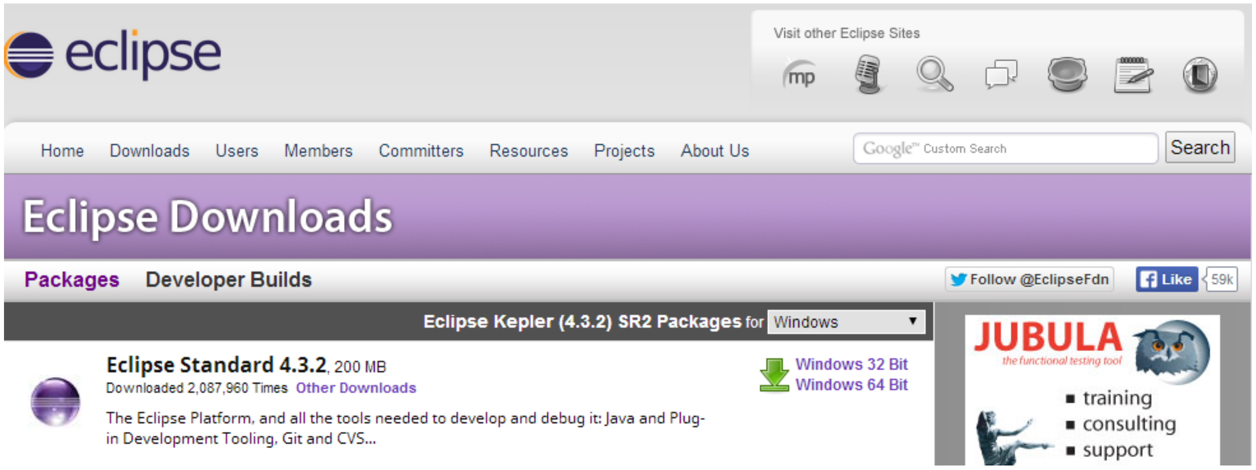
Using JUnit With Eclipse

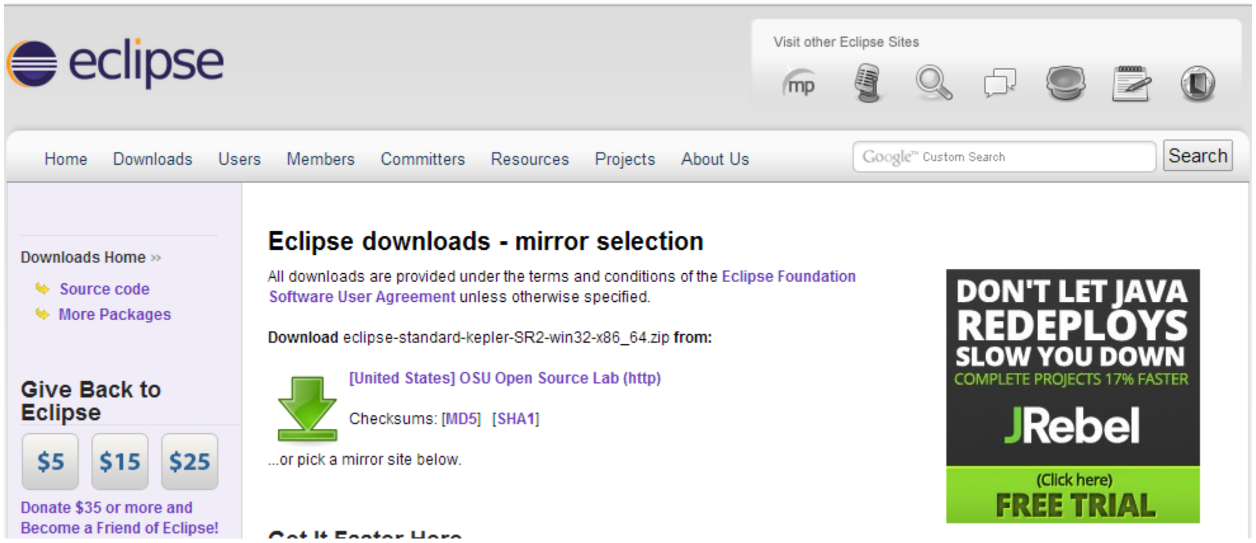
First, you need to obtain Eclipse

Go to https://www.eclipse.org/downloads/ and choose the package for your machine.


eclipse
Visit other Eclipse Sites
Home Downloads Users Members Committers Resources Projects About Us
Google™ Custom Search
Search
Eclipse Downloads
Packages
Developer Builds
Follow @EclipseFdn
Like 59k
Eclipse Kepler (4.3.2) SR2 Packages for Windows
JUBULA the functional testing tool
training
consulting
support

Eclipse Standard 4.3.2, 200 MB
Downloaded 2,087,960 Times Other Downloads
The Eclipse Platform, and all the tools needed to develop and debug it: Java and Plug-in Development Tooling, Git and CVS...
Windows 32 Bit
Windows 64 Bit

Next choose the mirror to download from.


eclipse
Visit other Eclipse Sites
Home Downloads Users Members Committers Resources Projects About Us
Google™ Custom Search
Search
Downloads Home »
Source code
More Packages
Give Back to Eclipse
$5 $15 $25
Donate $35 or more and Become a Friend of Eclipse!
DON'T LET JAVA REDEPLOYS SLOW YOU DOWN
COMPLETE PROJECTS 17% FASTER
JRebel
(Click here)
FREE TRIAL

Eclipse downloads - mirror selection

All downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified.

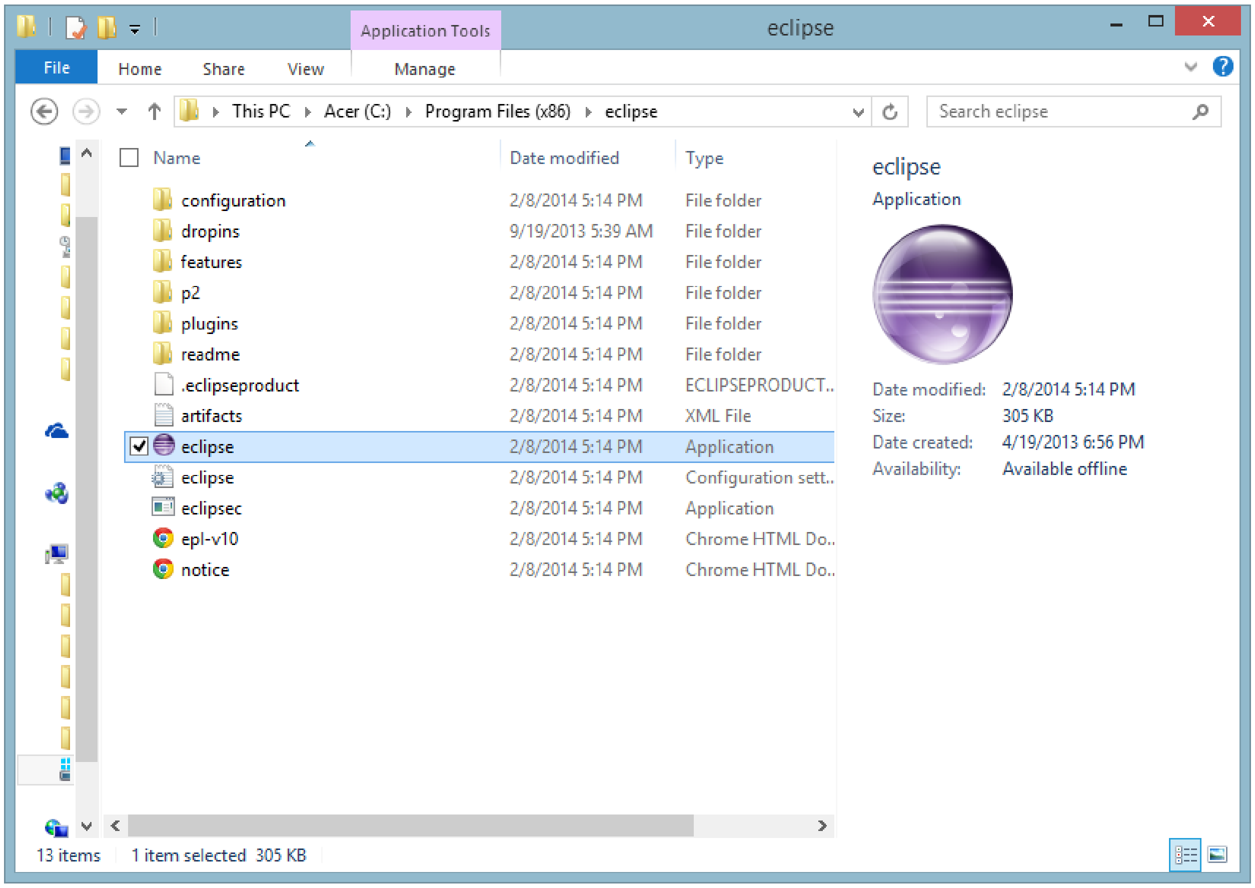
Download eclipse-standard-kepler-SR2-win32-x86_64.zip **from:**

