

SWE 434

Software Testing and Validation

Lecture slides are available on following website through your university student login

<https://lms.ksu.edu.sa/>

Raja Majid Mehmood

rmehmood@ksu.edu.sa

Department of Software Engineering,
King Saud University, Riyadh, Saudi Arabia.

Lecture's Agenda

- Brief discussion about programming (Java)
- Introduction to IDE (Eclipse)
- Introduction to Software Testing Framework (JUnit)
- **Quiz-1**

Java Program

```
1. package lab1;

2. public class MyMath
3. {
4.     public static void main( String[] args )
5.     {
6.         int result = MyMath.div( 6, 2 );
7.         System.out.println( result );
8.     }

9.     public static int div( int a, int b )
10.    {
11.        return a / b;
12.    }
13.}
```

Integrated Development Environment (IDE)

- Eclipse downloads are available on <http://www.eclipse.org/downloads/packages/eclipse-classic-372/indigosr2>
















The screenshot shows the Eclipse Classic 3.7.2 download page. The browser address bar displays www.eclipse.org/downloads/packages/eclipse-classic-372/indigosr2. The page features the Eclipse logo and a navigation menu with links: Home, Downloads, Users, Members, Committers, Resources, Projects, and About Us. A search bar is also present.

On the left sidebar, under "Downloads Home", the "Indigo Packages" link is highlighted with a red box. The main content area has a red box around the "Eclipse Classic 3.7.2" title. Below the title, the "Package Details" section describes the classic Eclipse download, including the Eclipse Platform, Java Development Tools, and Plug-in Development Environment. The "Feature List" section is also visible.

On the right side, the "Download Links" section is highlighted with a red box, listing various operating system and architecture combinations: Windows 32-bit, Windows 64-bit, Mac OS X(Carbon), Mac OS X(Cocoa 32), Mac OS X(Cocoa 64), Linux 32-bit, and Linux 64-bit. Below this, there are links for "Checksums...", "Bugzilla", "Open Bugs: 0", "Resolved Bugs: 0", and "File a Bug on this Package".

Java Runtime Environment(JRE 7)

- If JRE is not installed in system, then download it from
 - <http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html>

Java SE Runtime Environment 7u13		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
<input type="radio"/> Accept License Agreement <input checked="" type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux x86	54.7 MB	 jre-7u13-linux-i586.rpm
Linux x86	45.9 MB	 jre-7u13-linux-i586.tar.gz
Linux x64	52.84 MB	 jre-7u13-linux-x64.rpm
Linux x64	44.63 MB	 jre-7u13-linux-x64.tar.gz
Mac OS X x64	50.32 MB	 jre-7u13-macosx-x64.dmg
Mac OS X x64	46.66 MB	 jre-7u13-macosx-x64.tar.gz
Solaris x86	45.41 MB	 jre-7u13-solaris-i586.tar.gz
Solaris x64	14.8 MB	 jre-7u13-solaris-x64.tar.gz
Solaris SPARC	48.7 MB	 jre-7u13-solaris-sparc.tar.gz
Solaris SPARC 64-bit	17.4 MB	 jre-7u13-solaris-sparcv9.tar.gz
Windows x86 Online	0.86 MB	 jre-7u13-windows-i586-iftw.exe
Windows x86 Offline	30.05 MB	 jre-7u13-windows-i586.exe
Windows x86	39.77 MB	 jre-7u13-windows-i586.tar.gz
Windows x64	31.47 MB	 jre-7u13-windows-x64.exe
Windows x64	41.49 MB	 jre-7u13-windows-x64.tar.gz

- **Download** the relevant installation package, and
- Do the JRE 7 **installation!**

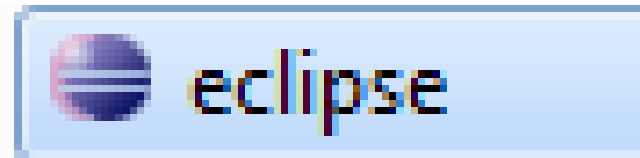
Integrated Development Environment (IDE)

- Eclipse

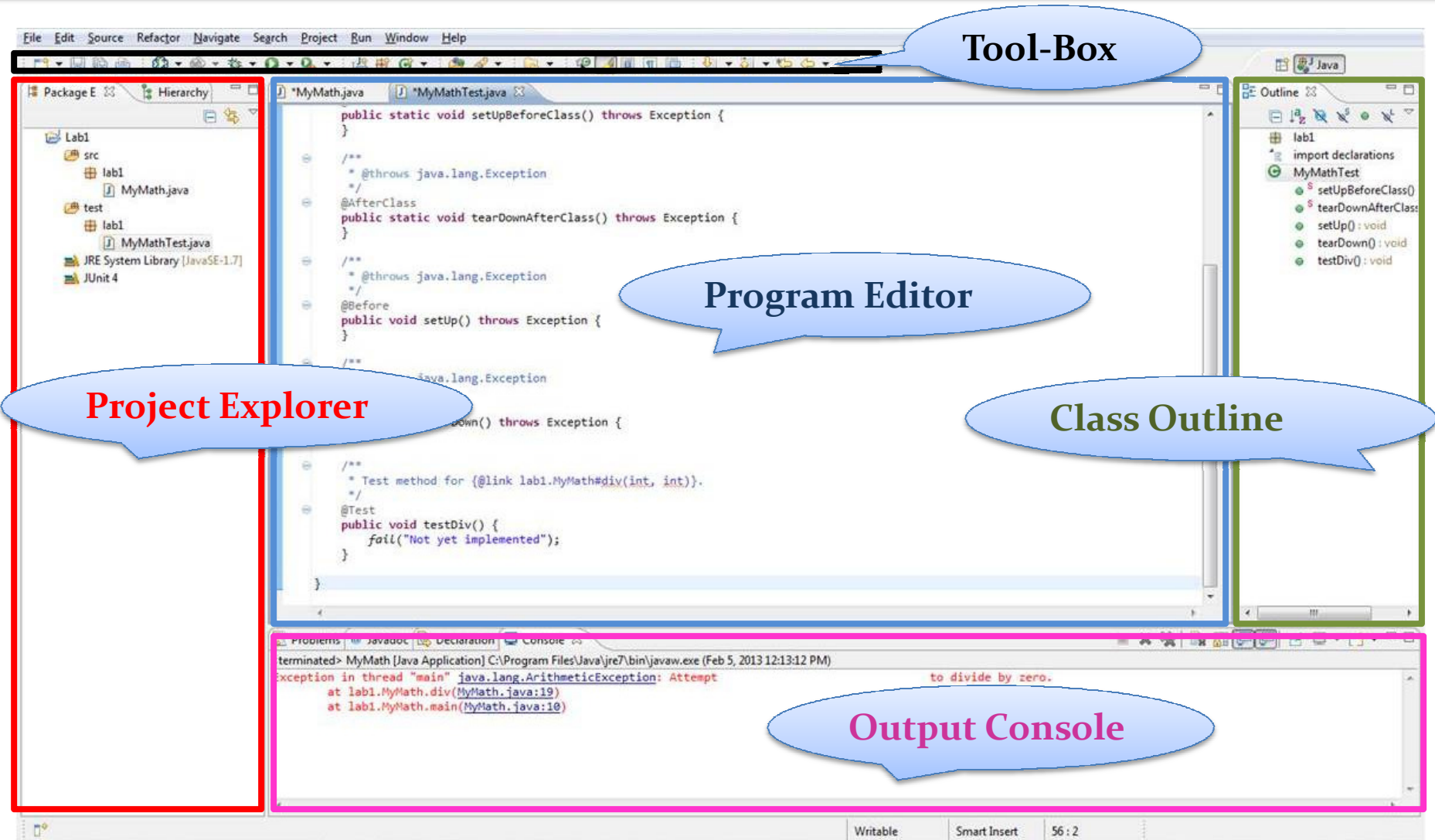
- It should be in following folder **eclipse-SDK-3.7.2-win32**

Name	Date modified	Type	Size
configuration	2/19/2011 12:09 AM	File folder	
features	10/10/2012 7:45 PM	File folder	
plugins	10/10/2012 7:45 PM	File folder	
readme	10/13/2010 5:44 AM	File folder	
workspace	2/19/2011 12:15 AM	File folder	
.eclipseproduct	10/23/2007 8:43 PM	ECLIPSEPRODUCT...	1 KB
eclipse	10/23/2007 8:43 PM	Application	56 KB
eclipse	10/22/2012 6:11 PM	Configuration sett...	1 KB
eclipsec	10/23/2007 8:43 PM	Application	28 KB
epl-v10	10/23/2007 8:43 PM	Chrome HTML Do...	17 KB

- Please open the **eclipse.exe**



Integrated Development Environment (IDE)



Software Testing

The **process** of executing a **program** with the intent of finding errors.

Myers, 1979

Unit Testing

- The process of validating a piece of code (*function* or *method*) with required input and expected output.



Test Case and Test Execution System

- What we need to do for automated testing?
 1. Test Case

“A test case is a small unit of code that tests a specific method”

 - **INPUT:** Actions send to System Under Test (**SUT**).
 - **OUTPUT:** Responses expected from **SUT**.
 - **VALIDATION:** To Check the *Test*, was successful or not?
 2. Test Execution System
 - Mechanism to read test scripts, and connect test cases to **System Under Test**, for example **JUNIT**.
 - Keeps track of test results.

