Lab jUnit

3. Using jUnit - from http://www.vogella.com/articles/JUnit/article.html

3.1. Preparation

Create a new project `LabJUnit`. We want to create the unit tests in a separate folder. The creation of a separate folder for tests is not mandatory. But it is a good practice to keep the code separated from the regular code. You might even create a separate project for the test classes, but we skip this step to make this example simpler.

Create a new source folder `test` by right-clicking on a project and selecting New → Source Folder.

3.2. Create a Java class

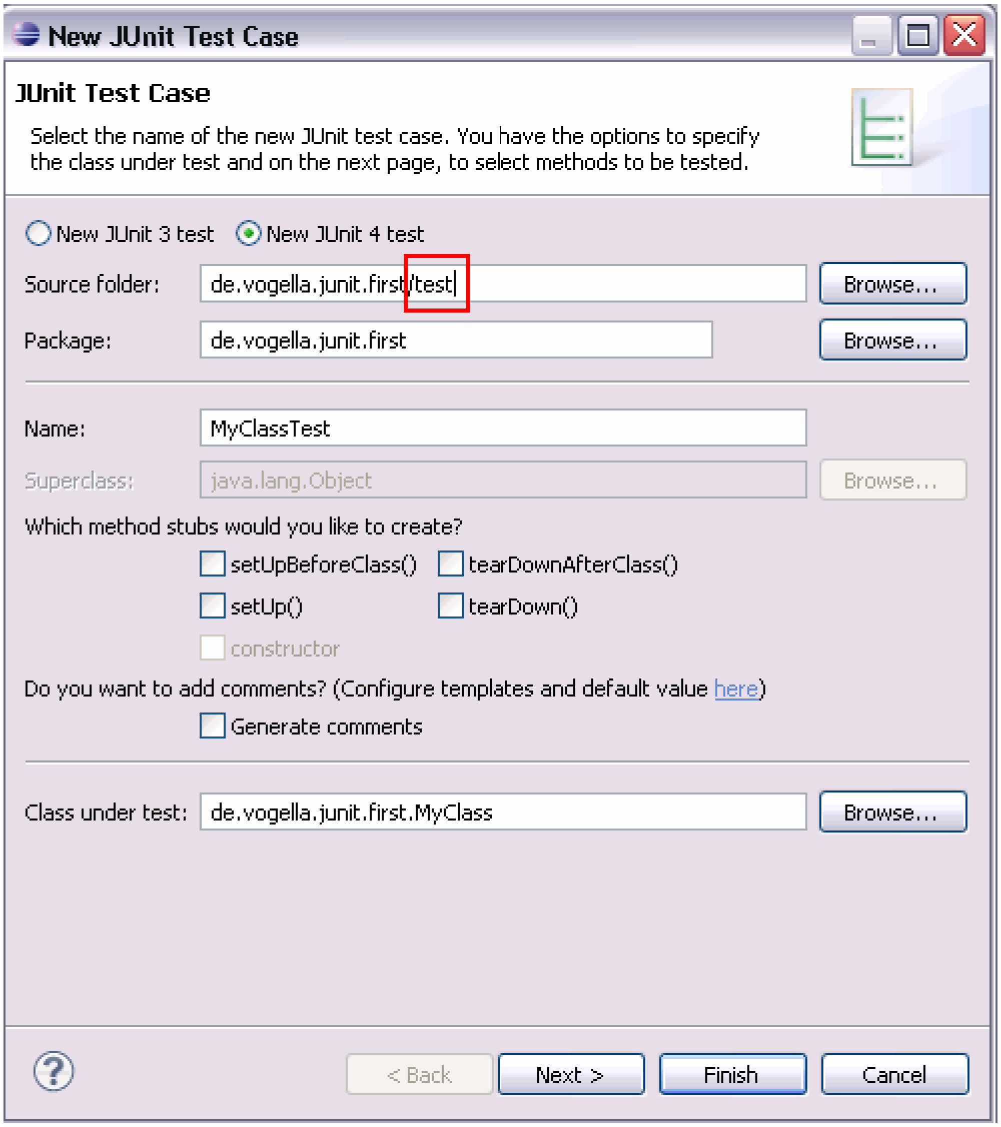
In the "src" folder, create the `labJUnit` package and the following class.

```
package labJUnit


public class MyClass {

	public int multiply(int x, int y) {

		return x / y;

	}

}
```

3.3. Create a JUnit test

Right click on your new class in the Package Explorer and select New → JUnit Test Case. Select "New JUnit 4 test" and set the source folder to "test", so that your test class gets created in this folder.

Note: The folder and package names throughout the tutorial are different because I changed them



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: de.vogella.junit.first/test

Package: de.vogella.junit.first

Name: MyClassTest

Superclass: java.lang.Object

Which method stubs would you like to create?