[United States] OSU Open Source Lab (http)

Checksums: [MD5] [SHA1]

...or pick a mirror site below.

Once you finish download, go ahead and open the zip file and extract the files into a folder of your choice and then you can run the executable.



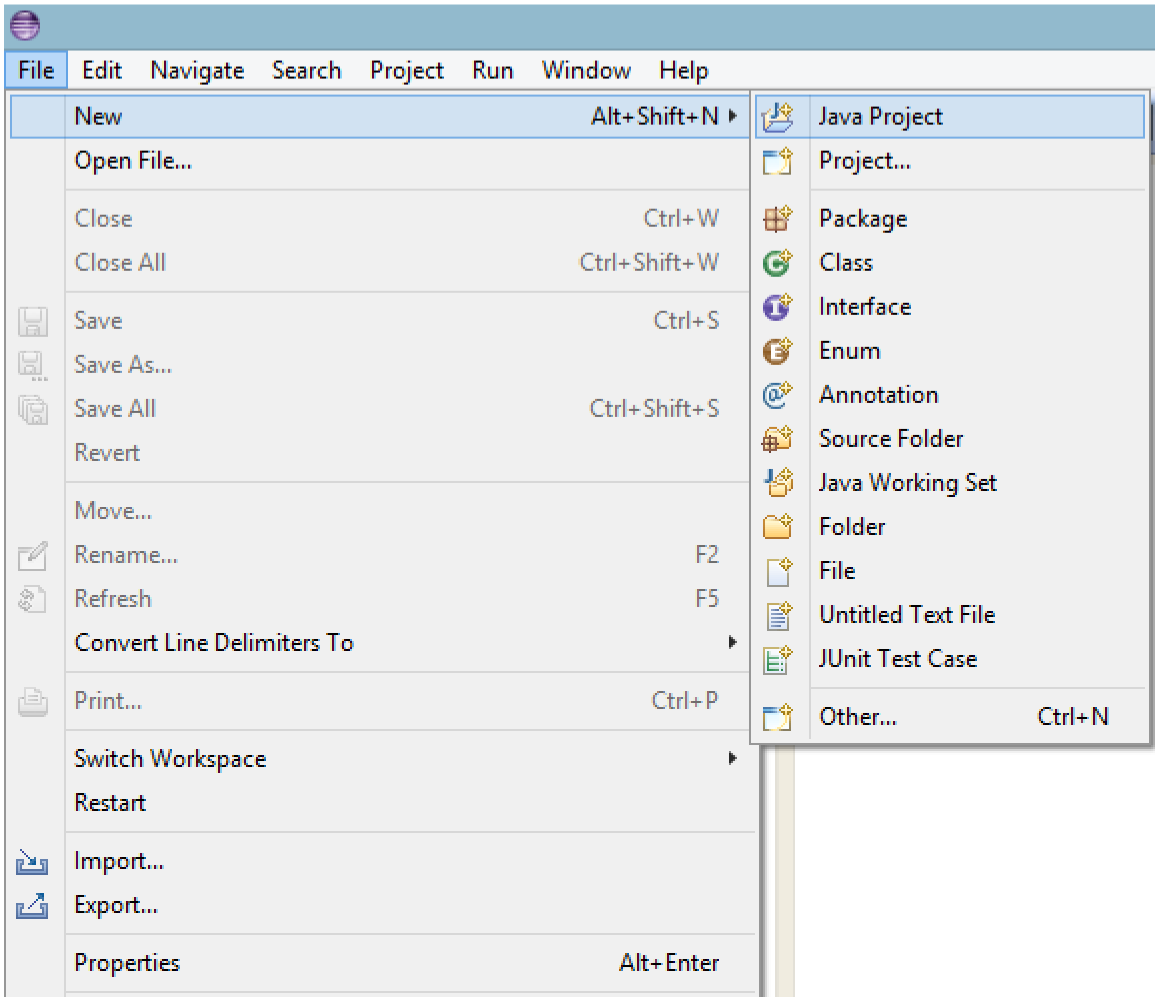
Name	Date modified	Type
configuration	2/8/2014 5:14 PM	File folder
dropins	9/19/2013 5:39 AM	File folder
features	2/8/2014 5:14 PM	File folder
p2	2/8/2014 5:14 PM	File folder
plugins	2/8/2014 5:14 PM	File folder
readme	2/8/2014 5:14 PM	File folder
.eclipseproduct	2/8/2014 5:14 PM	ECLIPSEPRODUCT..
artifacts	2/8/2014 5:14 PM	XML File
eclipse	2/8/2014 5:14 PM	Application
eclipse	2/8/2014 5:14 PM	Configuration sett..
eclipsec	2/8/2014 5:14 PM	Application
epl-v10	2/8/2014 5:14 PM	Chrome HTML Do..
notice	2/8/2014 5:14 PM	Chrome HTML Do..