Software Testing Process

System Under Test (SUT)

```
public class MyMath
{
    public static void main( String[] args )
    {}

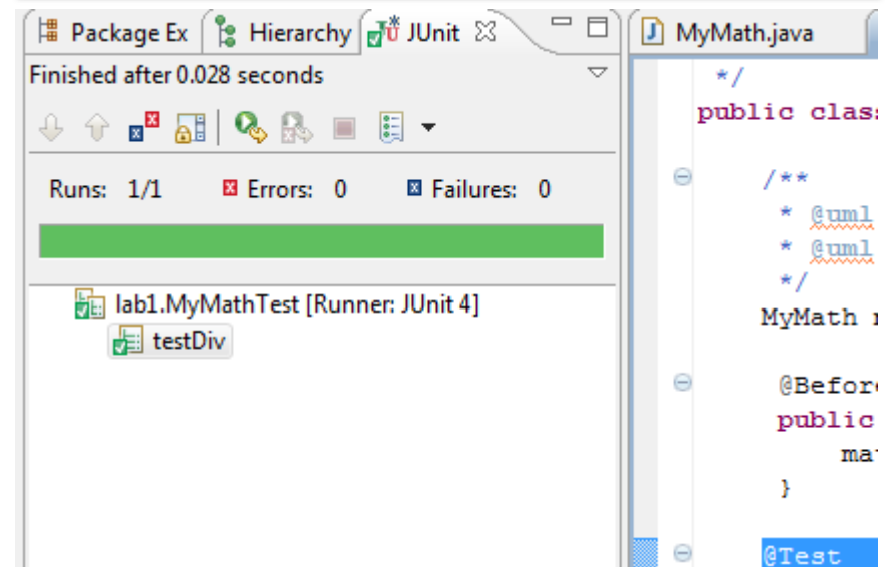
    public int div( int a, int b )
    {}
}
```

UNIT

Test Case

```
@Test
public void testDiv() {
    int actual=MyMath.div(6, 2);
    int expected=3;
    Assert.assertEquals(expected, actual);
}
```

Test Execution System (JUNIT)



Actions send to system under test (SUT).

Responses expected from SUT.

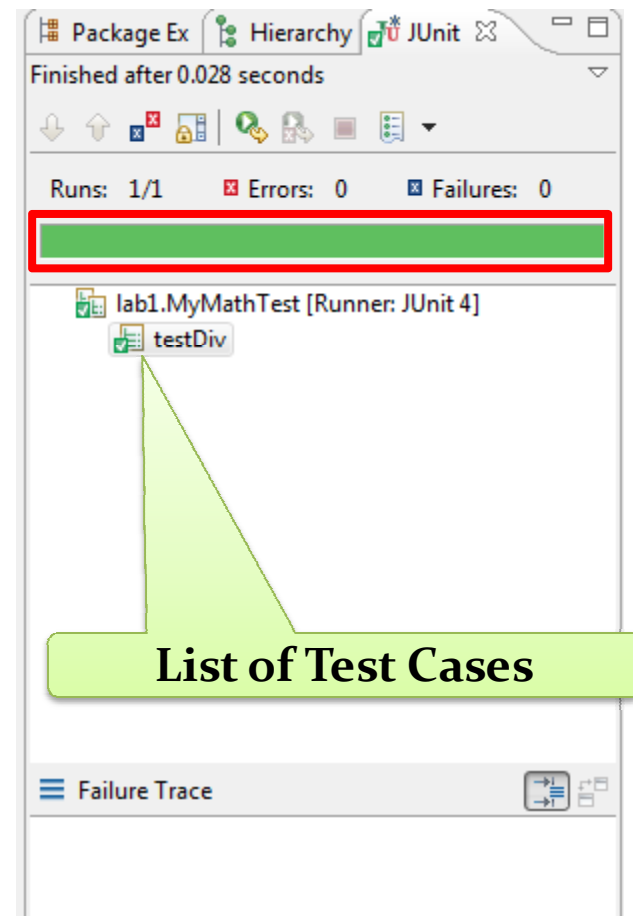
Determine whether a test was successful or not?

Test case verdicts

- A **verdict** is the declared result of executing a single test.
 - We can get the verdict after the assertion statement
- **Pass**: the test case achieved its intended purpose, and the software under test performed as expected.
- **Fail**: the test case achieved its intended purpose, but the software under test did not perform as expected.
- **Error**: the test case did not achieve its intended purpose.
 - Potential reasons:
 - An unexpected event occurred during the test case.
 - The test case could not be set up properly

JUnit

- **A Unit Testing Framework for Java**
 - It helps us to test the specific method of class
 - **Authors:** Erich Gamma, Kent Beck
 - <http://www.junit.org> or
 - <http://sourceforge.net/projects/junit/>
 - Plug-in is already installed in **Eclipse**.
 - **Recommended Version:** 4.3.1+



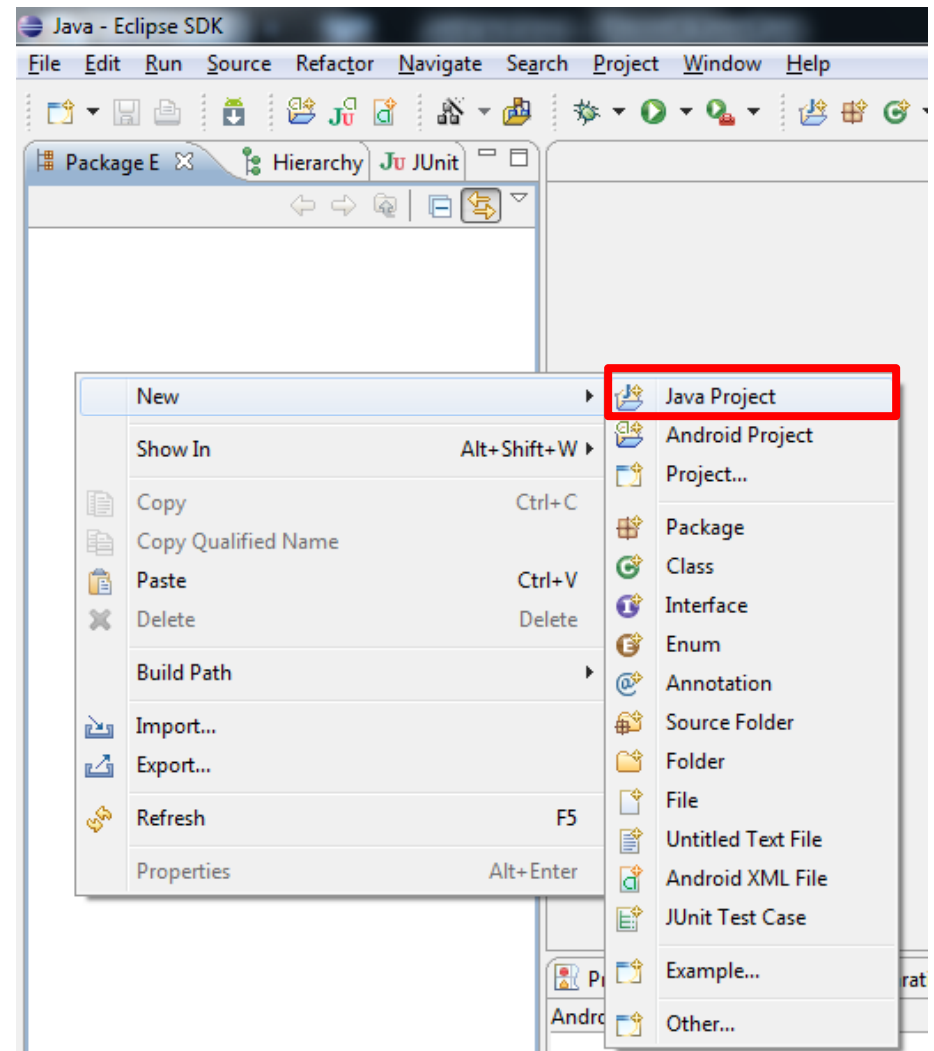
Failure console of JUnit

Practical Lab (Example-1)

- Problem Description:
 1. Implement the method “**div(a, b)**” for division of two integer numbers.
 2. Implement the *Test Case* by using the JUnit framework.
- Objective:
 - Creating the new java project in Eclipse through wizard
 - Creating the new class (SUT) and its implementation
 - Creating the test case class in JUnit through wizard
 - Implementing the test case method
- Create a new Java project in **Eclipse**.
 - *When creating the project, be sure the following is set up in the new project wizard:*

