

J2MEUnit in 10 minutes
Quick Tutorial to setup & learn J2MEUnit

By

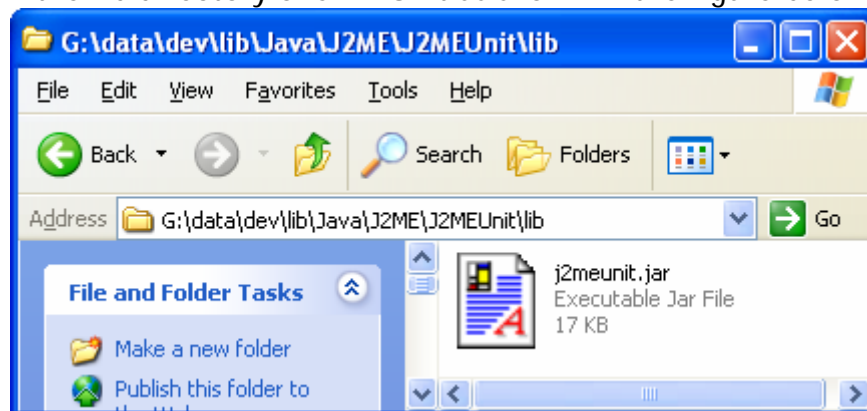
Gunjan Doshi
Gunjan At [instrumentalservices.com](http://www.instrumentalservices.com)
<http://www.instrumentalservices.com>

This tutorial provides a step-by-step guide to a programmer looking to setup J2MEUnit, an automatic unit testing framework for J2ME.

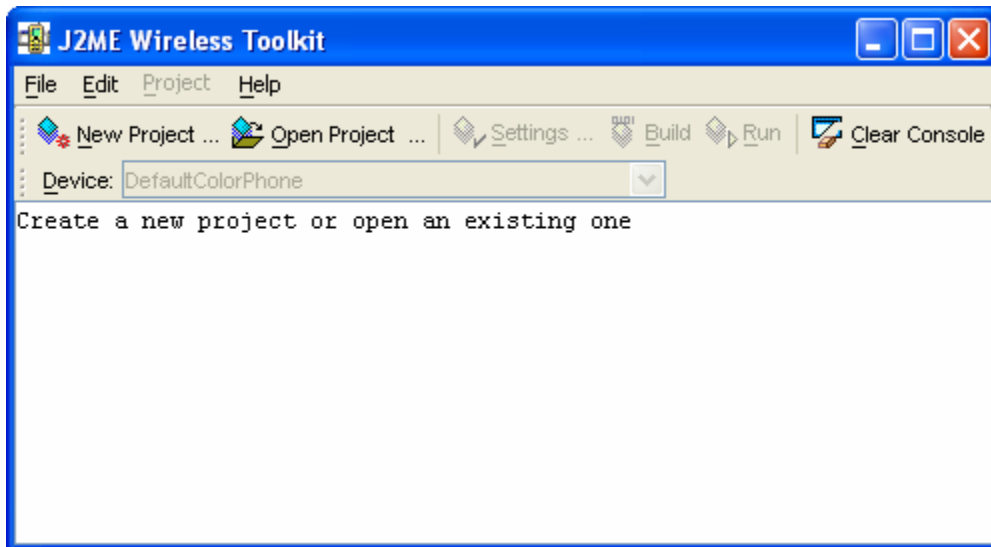
Any good article has to give some history. I will not break the tradition. If you do not like reading History, skip straight to the next paragraph. J2ME development is challenging. From small foot-print API to limited hardware, a programmer feels restrained with several resource constraints. Recently, I was brought on a J2ME project. I also felt restrained. However, my feeling of being restrained came from a different need. I practice & recommend Test-driven development and the first thing I need, when, I start on working on any project, is an automated unit testing framework. I was under the impression that I could use JUnit framework for this particular J2ME project. However, JUnit cannot be used with J2ME because JUnit requires reflection API and J2ME has no reflection. There are ways to make JUnit work partially with J2ME. However, it is outside the scope of this article. I came across [J2MEUnit](#), which is targeted for J2ME. This guide will show you how to setup J2MEUnit in a modern IDE like Eclipse and write your first test case. Once you have followed all the steps you will have a test-driven development environment ready to go.

Pre-requisites:

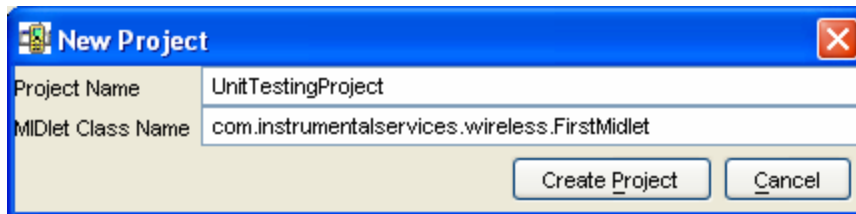
1. You have JDK 1.3 or higher installed. You can check this by typing “java -version” on the console.
2. You have a modern IDE like Eclipse installed. This article refers to Eclipse, but you can use any other IDE.
3. You have J2ME Wireless toolkit installed. If not, download J2ME Wireless toolkit from [here](#). J2ME Wireless toolkit is not required for unit testing. It is only required to compile, preverify and run a midlet in the emulator.
4. Download j2meunit-all.zip1.1.1 from [here](#). Extract the zip to a folder where you keep your external libraries. In my case, I extracted it to G:\data\dev\lib\Java\J2ME. We are only interested in the j2meunit.jar file in the lib directory of J2MEUnit as shown in the figure below:



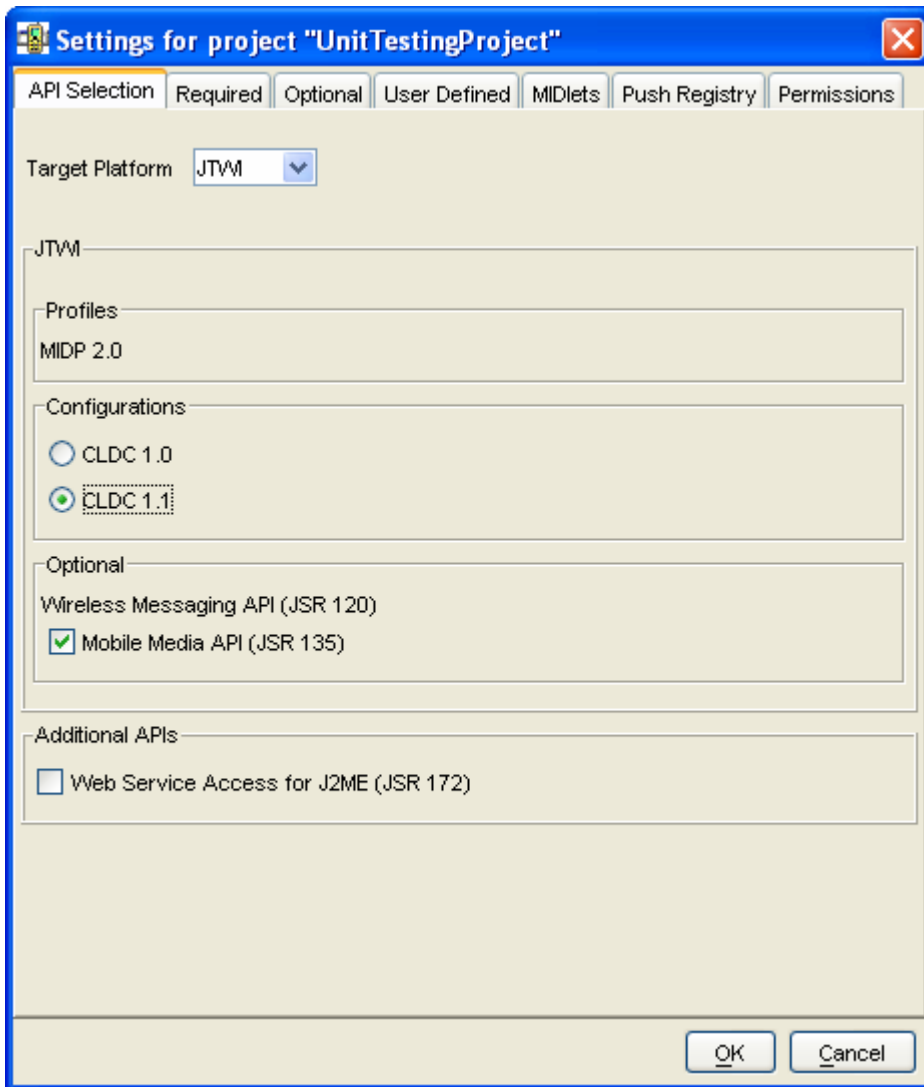
Step I: You have to first create the project using the J2ME wireless toolkit. Start the Ktoolbar that gets installed as part of the wireless toolkit.



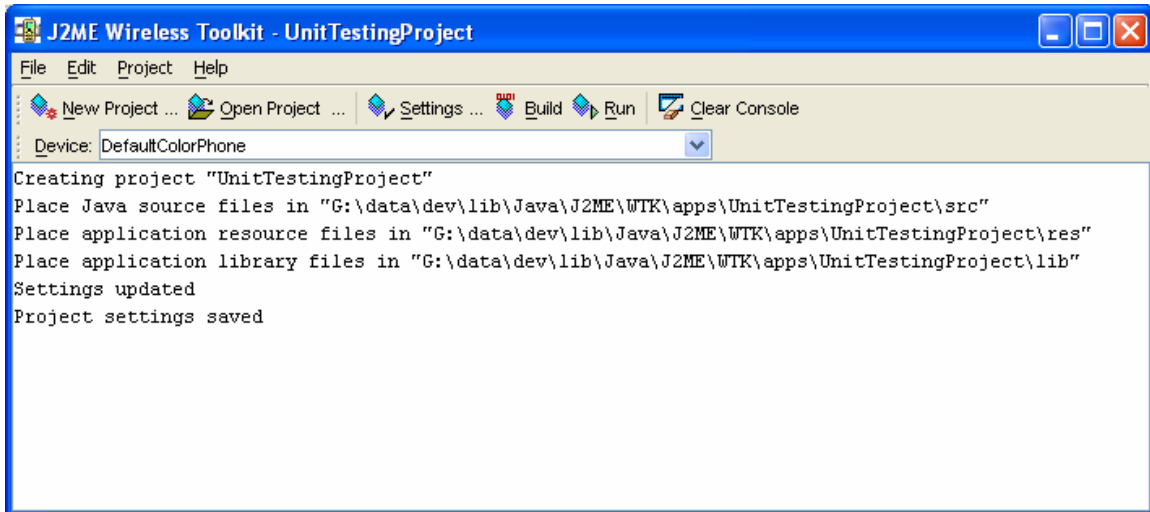
Step II: Click on the 'New Project' button in the toolkit. A 'New Project' window pops up. Call the project as 'UnitTestingProject'. You will also have to enter the name of the midlet class. In my case, I enter a fully qualified name: com.instrumentalservices.wireless.FirstMidlet.



Step III: Once you click on the 'Create Project' button, a new settings window will pop-up. Change the settings as you need. Here is the window from my setup:



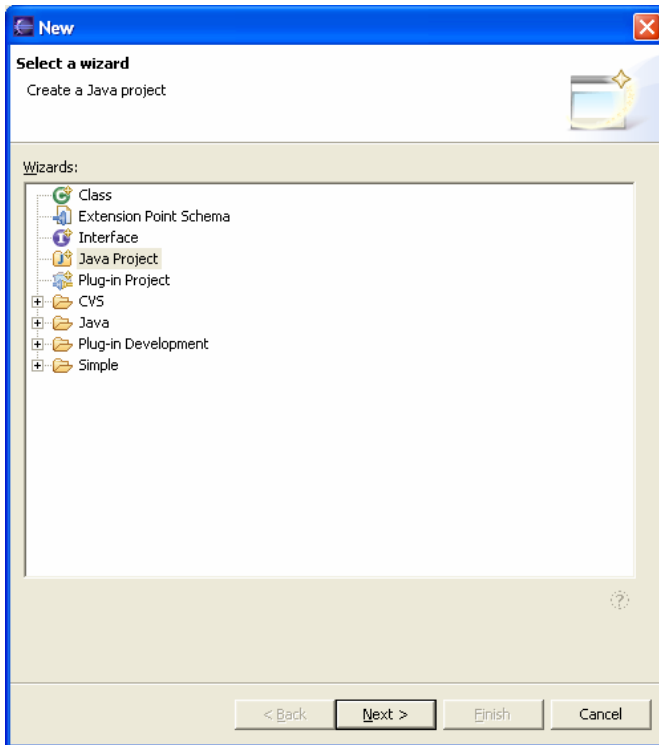
Step IV: Once you click the 'OK' button, you should see few log messages in wireless toolkit, telling you that the project creation was successful and it created the src, res and lib directory.