eclipse
Application

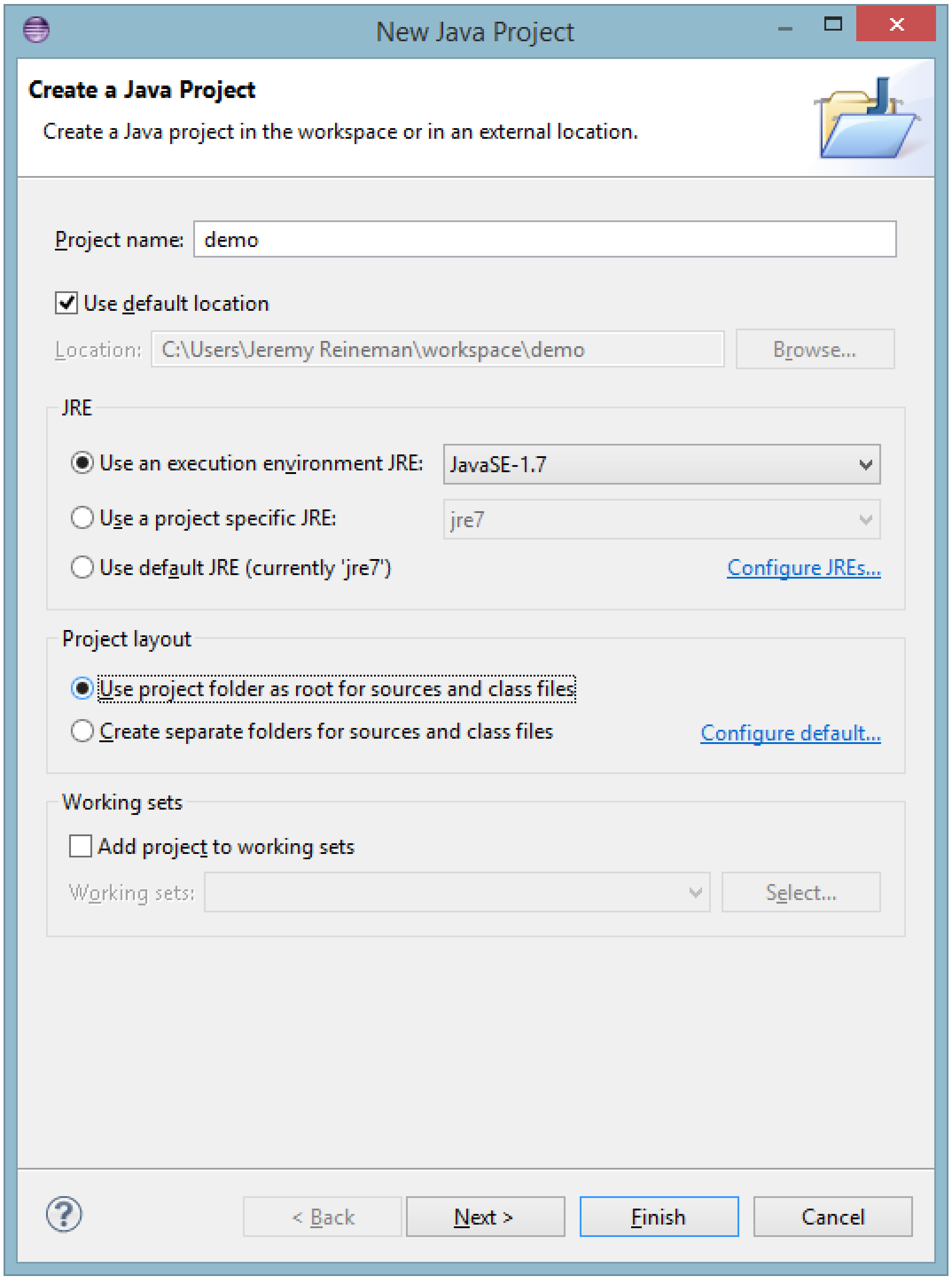
Date modified:	2/8/2014 5:14 PM
Size:	305 KB
Date created:	4/19/2013 6:56 PM
Availability:	Available offline

13 items   1 item selected 305 KB

Once you have Eclipse up and running it is time to start a new **Java Project**.



Now go ahead and name your project and select **Use project folder as root for sources and class files** and select **Finish**.


New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: demo

[✓] Use default location

Location: C:\Users\Jeremy Reineman\workspace\demo

JRE

(●) Use an execution environment JRE: JavaSE-1.7

( ) Use a project specific JRE: jre7

( ) Use default JRE (currently 'jre7')

Project layout

(●) Use project folder as root for sources and class files

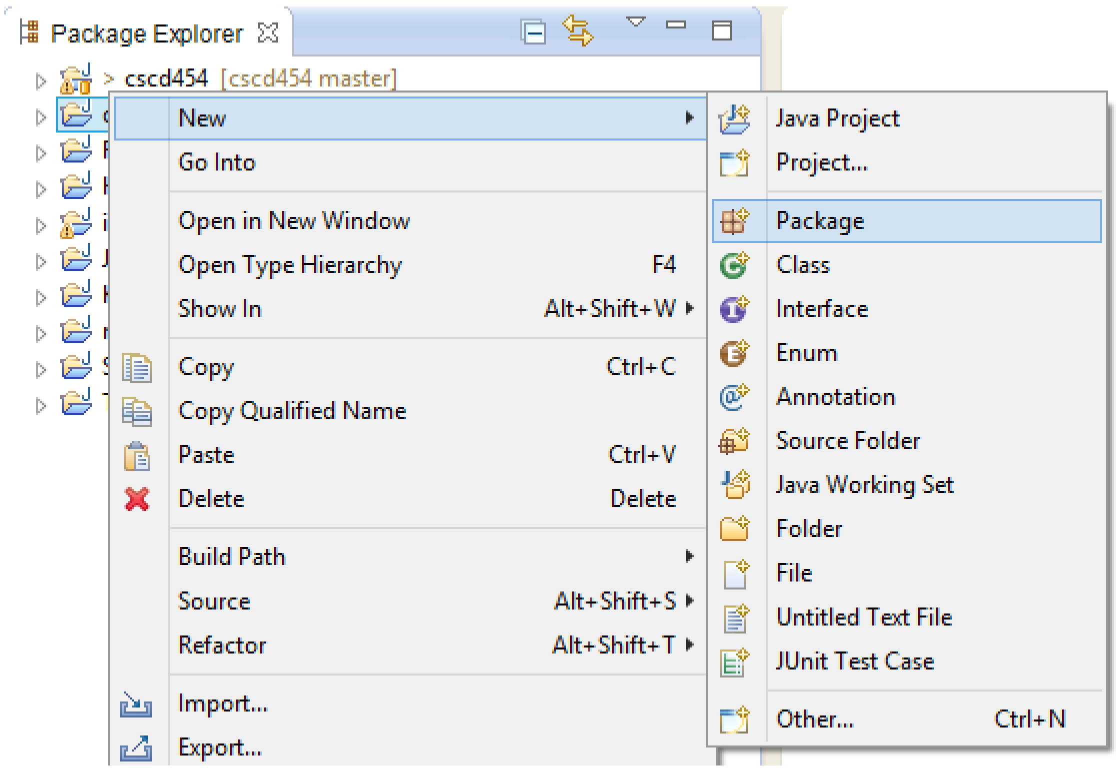
( ) Create separate folders for sources and class files