Create a New Java Project

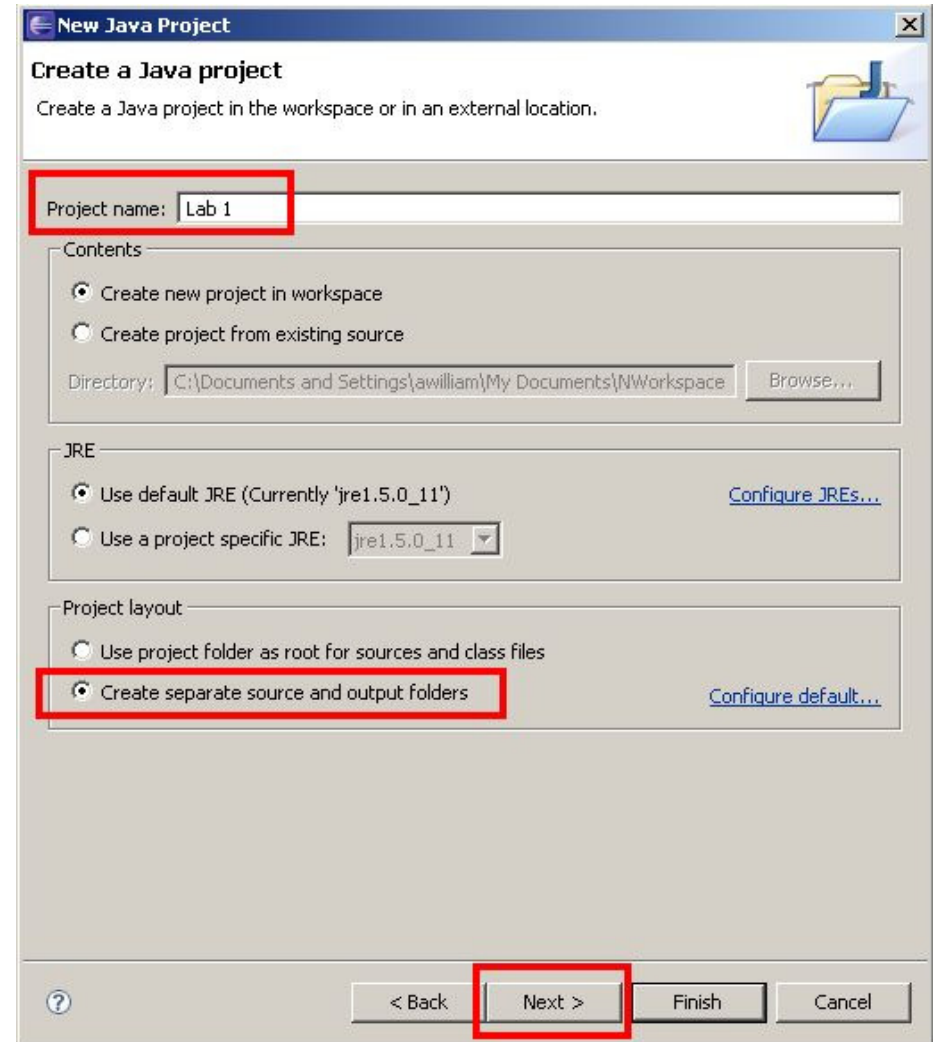
- Right click anywhere in package explorer, or go to in File menu.



Step-1

Create a Java Project

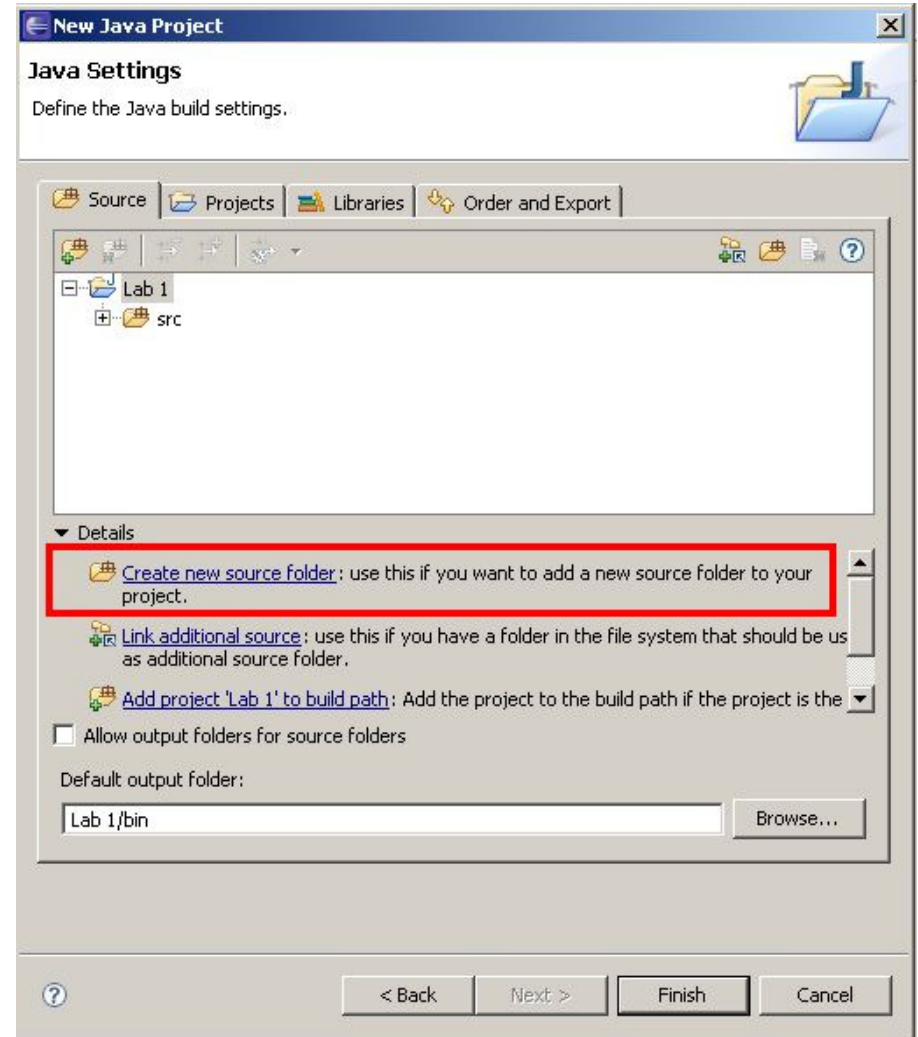
- On the “Create a Java Project” screen
 - Enter a project name,
 - Select the option “Create separate source and output folders.”
 - The exact version of the JRE is not important, but it should be version 1.5.0 or greater.
 - Then, click Next.



Step-1.1

Java Settings

- On the “Java Settings” screen, click on “Create a new source folder”.



Step-1.2

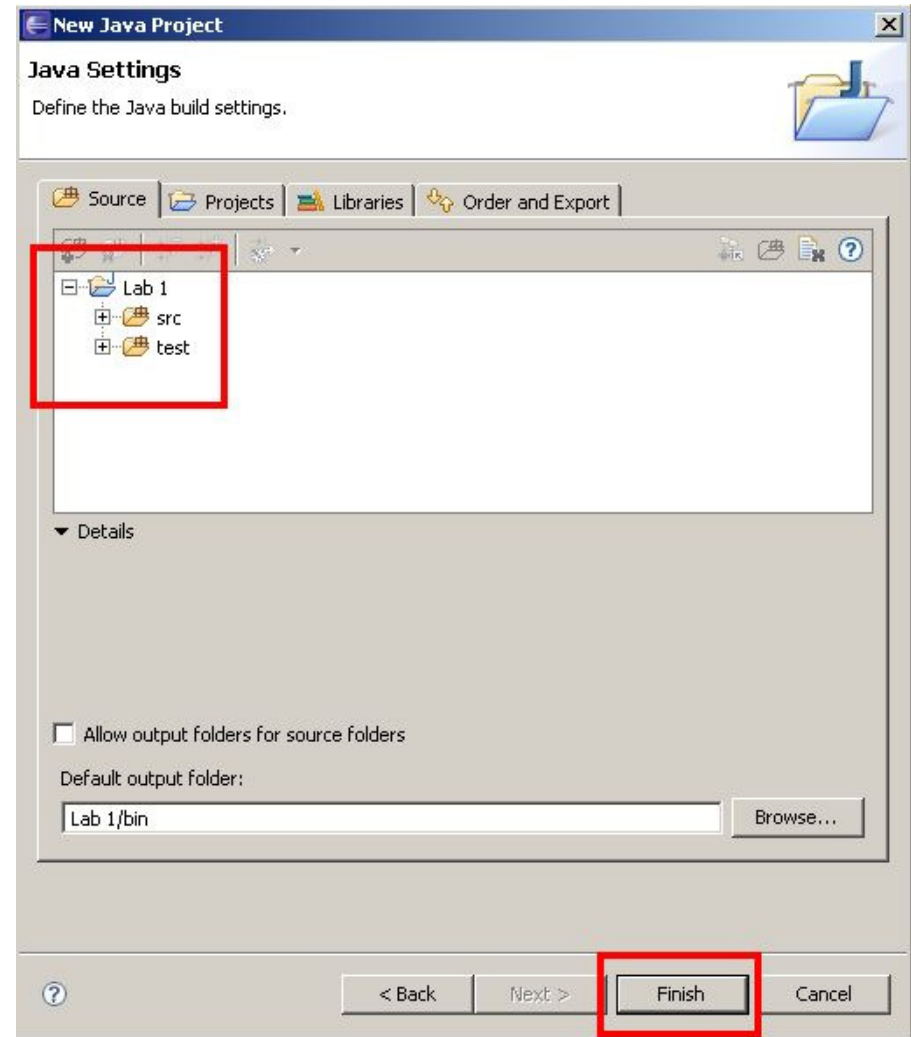
New Source Folder

- Name the new source folder **test**. Click Finish.