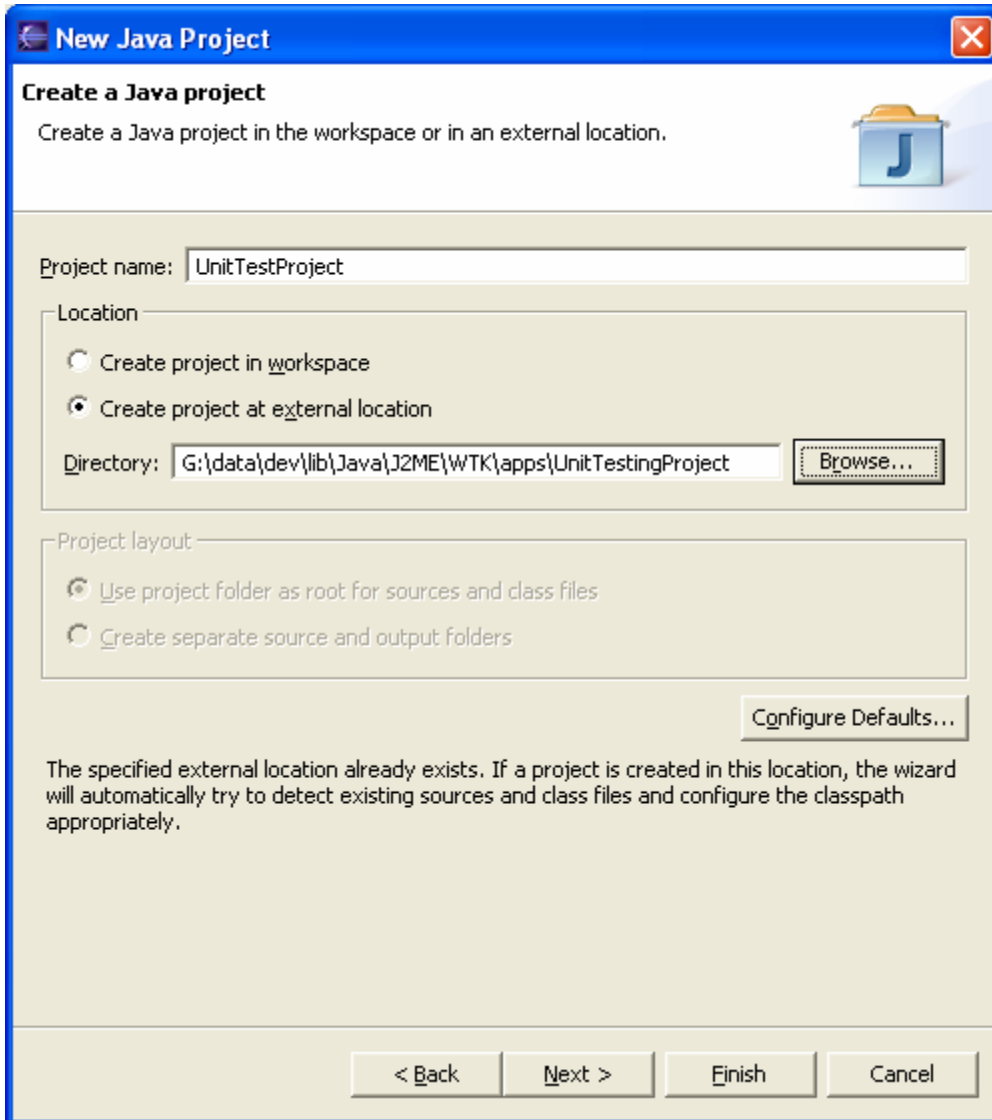
Step V: Start windows explorer and open the apps directory under the toolkit installation folder and check if the UnitTestingProject is created under applications. Make sure the src, res and the lib directory got generated.

Step VI: Copy the j2meunit.jar from the lib directory of J2MEUnit installation folder to the lib directory of the UnitTestProject project you just created. The project folder is under apps folder of wireless toolkit.

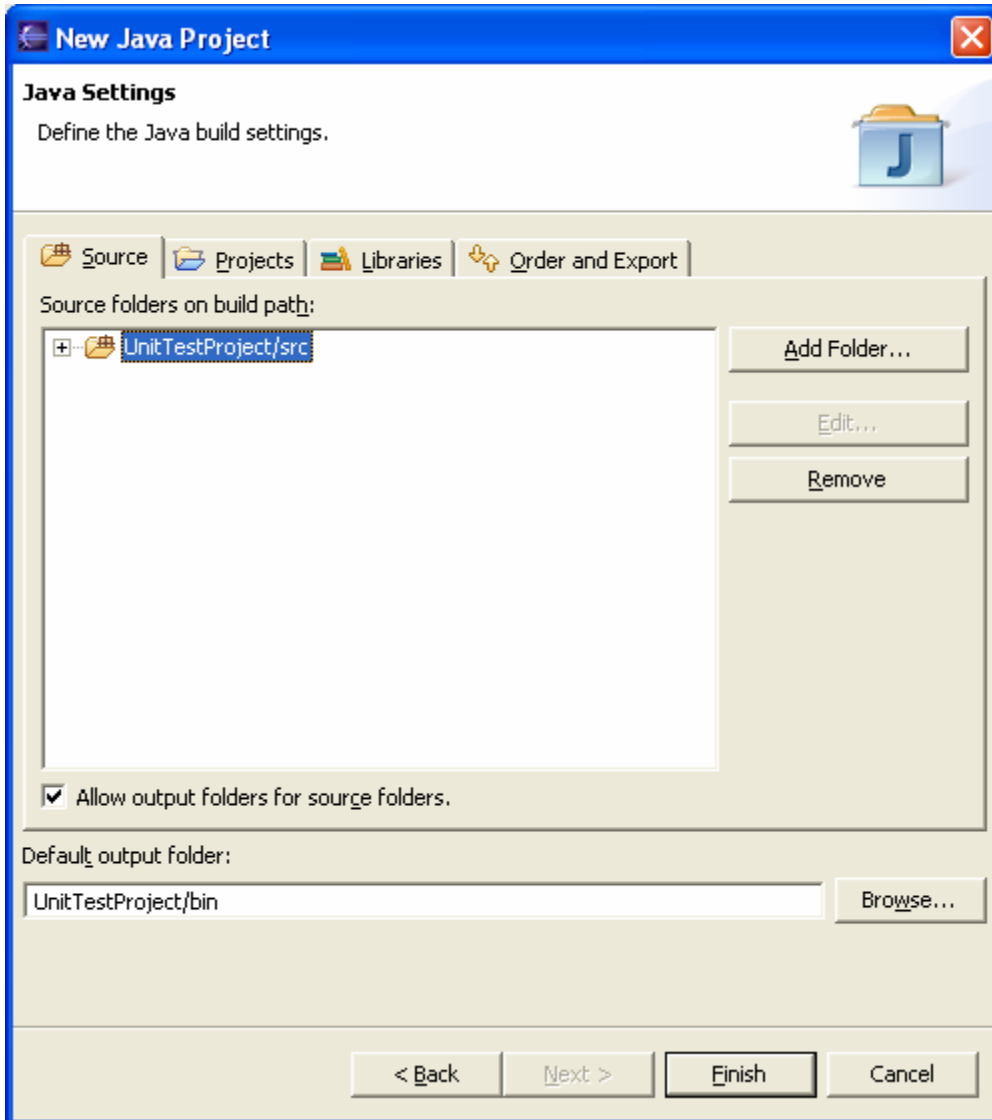
Step VII: Start your favorite IDE like Eclipse or IntelliJ. In my case, I will use Eclipse. Create a new project as shown here:



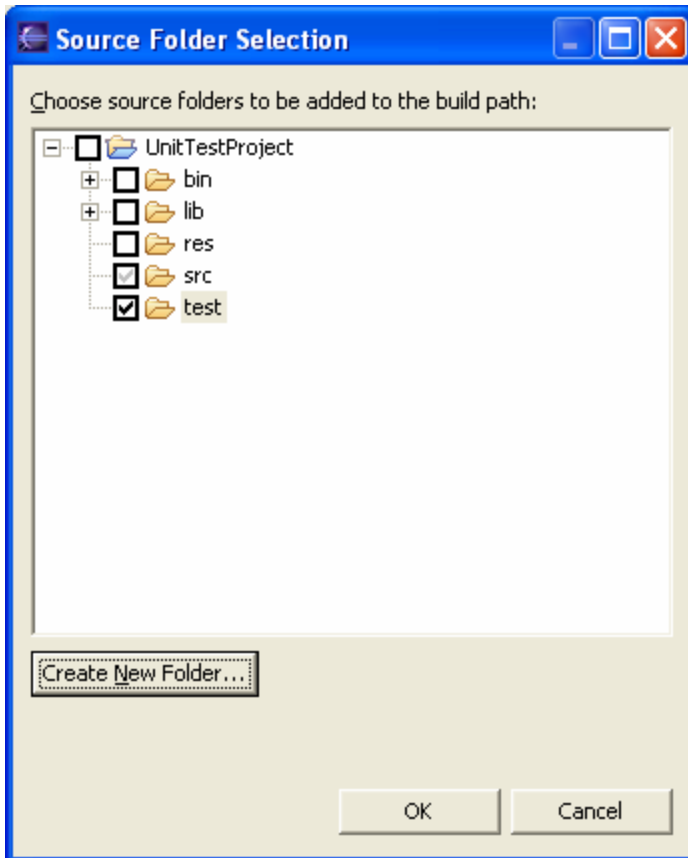
Step VIII: Press the 'Next' button, and choose to create the project at an external location. Choose browse and goto the apps folder under the wireless toolkit and choose UnitTestingProject that you created before. This is shown in the diagram below:



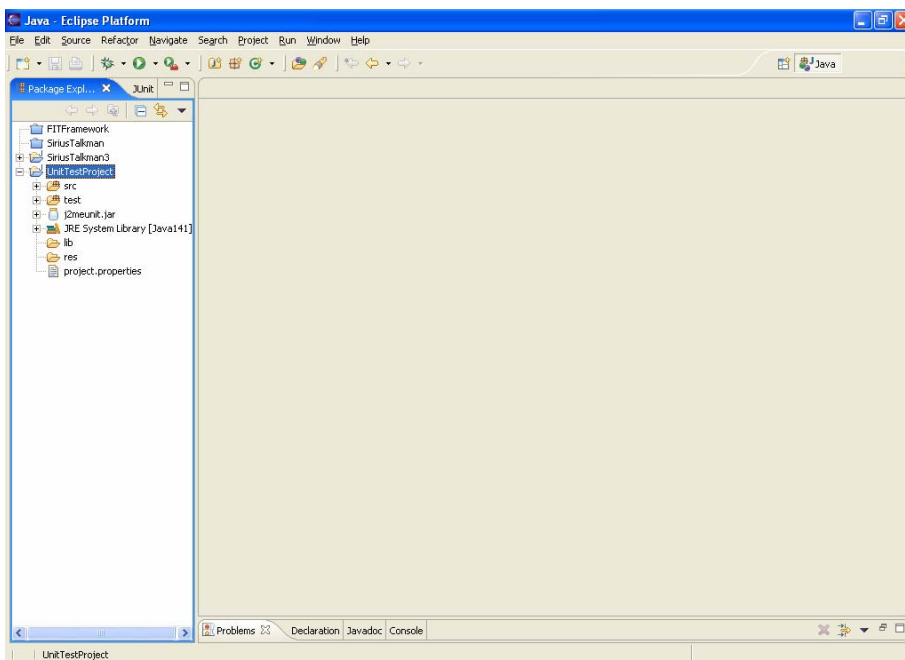
Step ix: Click on the 'Next button' and select the src folder already listed.