Working sets

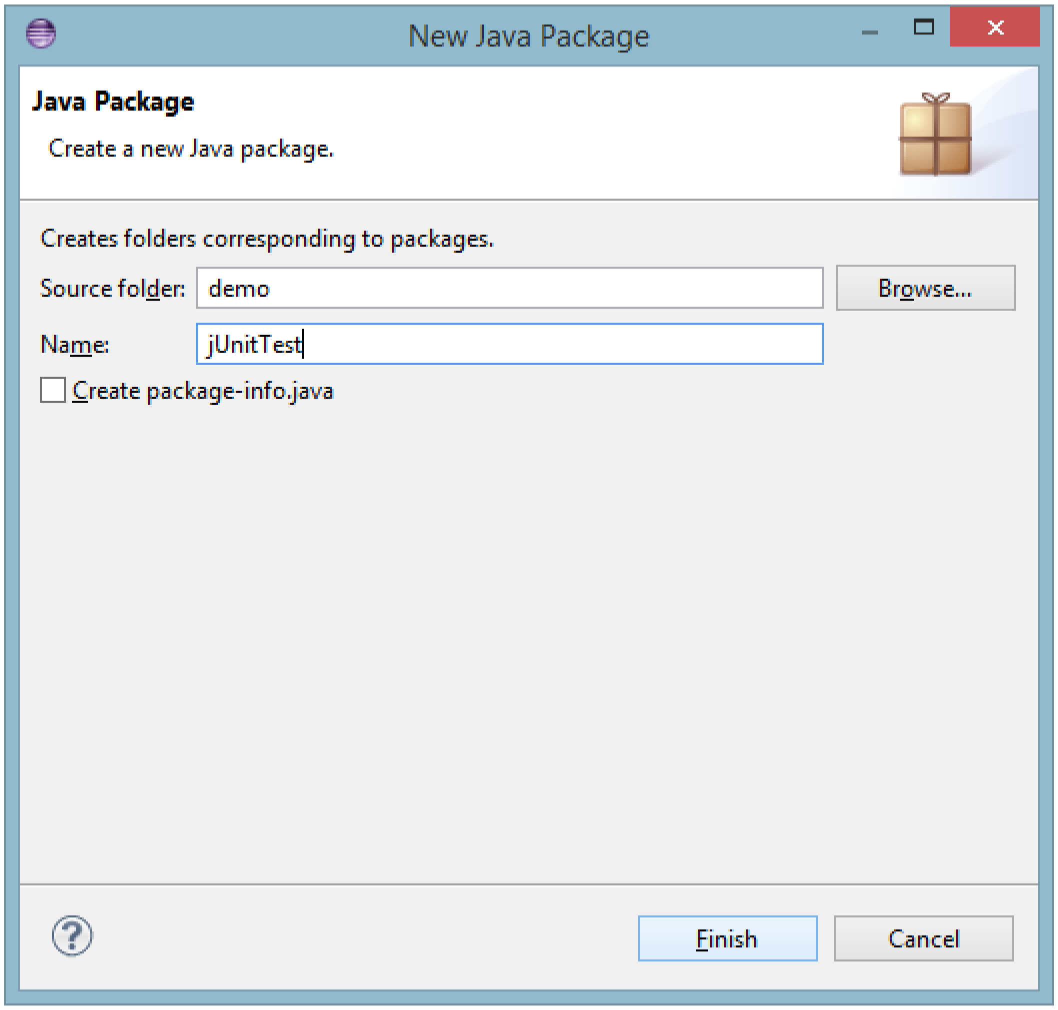
[ ] Add project to working sets

Working sets:

Now go ahead and right-click your project in the project folder and select **New→Package**.



Name your package.



Java Package

Create a new Java package.

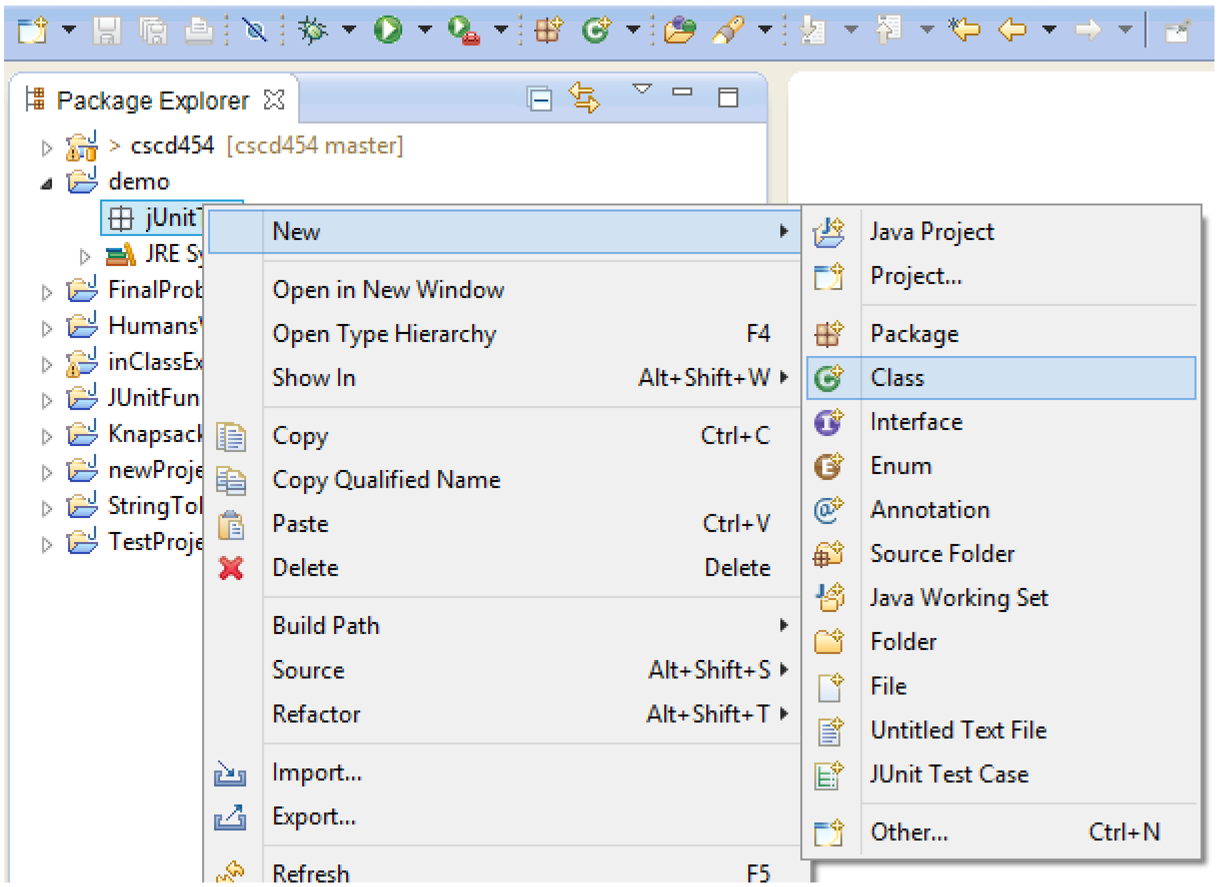
Creates folders corresponding to packages.

Source folder: demo

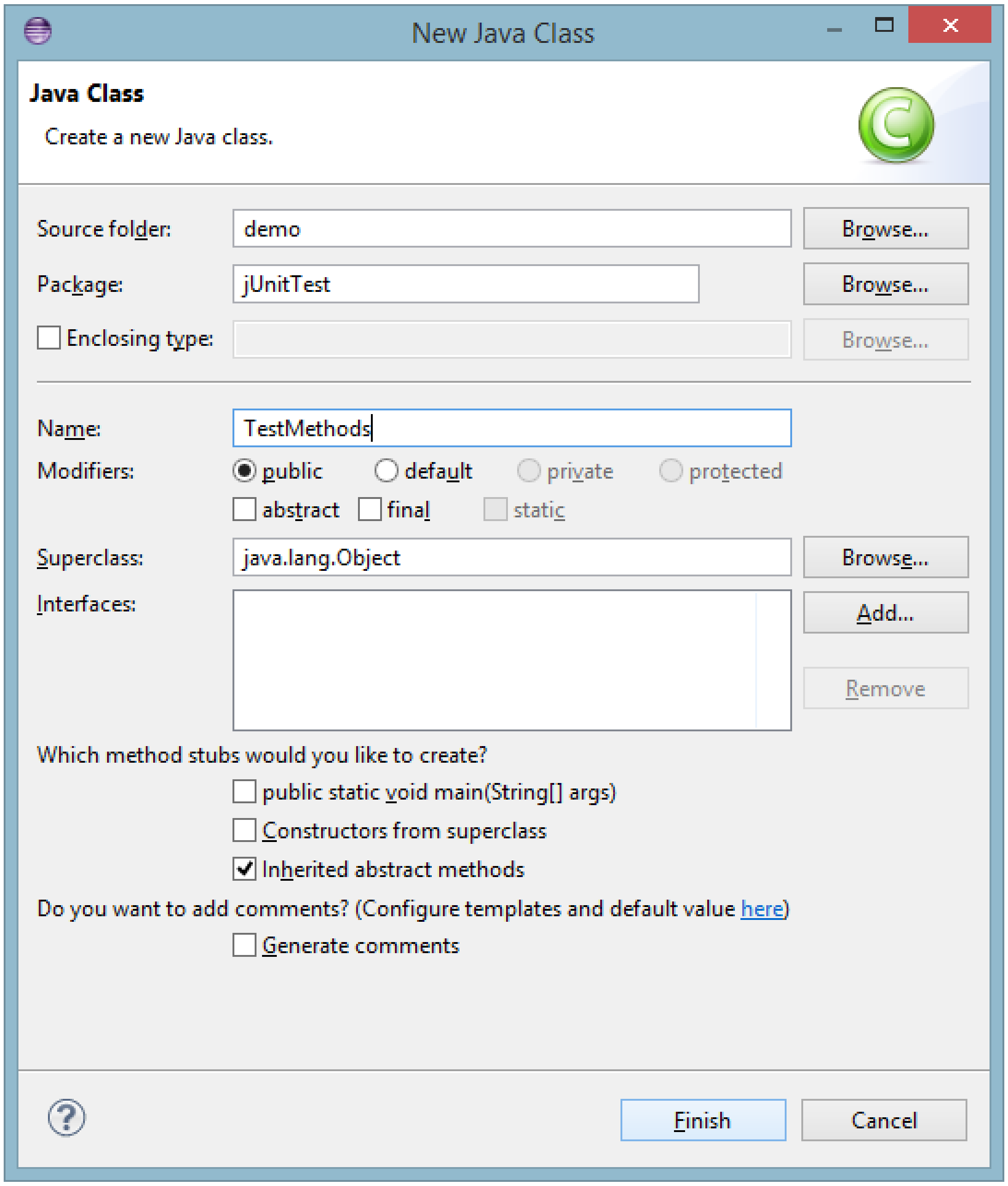
Name: jUnitTest

☐ Create package-info.java

Now go ahead and right-click your package in the project folder and select **New→Class**.