Step-1 Completed

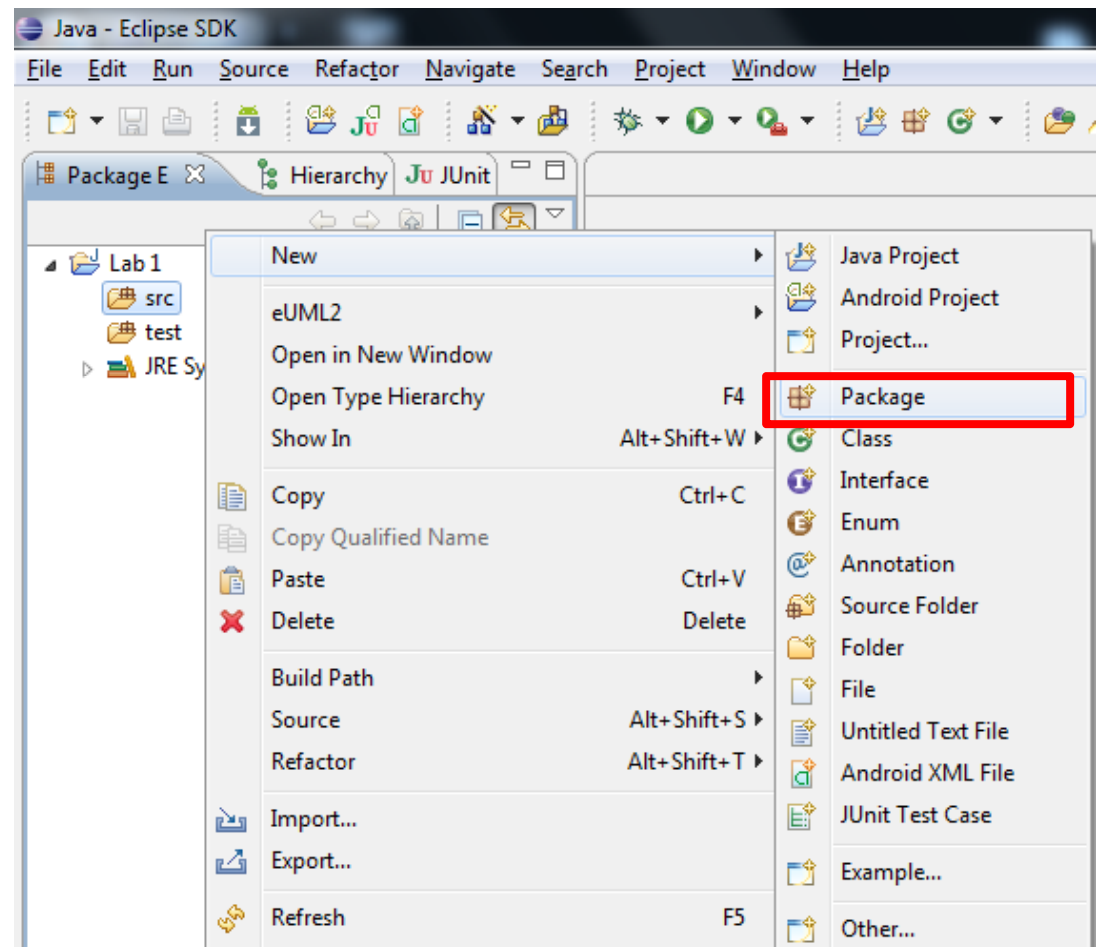
- The result should be two source folders, **src** and **test**.
- We are going to store the code to be tested in the **src** folder,
- and the test cases in the **test** folder.



Step-2

Create a Package in src

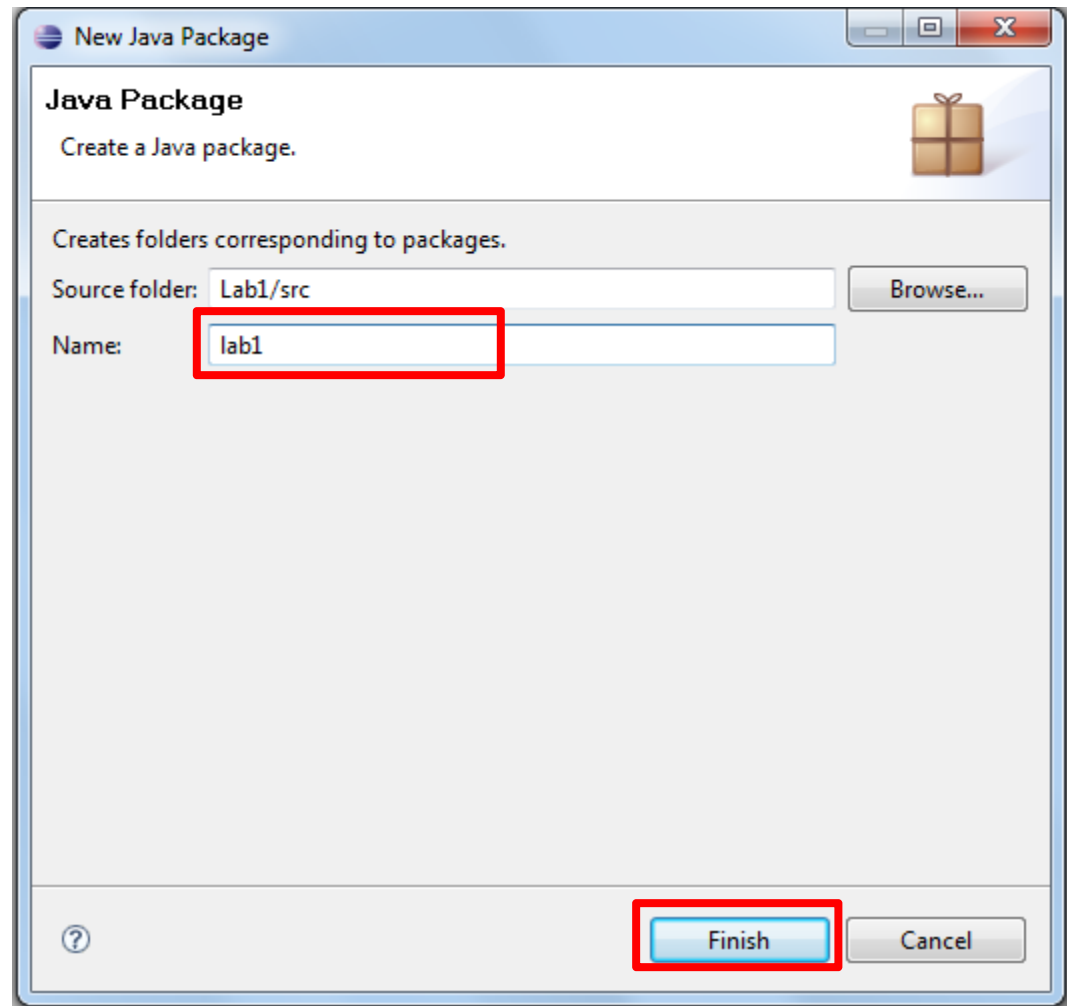
- Create package **lab1** in **src** folder,
- Right Click on **src**,
- see the steps in this figure,



Step-2

Create a Package in src

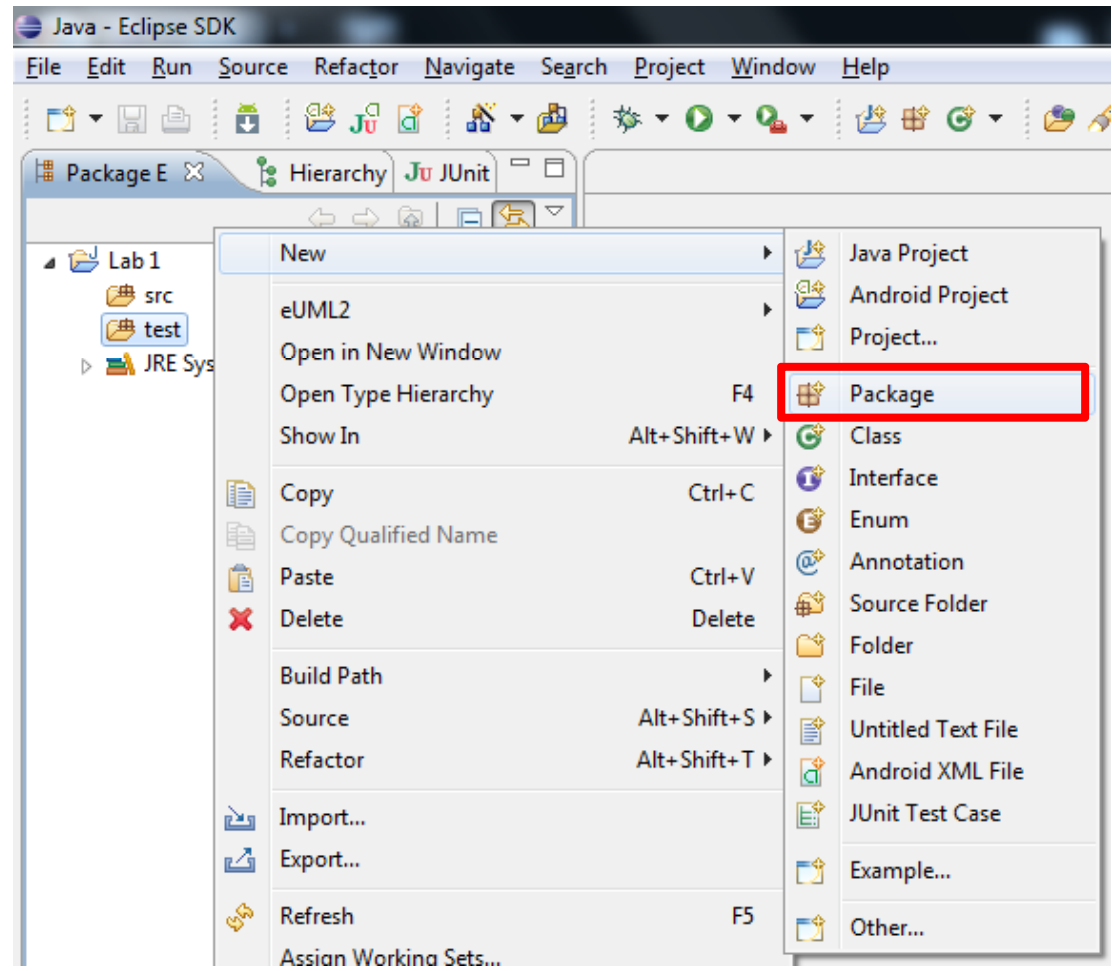
- Enter **lab1** in Name field.
- Press **Finish** Button