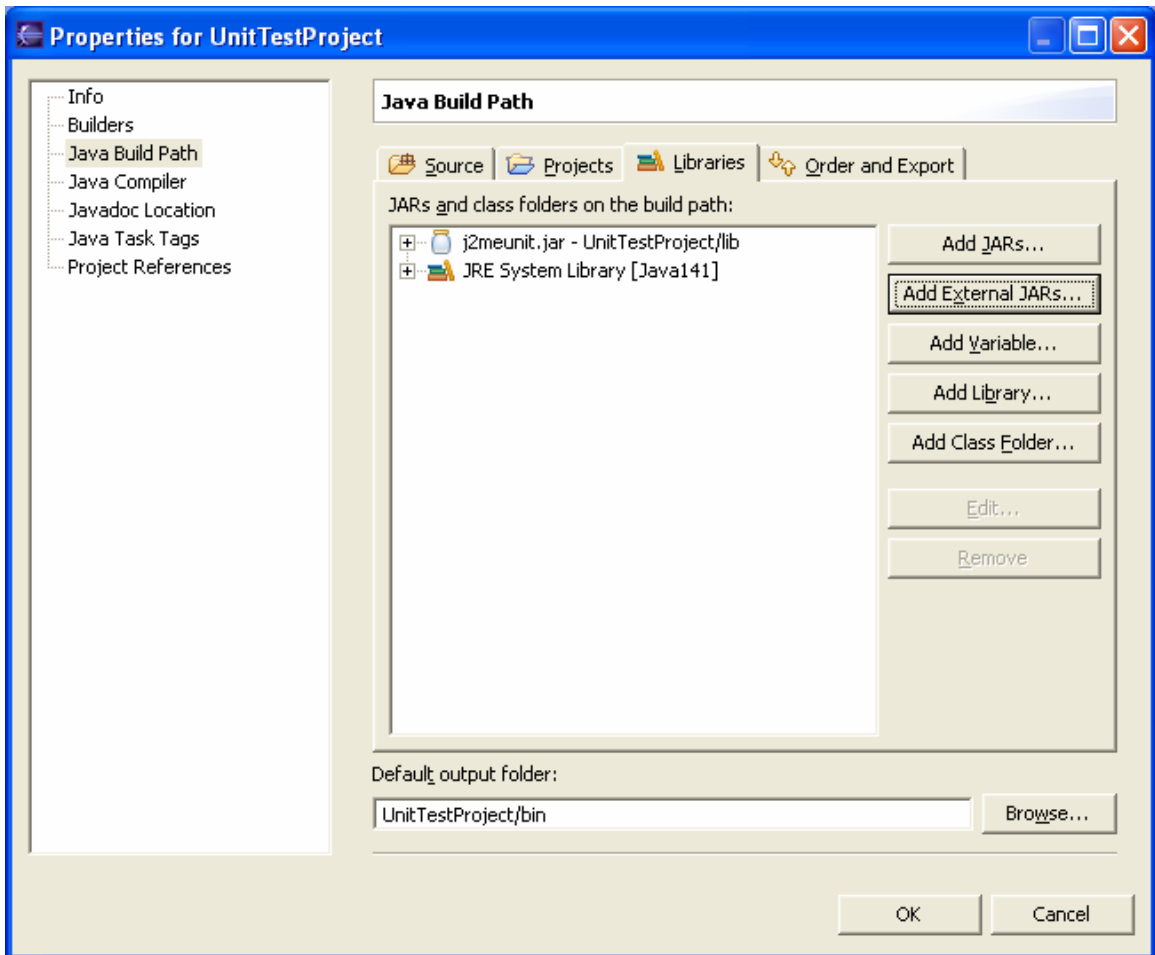
Click on the 'Add Folder' and then create a new folder called test. This is where all your test classes will go.



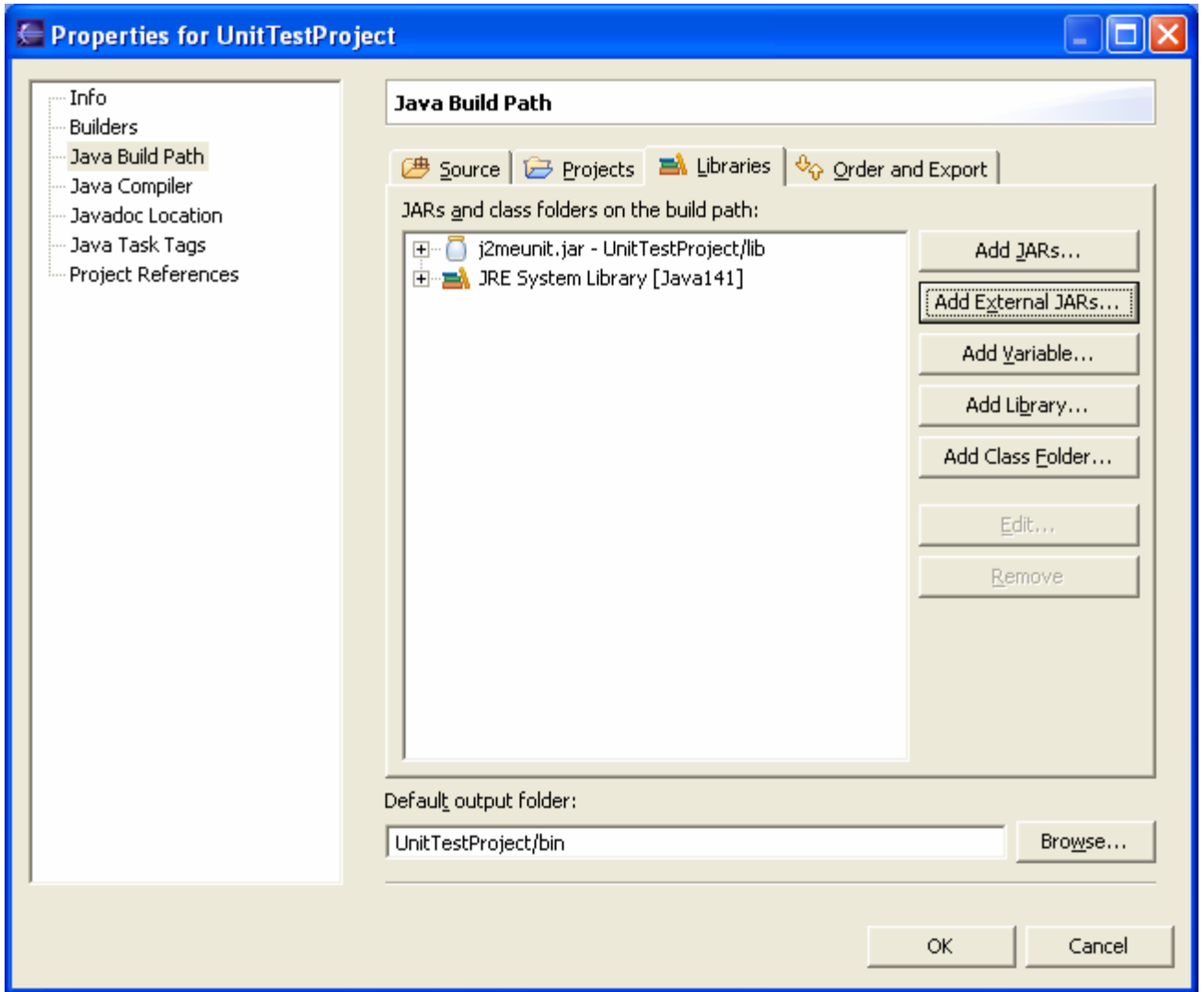
Click the 'OK' button and then the 'Finish' button. Your Eclipse workspace should be ready and should now look like this:



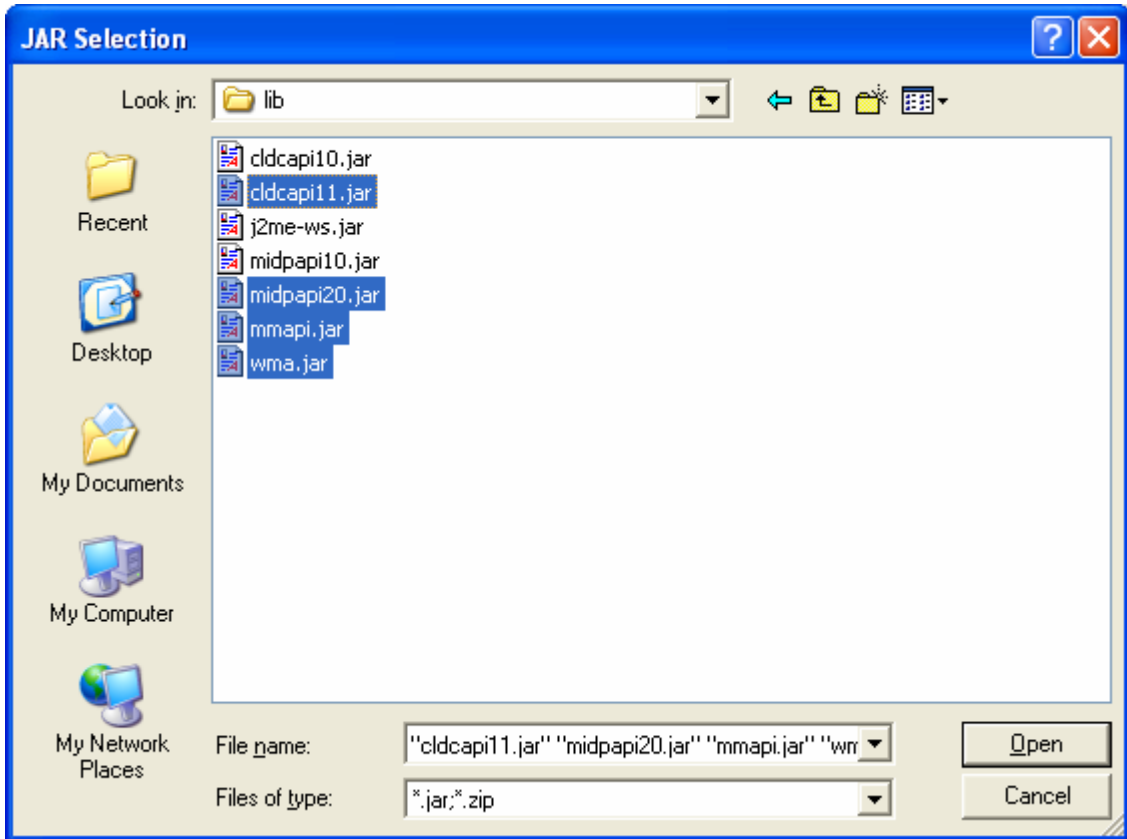
Check the properties of the project and make sure the classpath contains j2meunit.jar as shown in the diagram below:

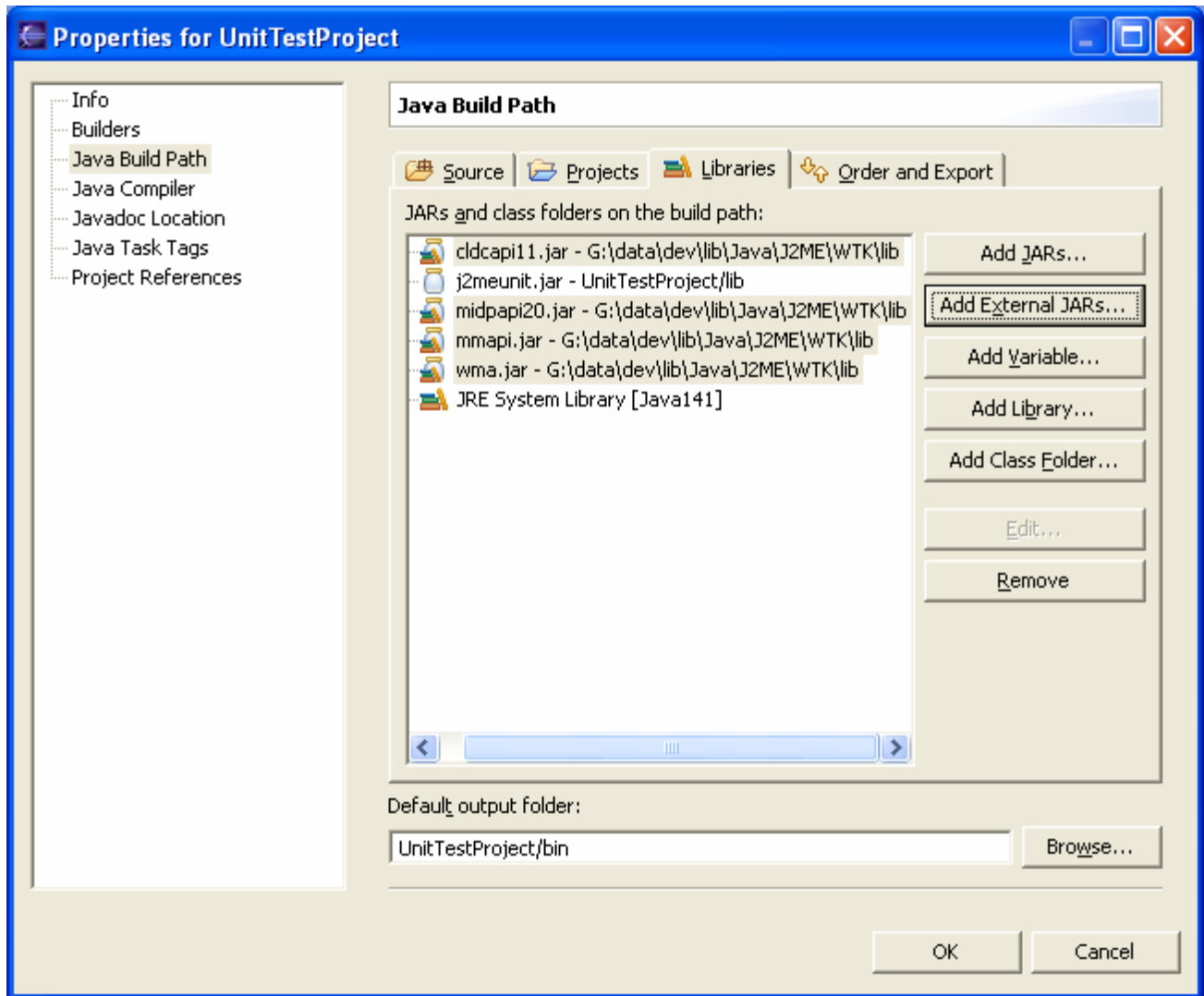


You need to add additional jars required by J2ME by clicking on the 'Add External Jars' button.