Name your class and select **Finish**.



New Java Class

Java Class

Create a new Java class.

Source folder: demo

Package: jUnitTest

☐ Enclosing type:

Name: TestMethods

Modifiers: [✓ public] default private protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

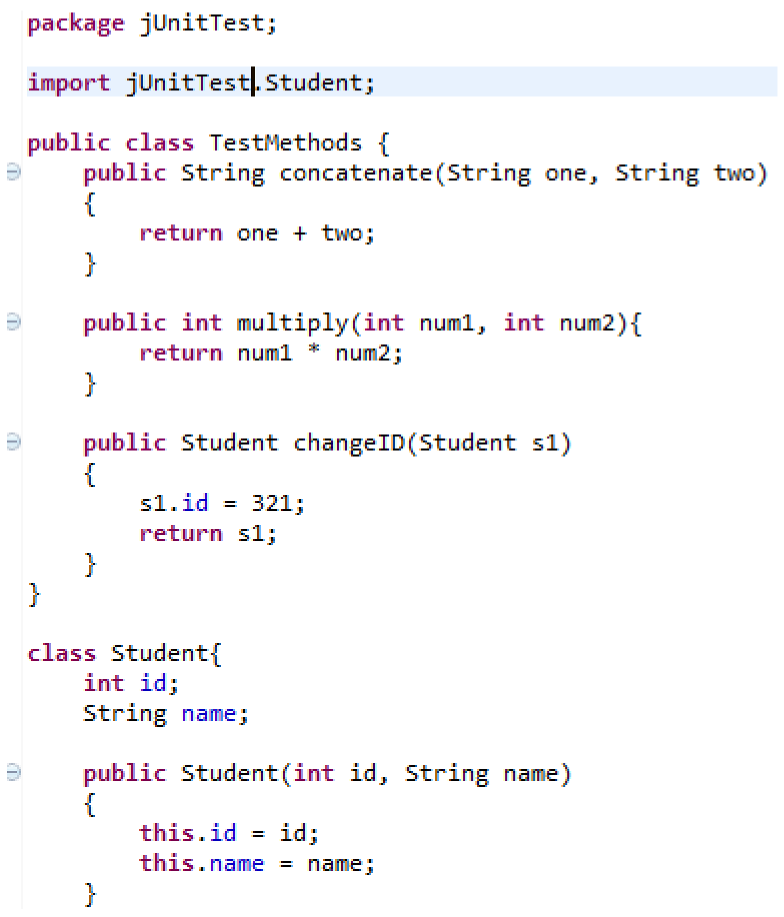
☐ Constructors from superclass

☑ Inherited abstract methods

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

☐ Generate comments

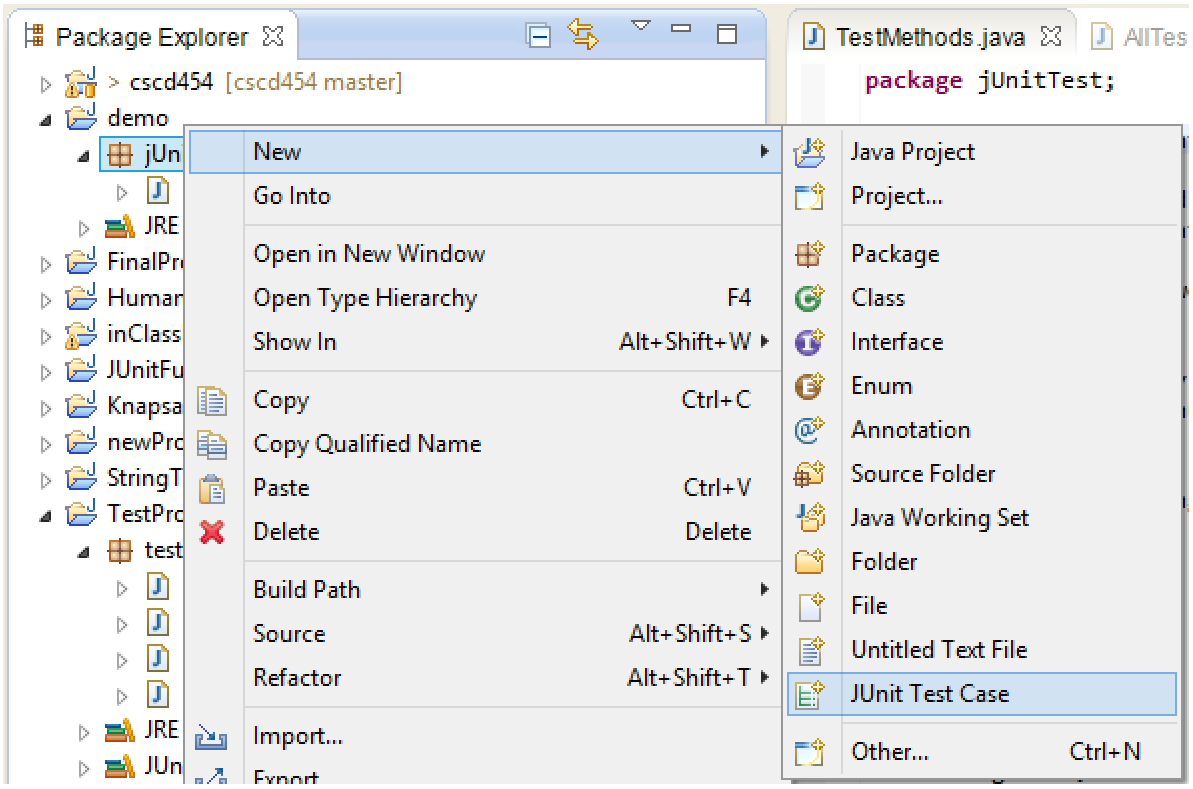
Now go ahead and put a few methods in the class to test basic things like string concatenation, multiplication or object tests.



```java
package jUnitTest;

import jUnitTest.Student;

public class TestMethods {
    public String concatenate(String one, String two)
    {
        return one + two;
    }

    public int multiply(int num1, int num2){
        return num1 * num2;
    }

    public Student changeID(Student s1)
    {
        s1.id = 321;
        return s1;
    }
}

class Student{
    int id;
    String name;

    public Student(int id, String name)
    {
        this.id = id;
        this.name = name;
    }
```

