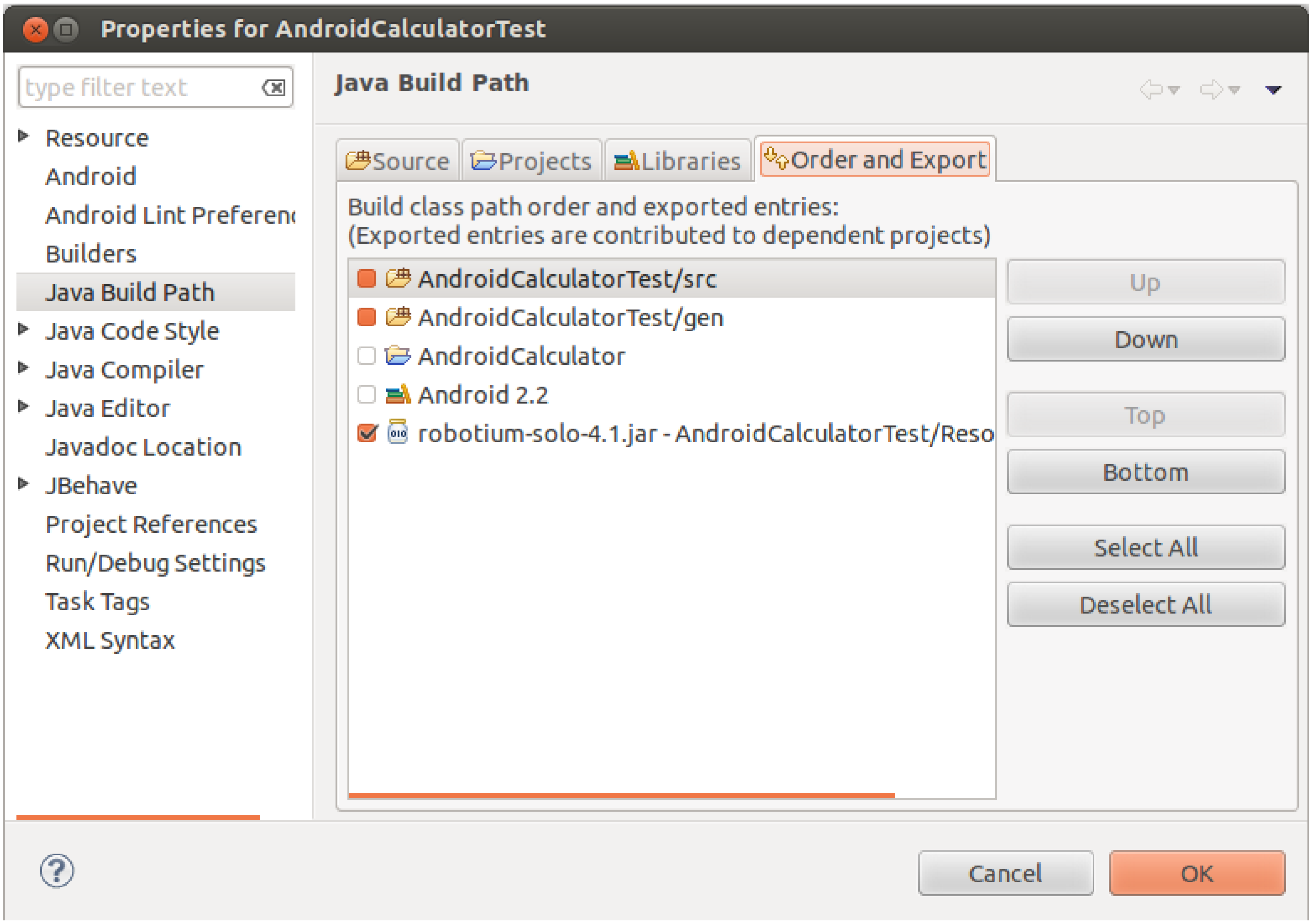
Properties for AndroidCalculatorTest

Java Build Path

JARs and class folders on the build path:

- robotium-solo-4.1.jar - AndroidCalculatorTest/Re
- Android 2.2

Note: In the latest Android SDK versions(17 or above) a **java.lang.NoClassDefFoundError: com.jayway.android.robotium.solo.Solo** error is shown if the Robotium jar is not exported . To fix the issue, after adding the Robotium jar go to the "Order & Export" tab and click the check-box besides the Robotium Jar and then click "OK". Please see the screenshot below.