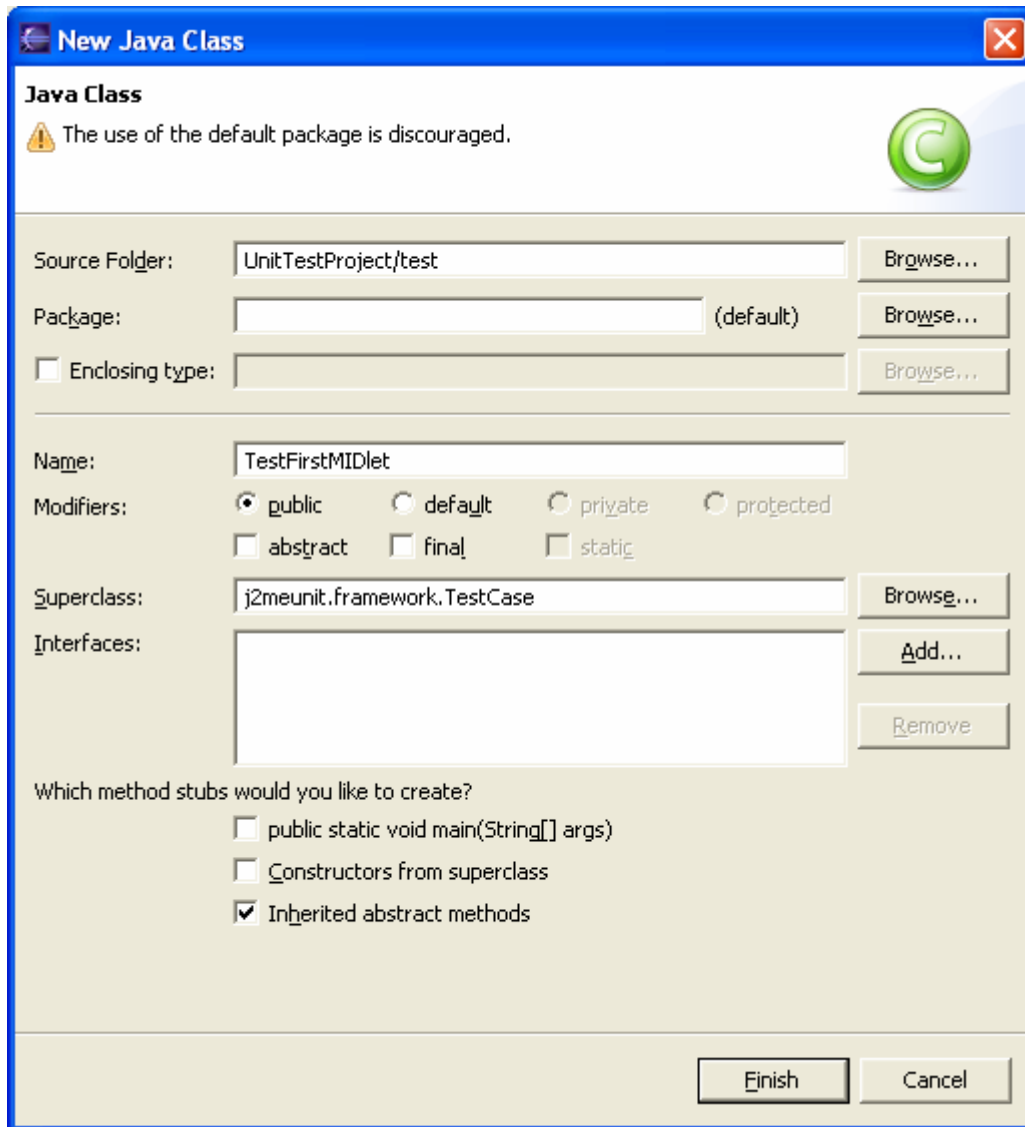


These jars are in the lib directory of the wireless toolkit.





Step x: Create a new class under the test folder and call it MyFirstTest. Make sure it extends `j2meunit.framework.TestCase`.