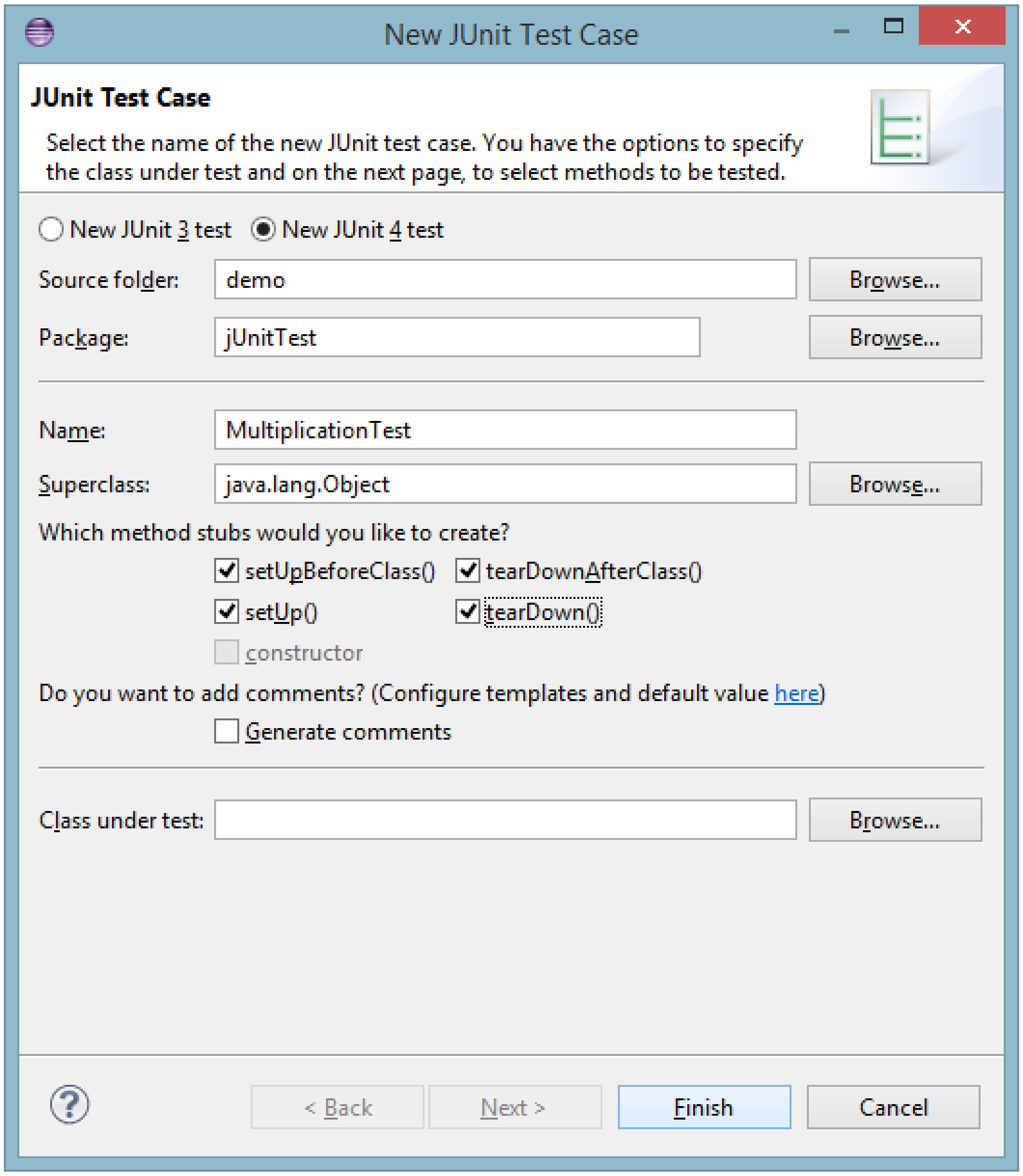
Now that we have some methods to test, let's go ahead and create a few simple tests.

Right click your package and select **New->JUnit Test Case**.



```
package jUnitTest;
```

Now go ahead and name your test and select **setUpBeforeClass(), setup(), tearDownAfterClass() and teardown().** Select finish.


New JUnit Test Case

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: demo

Package: jUnitTest

Name: MultiplicationTest

Superclass: java.lang.Object

Which method stubs would you like to create?

☑ setUpBeforeClass() ☑ tearDownAfterClass()

☑ setUp() ☑ tearDown()

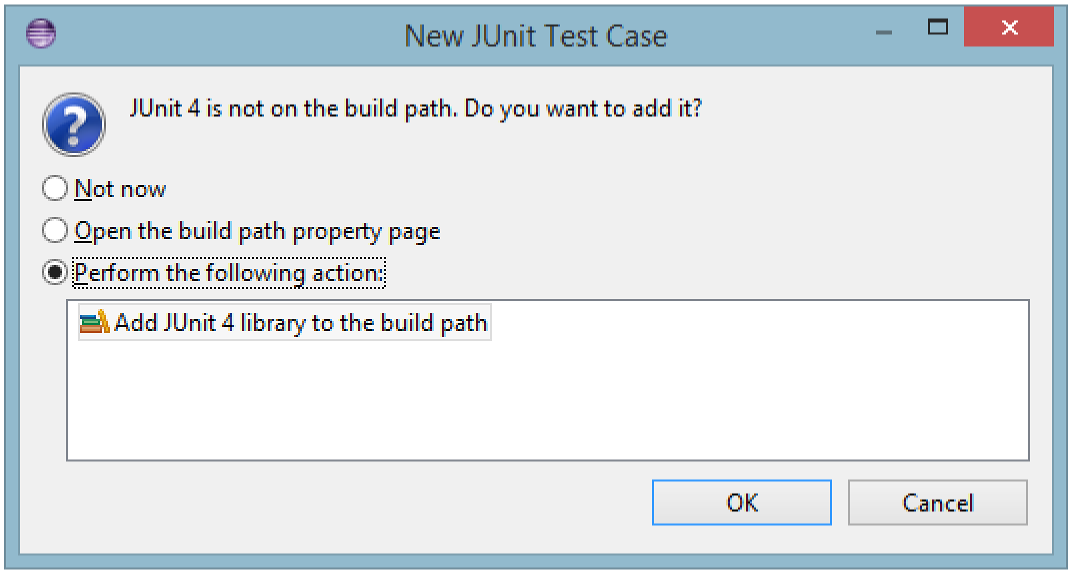
☐ constructor

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

☐ Generate comments

Class under test:

If prompted, select **OK.**



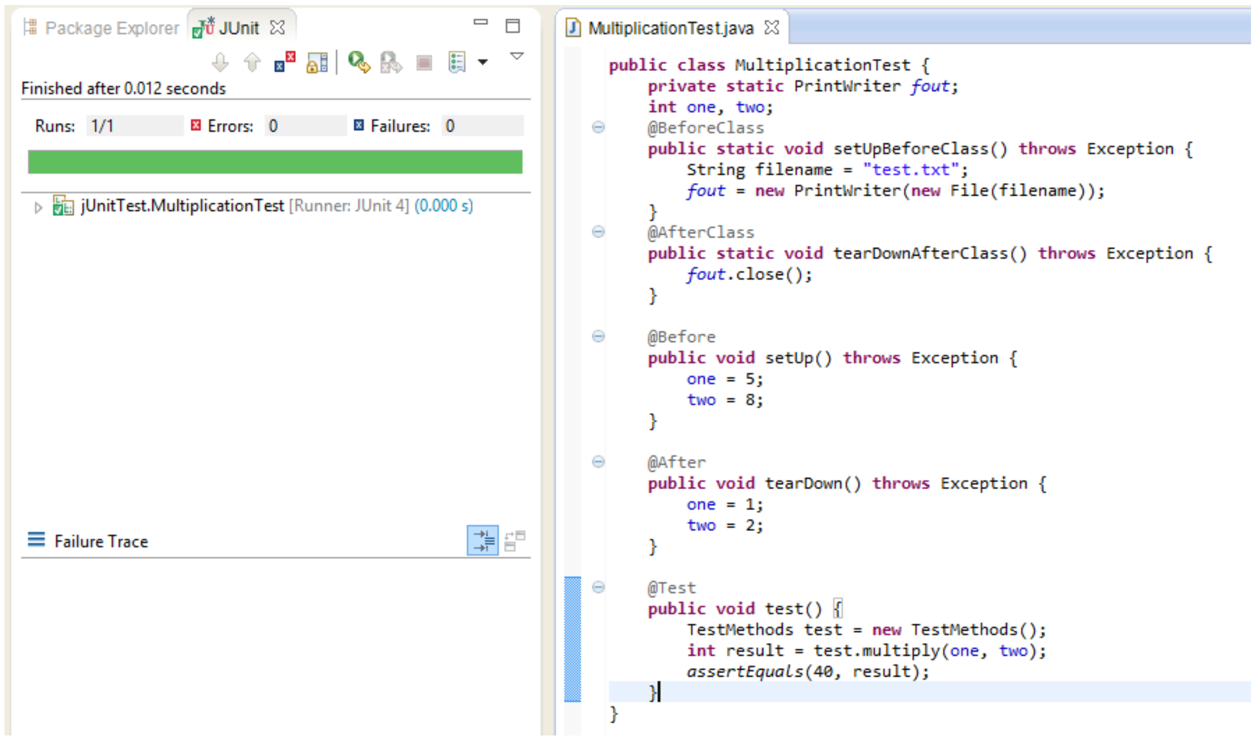
New JUnit Test Case

JUnit 4 is not on the build path. Do you want to add it?