□ setUpBeforeClass() □ tearDownAfterClass()

□ setUp() □ tearDown()

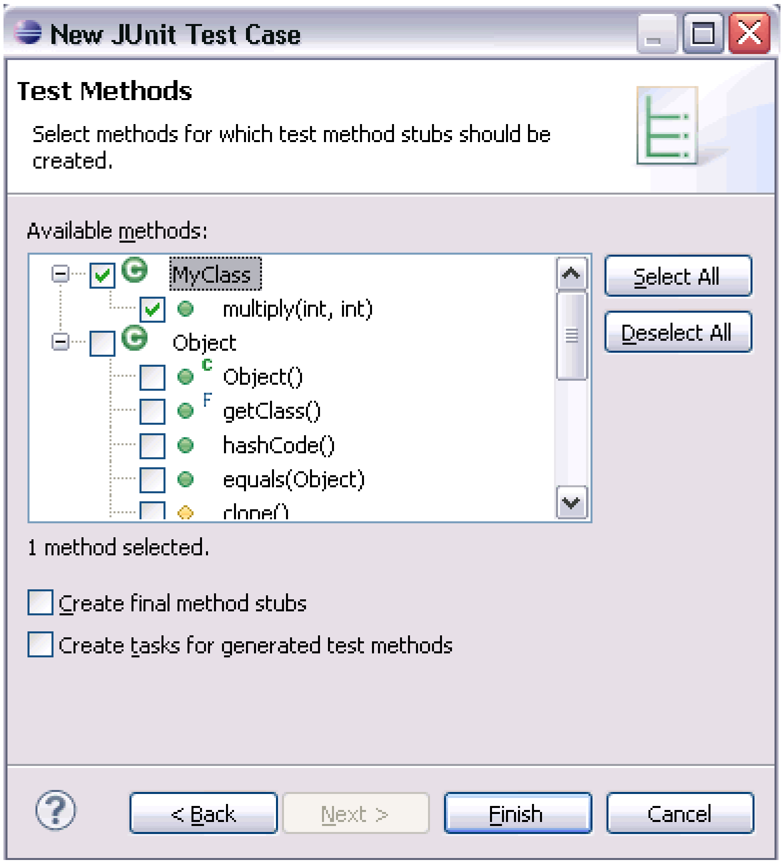
□ constructor

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

□ Generate comments

Class under test: de.vogella.junit.first.MyClass

Press "Next" and select the methods which you want to test.



Test Methods

Select methods for which test method stubs should be created.

Available methods:

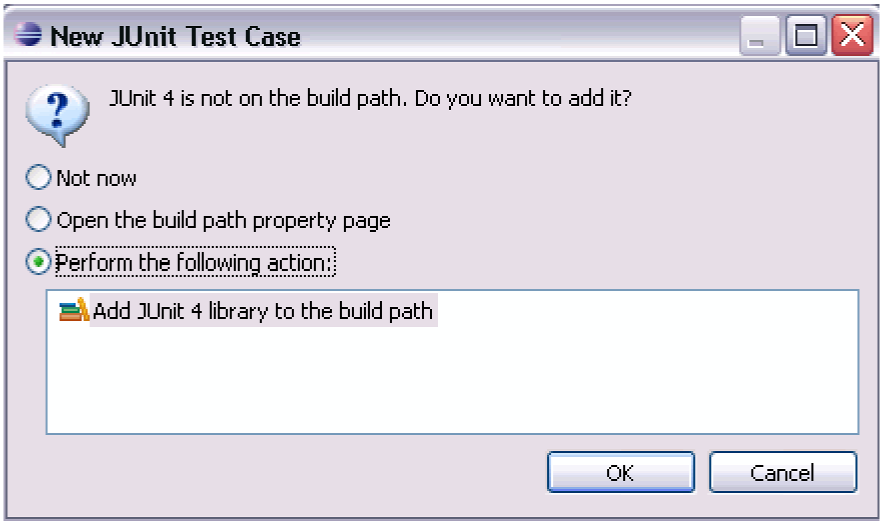
- ☑ MyClass
  - ☑ multiply(int, int)
- □ Object
  - □ Object()
  - □ getClass()
  - □ hashCode()
  - □ equals(Object)
  - □ clone()

1 method selected.

□ Create final method stubs

□ Create tasks for generated test methods

If the JUnit library in not part of your classpath, Eclipse will prompt you to do so.



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

○ Not now

○ Open the build path property page

● Perform the following action:

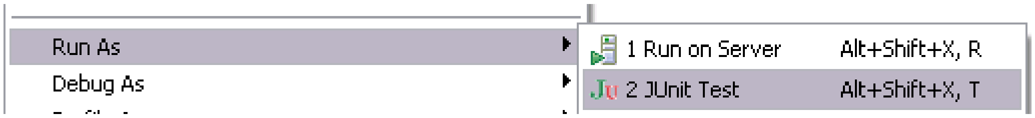
Add JUnit 4 library to the build path

Create a test with the following code.