Code it, so that it look likes the following:

```

import j2meunit.framework.Test;
import j2meunit.framework.TestCase;
import j2meunit.framework.TestMethod;
import j2meunit.framework.TestSuite;

public class TestFirstMidlet extends TestCase {
    public TestFirstMidlet (String arg0, TestMethod arg1) {
        super(arg0, arg1);
    }

    public TestFirstMidlet () {
        super();
    }

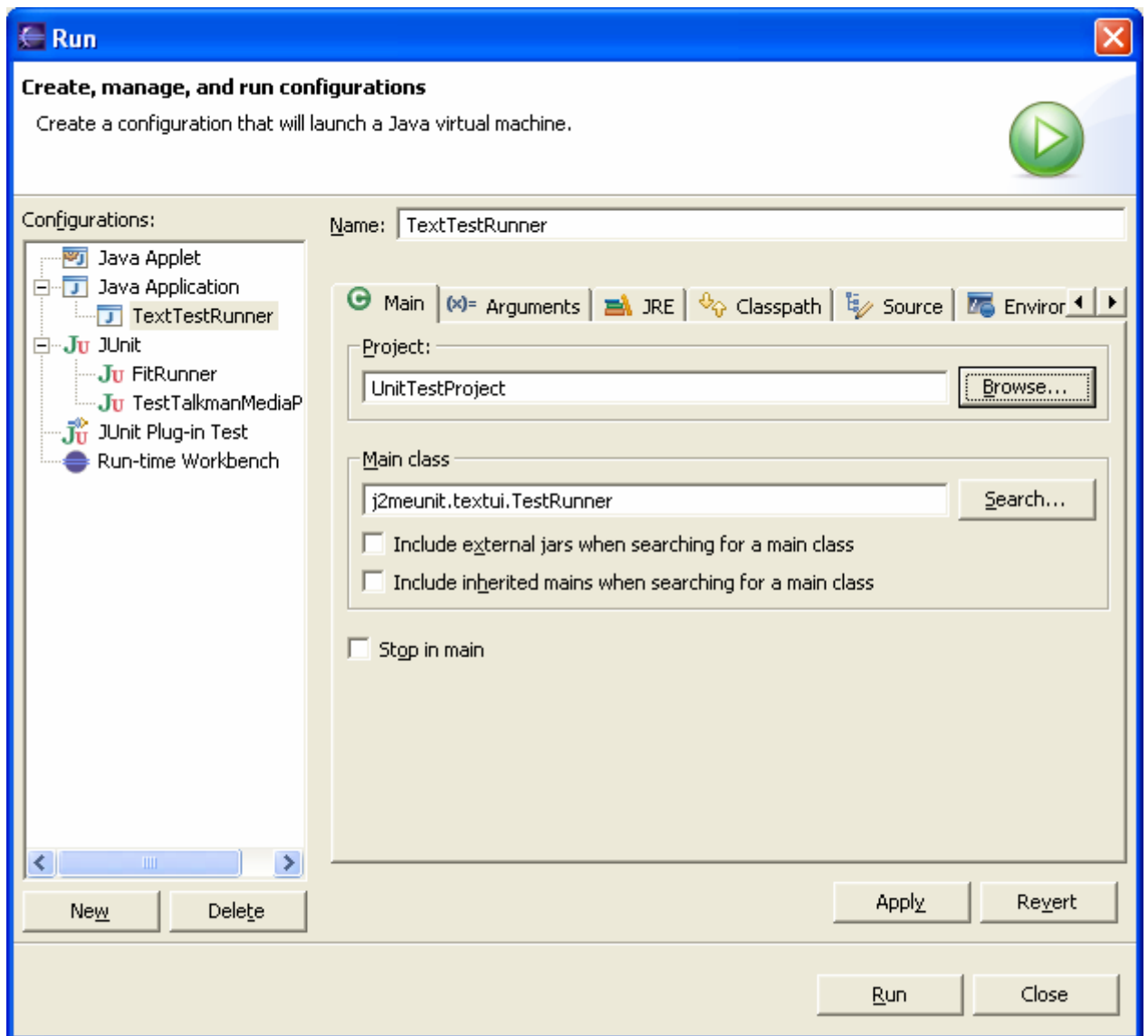
    public TestFirstMidlet (String arg0) {
        super(arg0);
    }

    //Actual test method. It has to start with test.
    protected void testAlwaysFails() {
        try {
            assertTrue(false);
        } catch (Exception e) {
            throw new RuntimeException("Problem creating the player");
        }
    }

    //suite method that returns all the tests to be run
    //Here it returns only one test - testAlwaysFails
    public Test suite() {
        TestSuite suite = new TestSuite();
        suite.addTest(new TestFirstMidlet ("First Test",
            new TestMethod()
            {
                public void run(TestCase tc)
                {
                    ((TestFirstMidlet)tc).testAlwaysFails();
                }
            }
        ));
        return suite;
    }
}

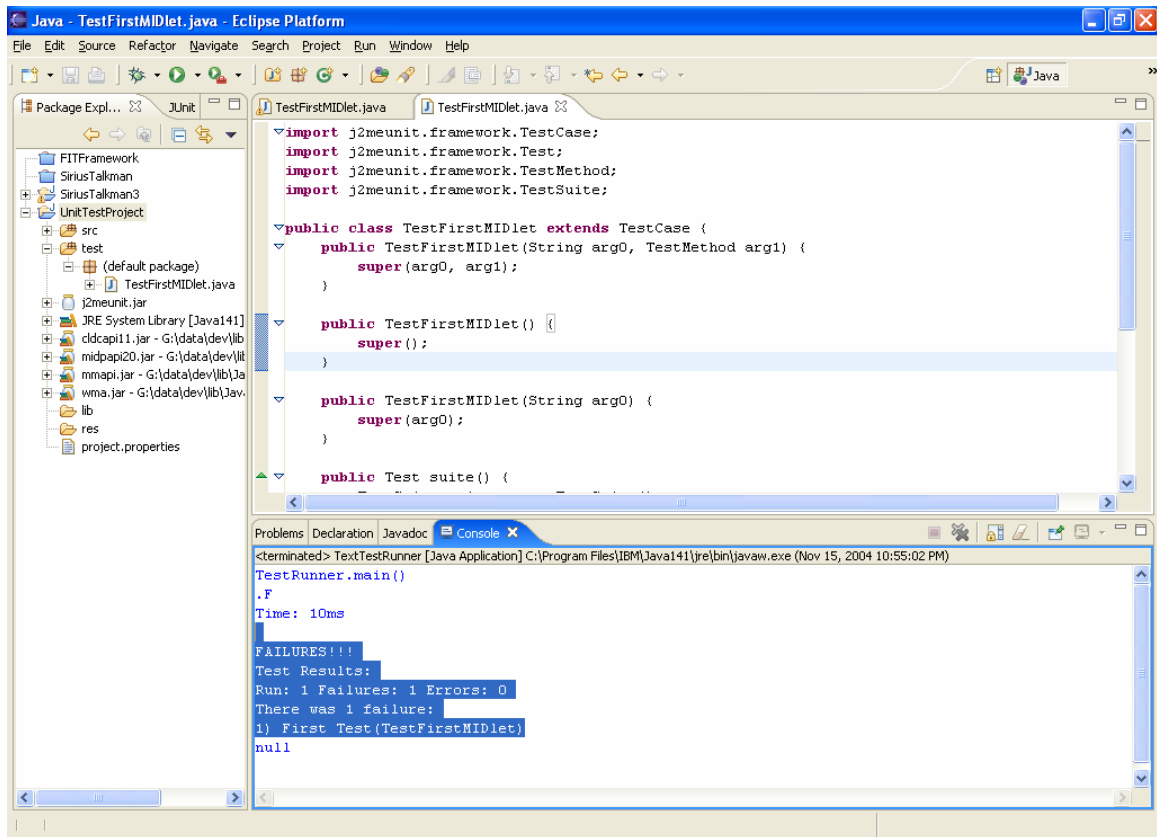
```


The following window appears. Select Java Application configuration and hit the 'New' button available on left bottom. Enter the information as shown in the diagram below:



Finally, click the 'Arguments' tab and enter TestFirstMidlet in the Program arguments window. This is required to tell the J2MEUnit, which midlet to run.

Click Apply and then hit Run. You should see the test fail as expected in the console in Eclipse.



Congrats on your first failing test. Use the quick reference guides available under the articles section of our main site <http://www.instrumentalservices.com> to get you jump started on Test-Driven development.