Not now [✓ Perform the following action:] Open the build path property page

Add JUnit 4 library to the build path

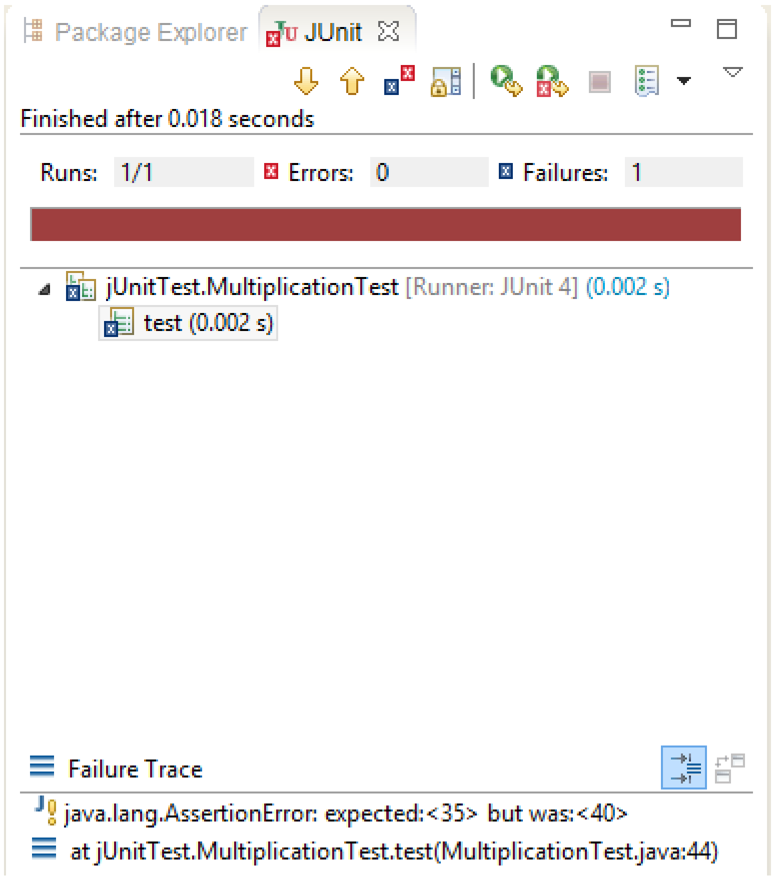
This here is a simple test for the multiplication method.



```
Finished after 0.012 seconds
Runs: 1/1    Errors: 0    Failures: 0
jUnitTest.MultiplicationTest [Runner: JUnit 4] (0.000 s)
```

```java
public class MultiplicationTest {
    private static PrintWriter fout;
    int one, two;
    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
        String filename = "test.txt";
        fout = new PrintWriter(new File(filename));
    }
    @AfterClass
    public static void tearDownAfterClass() throws Exception {
        fout.close();
    }

    @Before
    public void setUp() throws Exception {
        one = 5;
        two = 8;
    }

    @After
    public void tearDown() throws Exception {
        one = 1;
        two = 2;
    }

    @Test
    public void test() {
        TestMethods test = new TestMethods();
        int result = test.multiply(one, two);
        assertEquals(40, result);
    }
}
```

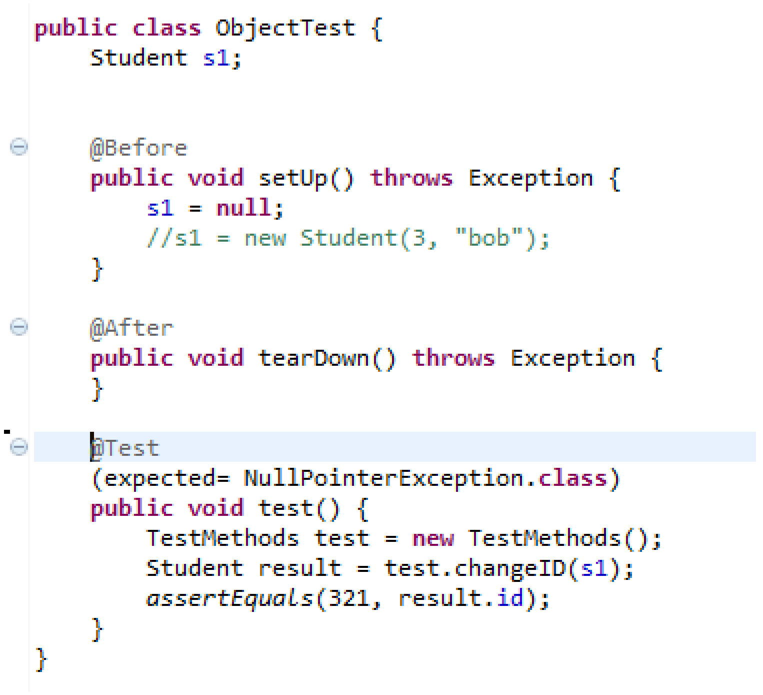
The **setUpBeforeClass()** method can be used do things such as opening a file or connecting to a database and the **tearDownAfterClass()** can be used to close the connection to those things. The **setUp()** and **tearDown()** methods are methods that execute before and after the test and you can use **setup()** to initialize variables amongst other things and use **teardown()** to clean up. In the test method is where you create your tests and run them. As you cans see the **Green** bar indicates that the test has passed. Whereas a **Red** bar indicates failure.



```
Finished after 0.018 seconds

Runs: 1/1    Errors: 0    Failures: 1

jUnitTest.MultiplicationTest [Runner: JUnit 4] (0.002 s)
    test (0.002 s)

Failure Trace
java.lang.AssertionError: expected:<35> but was:<40>
at jUnitTest.MultiplicationTest.test(MultiplicationTest.java:44)
```