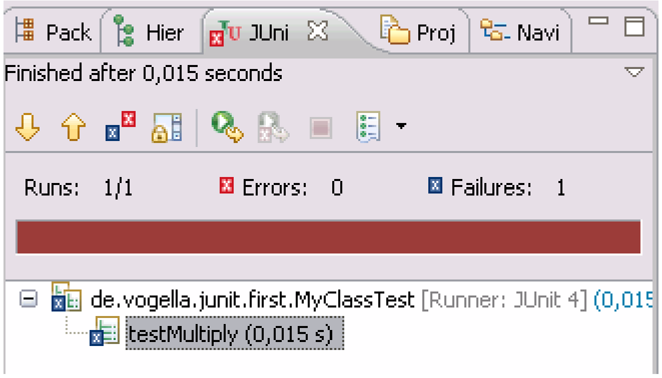
```
package de.vogella.junit.first;



import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class MyClassTest {

    @Test
    public void testMultiply() {
        MyClass tester = new MyClass();
        assertEquals("Result", 50, tester.multiply(10, 5));
    }
}
```

3.4. Run your test via Eclipse

Right click on your new test class and select Run-As → JUnit Test.



The result of the tests will be displayed in the JUnit `View`.



Finished after 0,015 seconds

Runs: 1/1 Errors: 0 Failures: 1

de.vogella.junit.first.MyClassTest [Runner: JUnit 4] (0,015

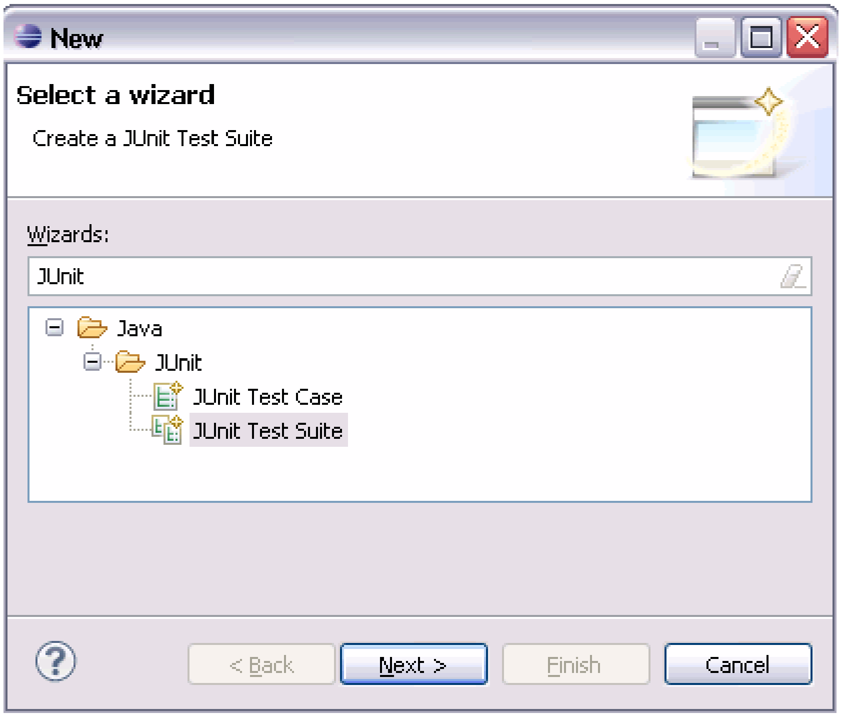
testMultiply (0,015 s)

The test should be failing (indicated via a red bar).

This is because our multiplier class is currently not working correctly (it does a division instead of multiplication). Fix the bug and re-run test to get a green bar.

If you have several tests you can combine them into a test suite. Running a test suite will execute all tests in that suite.

To create a test suite, select your test classes → right click on it → New → Other → JUnit → Test Suite.



Select a wizard

Create a JUnit Test Suite

Wizards: JUnit

Select "Next" and select the methods for which you want to create a test.

Change the code to the following to make your test suite run your test. If you develop another test later you can add it to `@Suite.SuiteClasses`.

```java
package mypackage;


import org.junit.runner.RunWith;
import org.junit.runners.Suite;


@RunWith(Suite.class)
@Suite.SuiteClasses({ MyClassTest.class })
public class AllTests {
}
```

3.5. Run your test via code

You can also run your tests from via your own code. The class `org.junit.runner.JUnitCore` provides the method runClasses() which allows you to run one or several tests classes. As a return parameter you receive an object of the type `org.junit.runner.Result`. This object can be used to retrieve information about the tests.

In your "test" folder create a new class `MyTestRunner` with the following code. This class will execute your test class and write potential failures to the console.

```java
package de.vogella.junit.first;

import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class MyTestRunner {
       public static void main(String[] args) {
              Result result = JUnitCore.runClasses(MyClassTest.class);
              for (Failure failure : result.getFailures()) {
                     System.out.println(failure.toString());
              }
       }
}
```

Last but not least modify the main method so that it prints the number of tests run, the number of tests passed and the number of tests that failed. (labeled)