Step-2

Create a Package in test

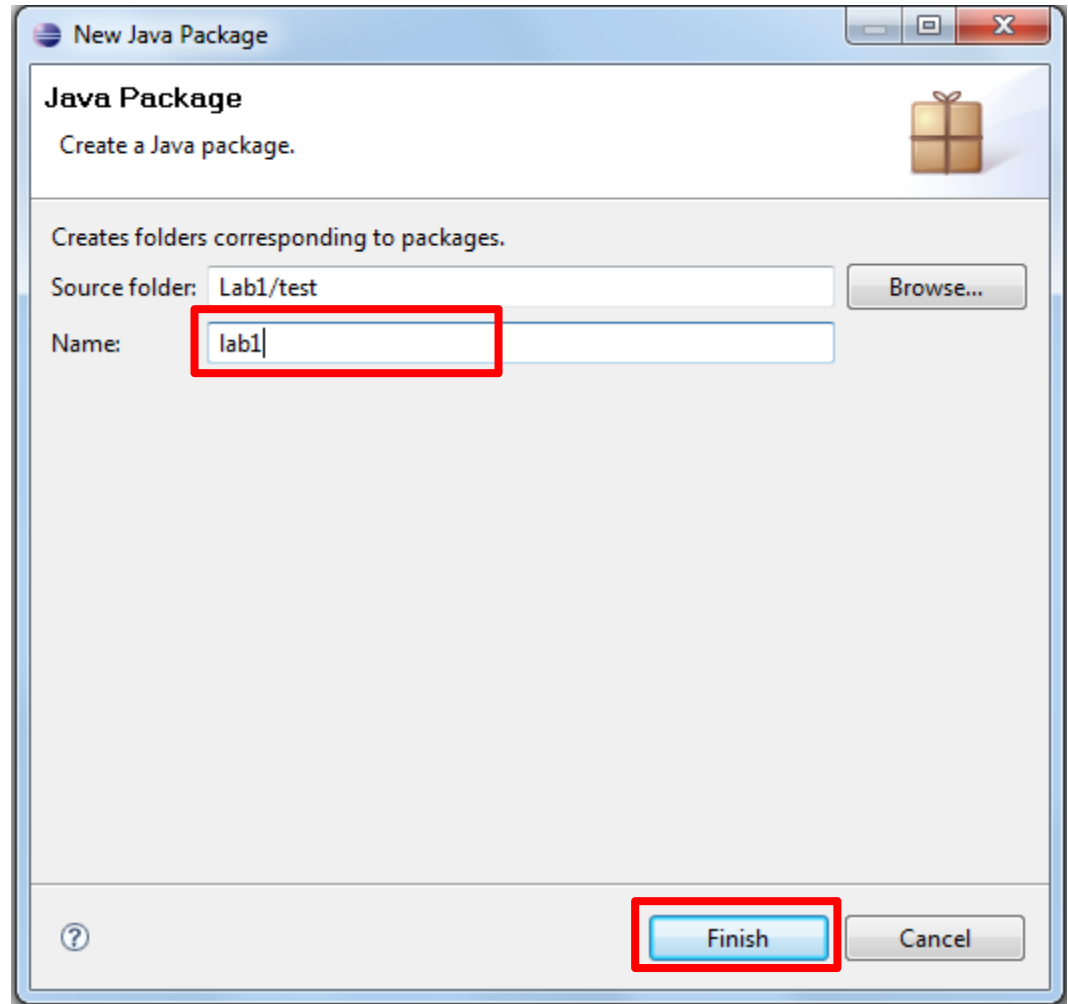
- Create package **lab1** in **test** folder,
- Right Click on **test**,
- see the steps in this figure,



Step-2

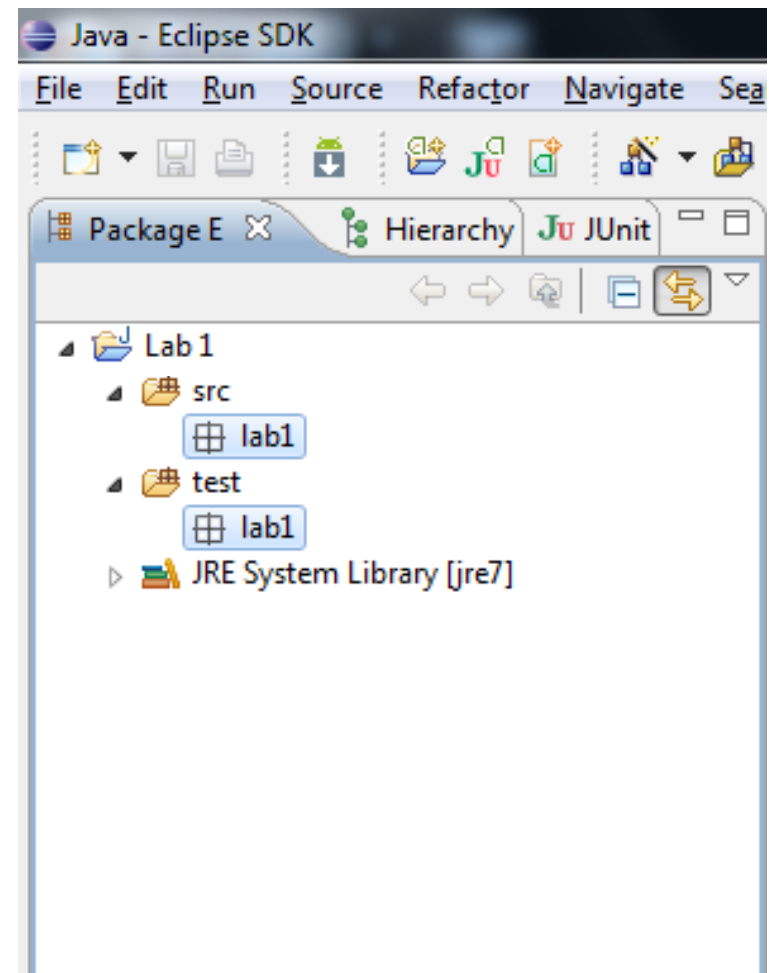
Create a Package in test

- Enter **lab1** in Name field.
- Press **Finish** Button



Package View

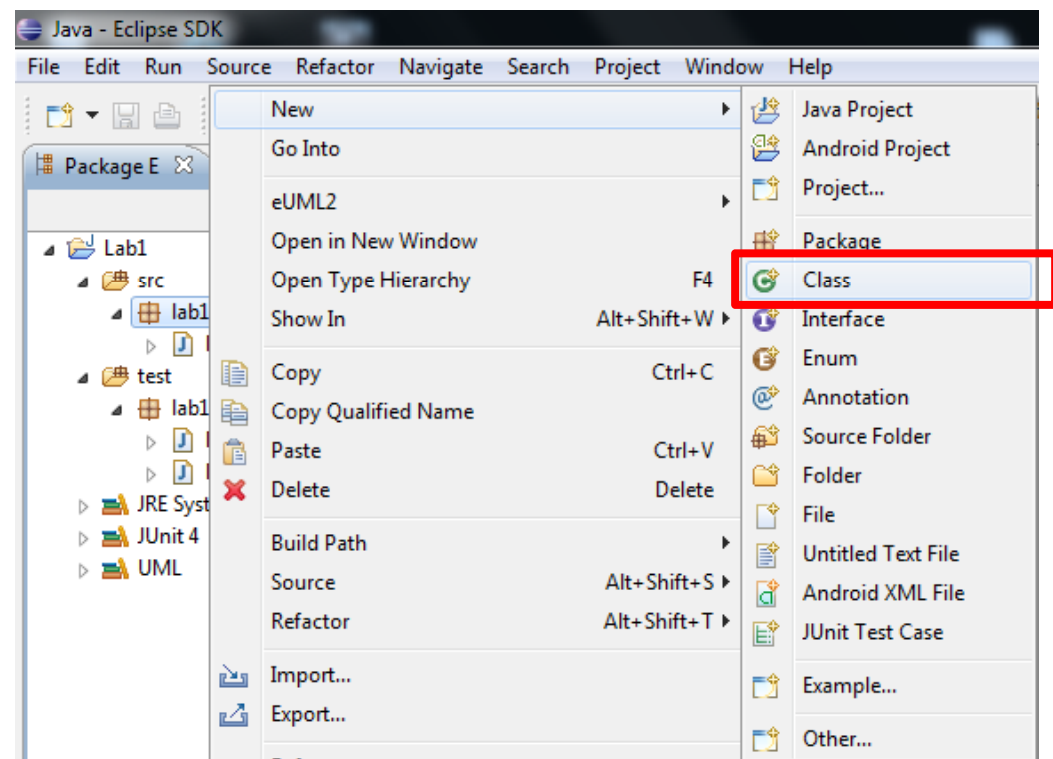
- Now, we already created the package **lab1** in both **src** and **test** folders.