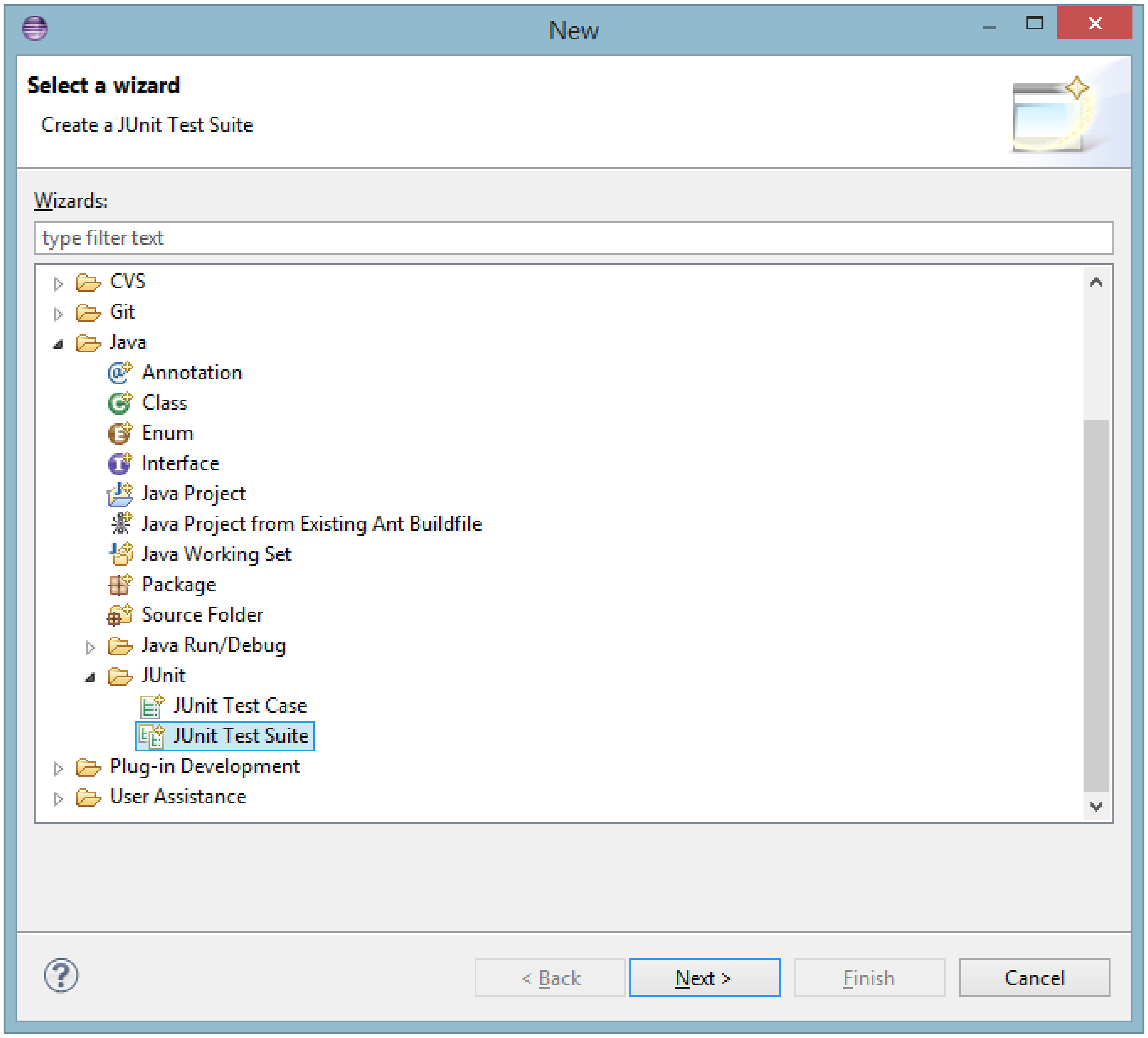
Now create another test using **setUp()** and **tearDown()** to test on the Student object.

This test passes in a null object and it will pass because we are expecting it to throw a NullPointerException.

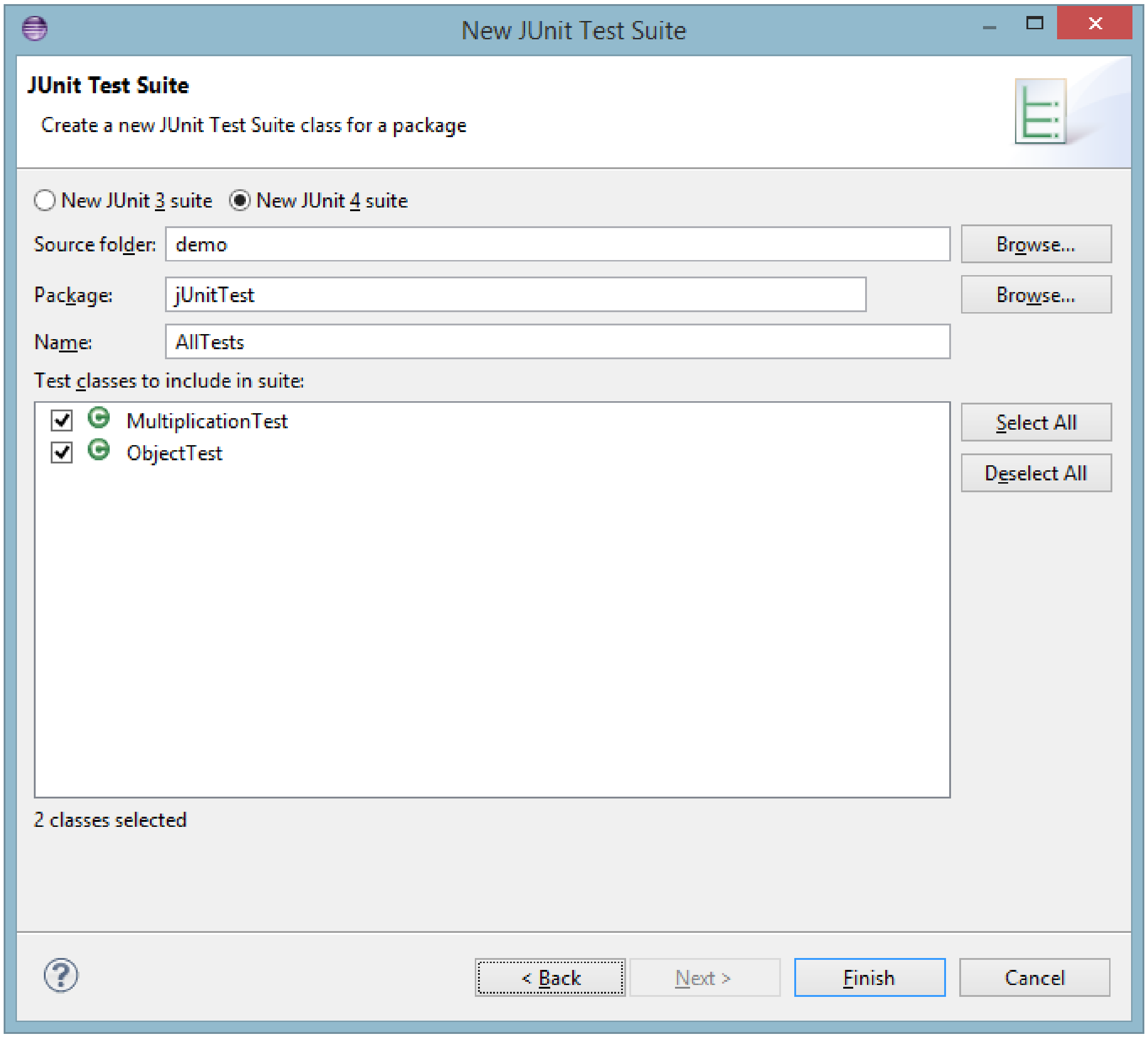


```
public class ObjectTest {
    Student s1;


    @Before
    public void setUp() throws Exception {
        s1 = null;
        //s1 = new Student(3, "bob");
    }

    @After
    public void tearDown() throws Exception {
    }

    @Test
    (expected= NullPointerException.class)
    public void test() {
        TestMethods test = new TestMethods();
        Student result = test.changeID(s1);
        assertEquals(321, result.id);
    }
}
```

Now we are going to take those two tests and put them into a test suite which will run both tests. Click the **New** button right below **File** and select the **JUnit Test Suite** as shown below.



Here you can select which classes to include in the test suite and then click **Finish**.


New JUnit Test Suite

JUnit Test Suite

Create a new JUnit Test Suite class for a package

New JUnit 3 suite [✓ New JUnit 4 suite]

Source folder: demo

Package: jUnitTest

Name: AllTests

Test classes to include in suite:

- ☑ MultiplicationTest
- ☑ ObjectTest

2 classes selected

Now you can run the test suite!