Properties for AndroidCalculatorTest

Java Build Path

Build class path order and exported entries:
(Exported entries are contributed to dependent projects)

- ■ AndroidCalculatorTest/src
- ■ AndroidCalculatorTest/gen
- □ AndroidCalculator
- □ Android 2.2
- ☑ robotium-solo-4.1.jar - AndroidCalculatorTest/Reso

4. Write Test Case code

In our create test case we will access the contents of AndroidCalculator and do followings,

1. Call/Access first & second input controls (*EditFields*)
2. Enter values of our own choice
3. Access & Click on Multiply button
4. Put assert to verify their multiplication result into result field. And add

following code into *TestMain.java* class and save it.

```
package com.calculator.test;

import android.test.ActivityInstrumentationTestCase2;
import android.widget.EditText;
import android.widget.TextView;
import com.calculator.Main;
import com.calculator.R;
import com.jayway.android.robotium.solo.Solo;

public class TestMain extends ActivityInstrumentationTestCase2<Main> {
        private Solo solo;

        public TestMain() {
                super(Main.class);
        }

        @Override
        protected void setUp() throws Exception {
                super.setUp();
                solo = new Solo(getInstrumentation(), getActivity());
        }

        public void testDisplayBlackBox() {

                //Enter 10 in first edit-field
                solo.enterText(0, "10");

                //Enter 20 in first edit-field
                solo.enterText(1, "20");

                //Click on Multiply button
                solo.clickOnButton("Multiply");

                //Verify that resultant of 10 x 20
                assertTrue(solo.searchText("200"));

                }
```

```
    public void testDisplayWhiteBox() {

        //Defining our own values to multiply
        float firstNumber = 10;
        float secondNumber = 20;
        float resutl = firstNumber * secondNumber ;

        //Access First value (edit-filed) and putting firstNumber value in it
        EditText FirsteditText = (EditText) solo.getView(R.id.EditText01);
        solo.enterText(FirsteditText, String.valueOf(firstNumber));

        //Access Second value (edit-filed) and putting SecondNumber value in it
        EditText SecondeditText = (EditText) solo.getView(R.id.EditText02);
        solo.enterText(SecondeditText, String.valueOf(secondNumber));

        //Click on Multiply button
        solo.clickOnButton("Multiply");

        assertTrue(solo.searchText(String.valueOf(resutl)));
        TextView outputField = (TextView) solo.getView(R.id.TextView01);
        //Assert to verify result with visible value
        assertEquals(String.valueOf(resutl), outputField.getText().toString());
    }

    @Override
    protected void tearDown() throws Exception{

            solo.finishOpenedActivities();
    }
}
```

5. Run Test Case

Now as we are almost done so now its time to run our test case.

Right click on *TestMain.java* file select *Run As* option and then click on *Android Junit Test.* It will start running Junit test.

Select the emulator or device to run the test (we will be using Android default emulator) , and wait for a while to see the magic of Robotium.

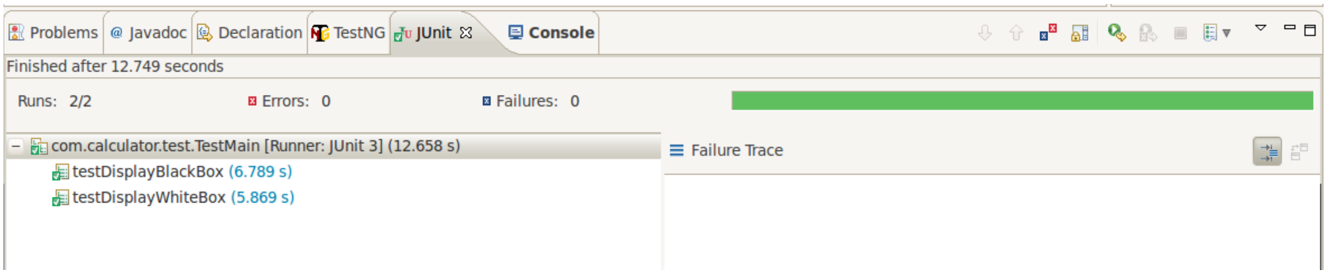
If things are going fine

1. Emulator will load, Unlock it.

2. AndroidCalculator application will load

3. It will automatically enter first & second values in First and Second EditField, and click on Multiply button (you can see all this happening as record & play scripts)

4. After successfully execution it will show green bar showing the successful execution and all results are passed.



```
Finished after 12.749 seconds

Runs:  2/2          Errors:  0          Failures:  0

com.calculator.test.TestMain [Runner: JUnit 3] (12.658 s)
    testDisplayBlackBox (6.789 s)
    testDisplayWhiteBox (5.869 s)
```