Step-3

Create a Class to Test

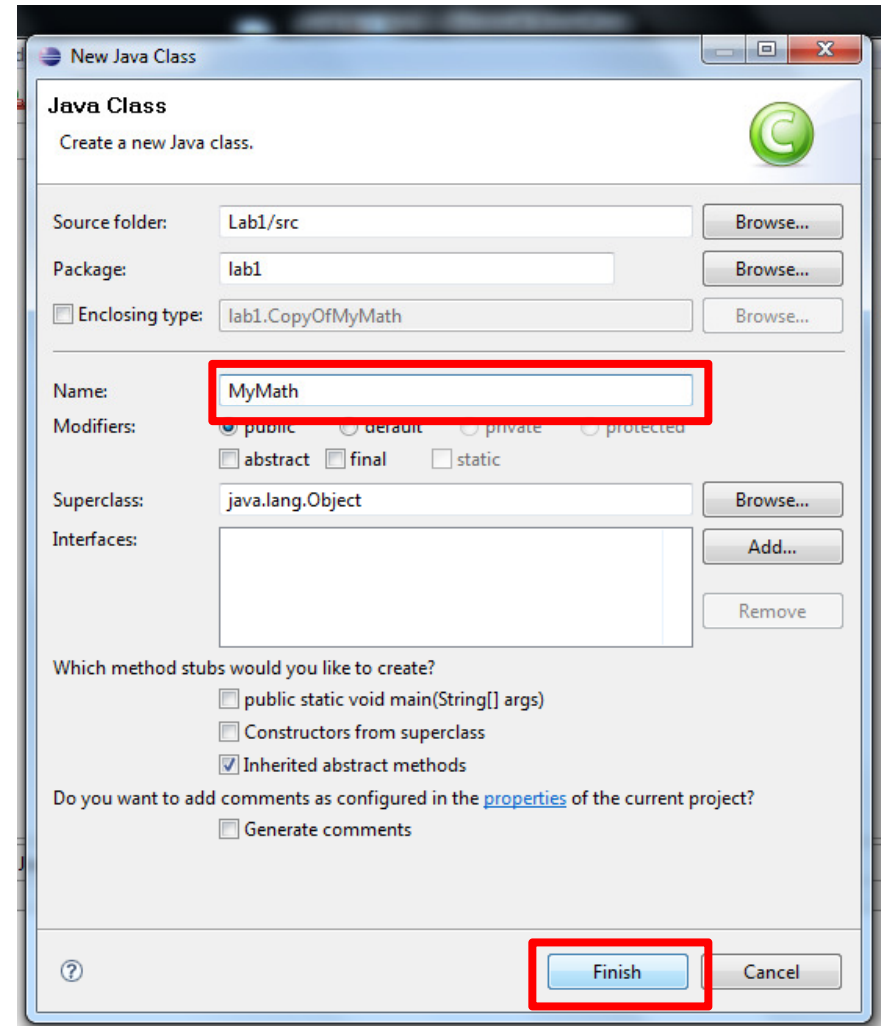
- *Right click on lab1 under src folder, and*
- **Click the Class**



Step-3

Create a Class to Test

- Enter class name **MyMath**
- Press **Finish**



Step-3

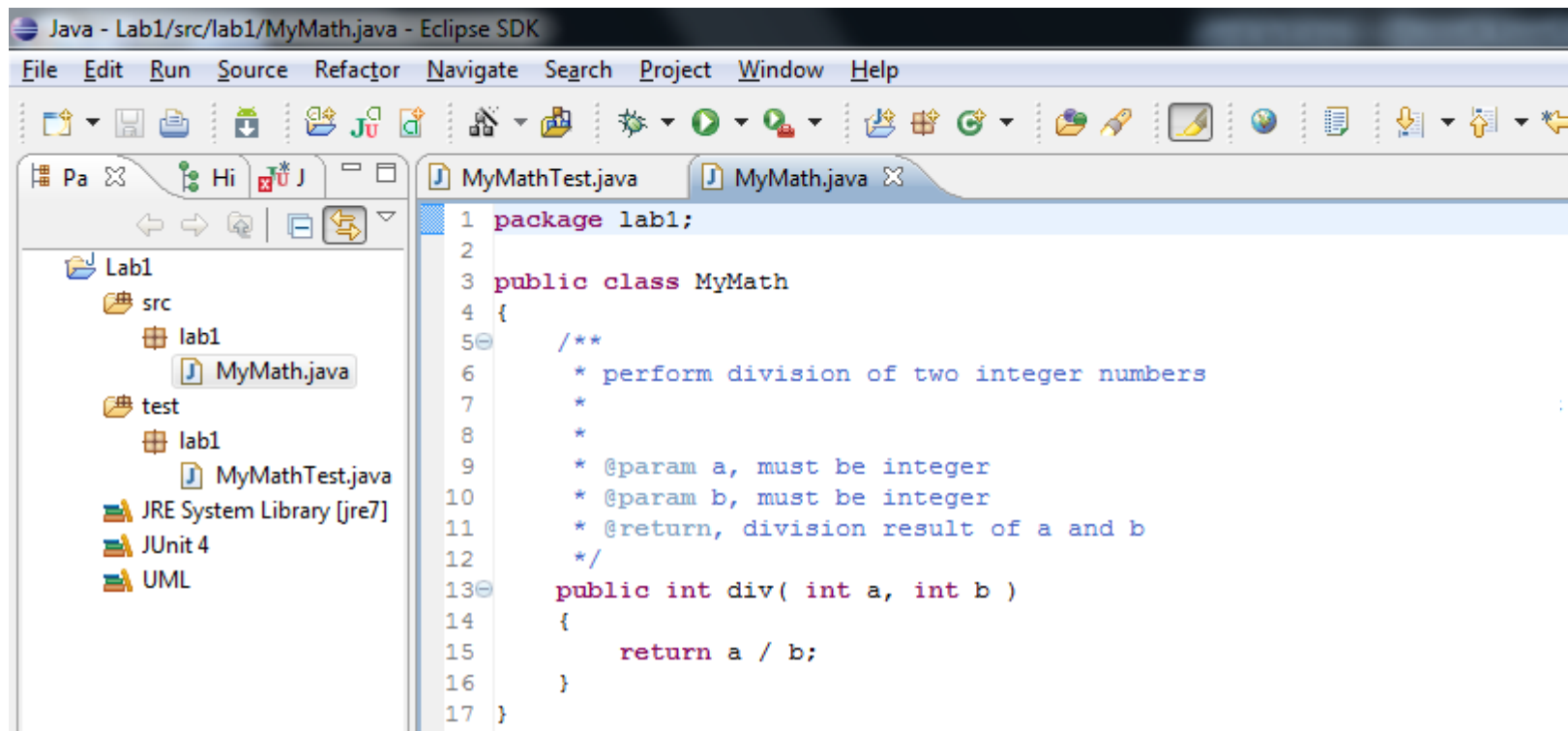
Create a Class to Test

- Add the following code in class **MyMath** within the **lab1** package of the **src** folder.

```
1.package lab1;  
  
2.public class MyMath  
3.{  
4.    public int div( int a, int b )  
5.    {  
6.        return a / b;  
7.    }  
8. }
```

- The class has one method, **div(int, int)**, which performs some computation.

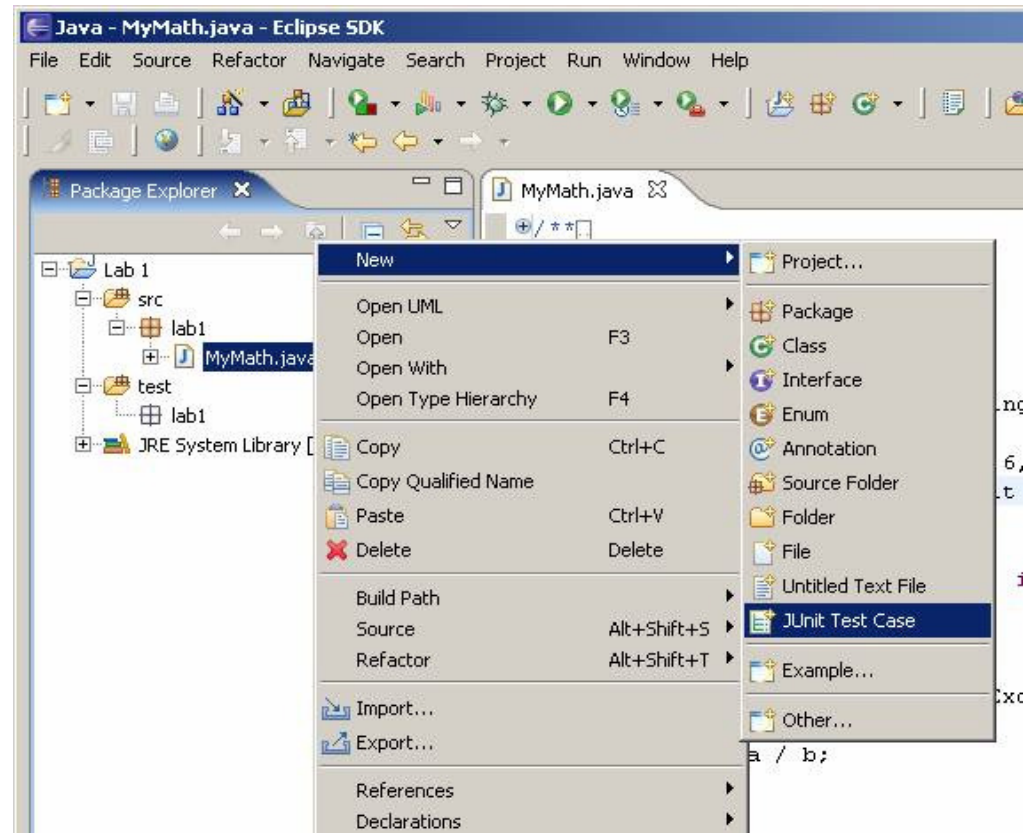
Class View (SUT)



Step-4

Create a Test Case Class

- Select the class **MyMath**, and
- Right-click to get the popup menu.
- Select **New > JUnit Test Case**.



Step-5

New (JUnit Test Case)

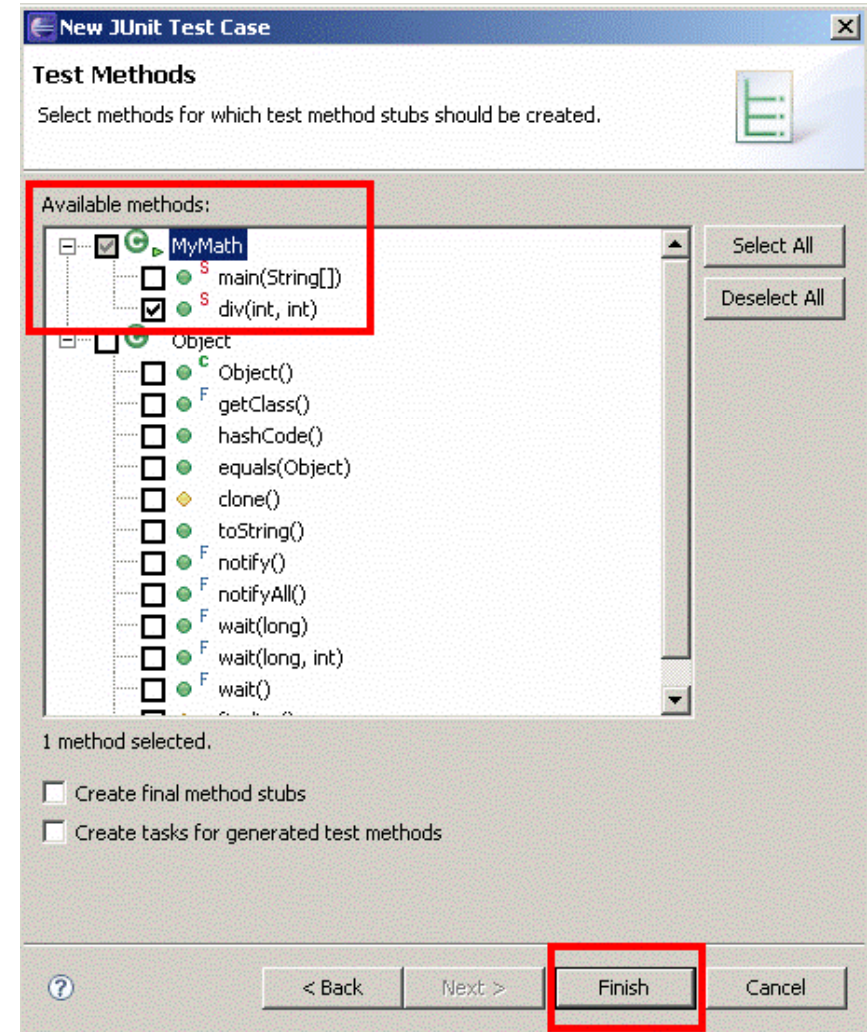
1. Select JUnit 4
2. Select the source folder of the test case to be **lab1/test**. To do this, click on the Browse... button on the same line and select the **test** folder.
3. [Optional] In the section “Which method stubs would you like to create?” check all of **setUpBeforeClass()**, **tearDownAfterClass()**, **setUp()**, and **tearDown()**. This will create methods with the annotations **@BeforeClass**, **@AfterClass**, **@Before**, and **@After**, respectively.
4. If you would like automatically generated comments, check the “Generate comments” box.
5. Click **NEXT**

The screenshot shows the 'New JUnit Test Case' dialog box. The 'JUnit Test Case' section at the top explains the purpose of the dialog. Below this, there are two radio buttons: 'New JUnit 3.8.1 test' (unselected) and 'New JUnit 4 test' (selected). The 'Source folder' is 'Lab 1/test' and the 'Package' is 'lab1'. The 'Name' is 'MyMathTest' and the 'Superclass' is 'java.lang.Object'. The 'Which method stubs would you like to create?' section has checkboxes for 'setUpBeforeClass()', 'tearDownAfterClass()', 'setUp()', 'tearDown()', and 'constructor', all of which are unchecked. The 'Do you want to add comments as configured in the properties of the current project?' section has a checkbox for 'Generate comments' which is also unchecked. The 'Class under test' is 'lab1.MyMath'. At the bottom, there are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'. The 'Next >' button is highlighted with a red rectangle.

Step-5.3

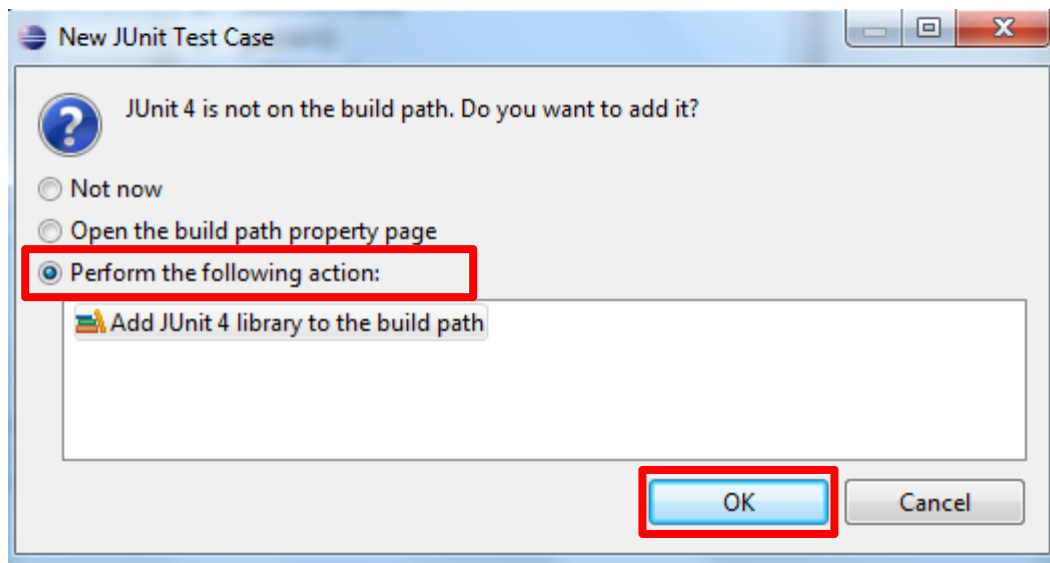
Test Methods

- Select the method **div(int,int)** for testing, and then click Finish.



Choosing JUnit-4 Library [Optional]

- Select the highlighted option
 - Click on **OK**



Step-6

Test Case Class

A new file **MyMathTest.java** will be created with a new class **MyMathTest**. The file will be opened in a new editor window, and the contents should resemble the following code:

```
package lab1;

import static org.junit.Assert.*;
import org.junit.Test;

public class MyMathTest {

    @Test
    public void testDiv() {
        fail("Not yet implemented");
    }

}
```

Remove this line

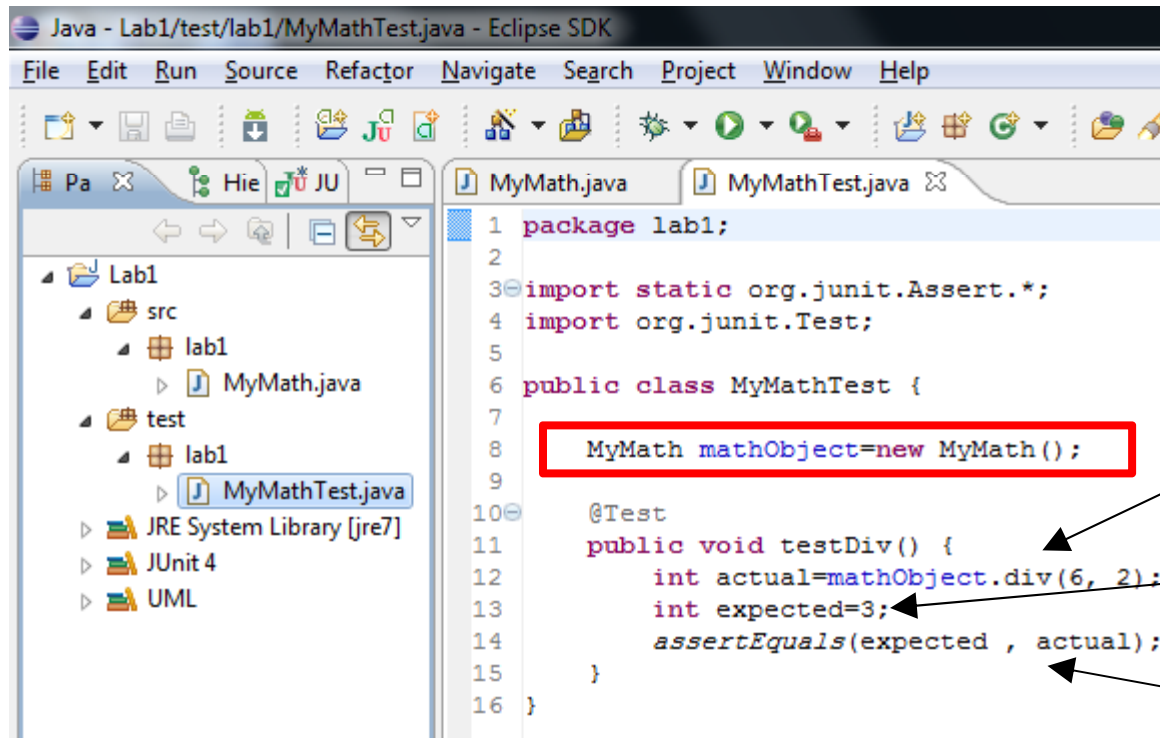
Proposed Test Case

- **Test Case for System Under Test (SUT)**
 - **MyMath.div(int, int)**
 - This method has two integer parameters and it will return the division of two input numbers (e.g. 6 and 2).

TEST CASE	INPUT		EXPECTED OUTPUT
	INTEGER-A	INTEGER-B	INTEGER-A/INTEGER-B = ?
TC-1	6	2	3

Implementation of Test Case

- Implementation of TC-1 in `testDiv()`
 - Add the required code in *MyMathTest*



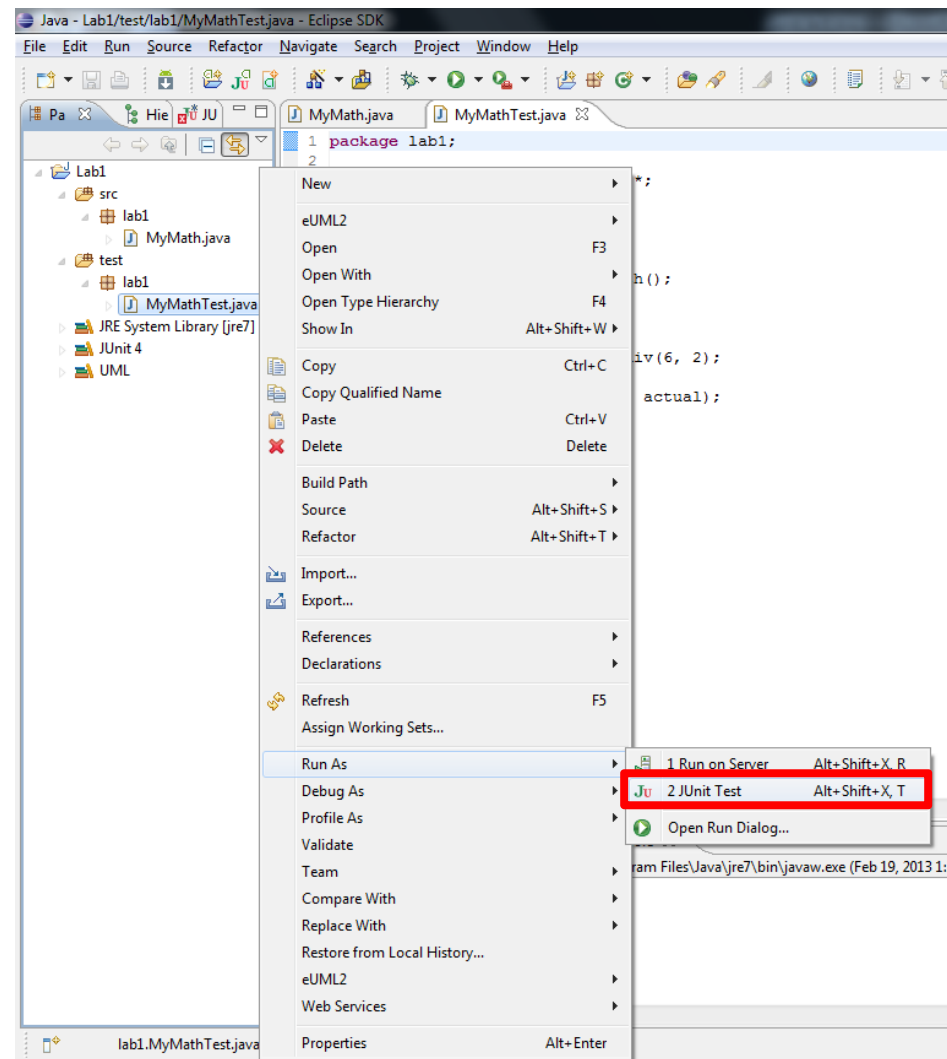
```
1 package lab1;
2
3 import static org.junit.Assert.*;
4 import org.junit.Test;
5
6 public class MyMathTest {
7
8     MyMath mathObject=new MyMath();
9
10    @Test
11    public void testDiv() {
12        int actual=mathObject.div(6, 2);
13        int expected=3;
14        assertEquals(expected , actual);
15    }
16 }
```

Actions send to
**System Under
Test (SUT).**

Responses
expected from
SUT.

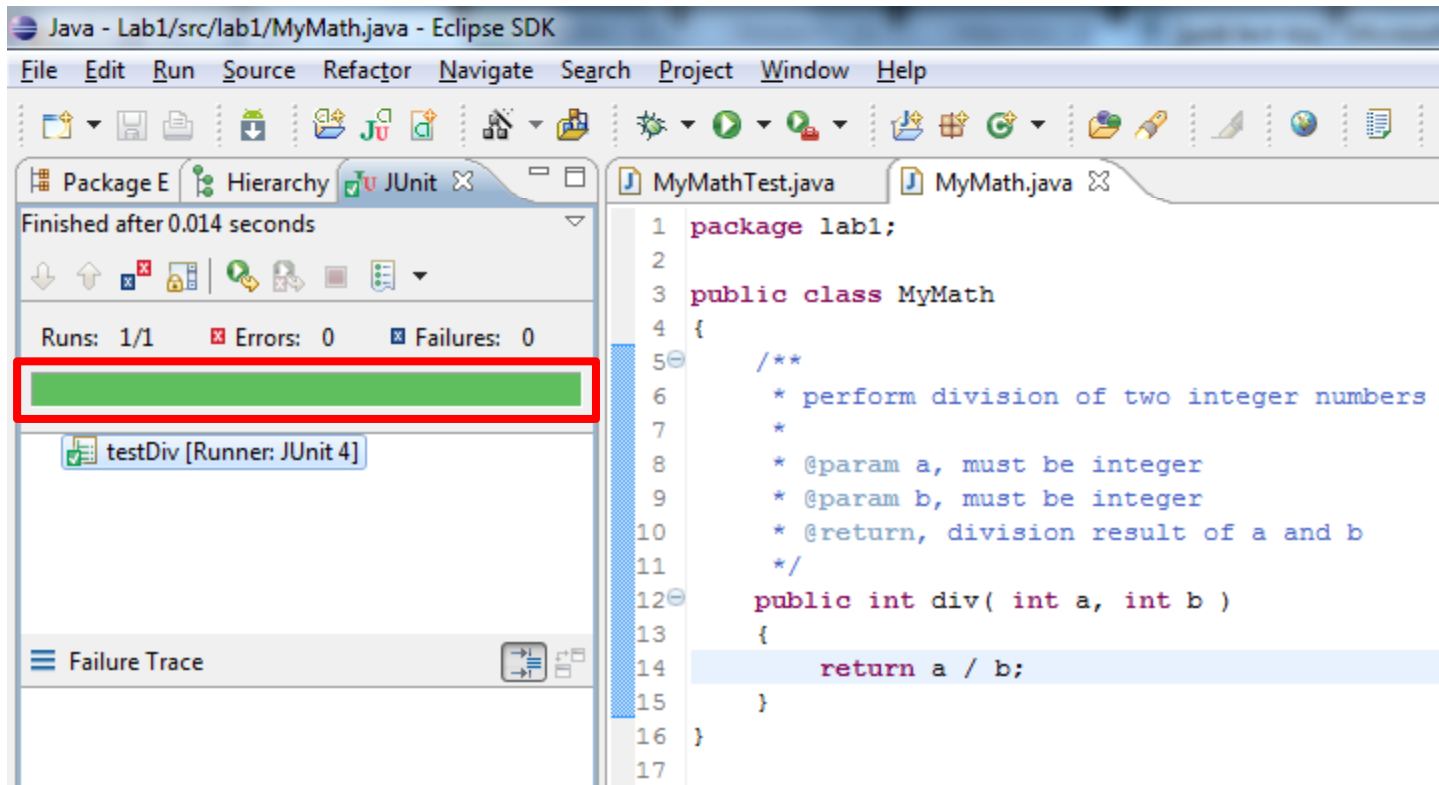
Determine
whether a test was
successful or not?

Run the JUnit Test Case



Test Verdict - *PASS*

- Finally, we got the green bar, our test is successful.



QUIZ - 1

- **Practical Lab Quiz**
 - **Date:** 24, 26 September 2013
 - **Time:** 12:00 to 13:00
- **Topics Covered (slides available on *LMS*)**
 - How to import java project in eclipse
 - Junit test method implementation
- **Rules**
 - Student will not allow to take a Quiz after 10 minutes of exam session.
 - No excuse for absent student.
 - In both above cases the student will get 0 (zero) mark.