

JUnit Guide

版本 作者 时间 描述
v1.0 Paolo 2014-03-01 Initial version

下载JUnit	1
将JUnit配置入eclipse	2
@ 导入JUnit	2
@ 配置JUnit源码(可选)	4
@ 另一种导入JUnit并配置其源码的方法（即从workspace中导入和配置）	6
创建JUnit 测试类（Test Case类）	12
运行JUnit测试类	17
@ 测试整个测试类	17
@ 测试JUnit测试类内的一个方法	19

下载JUnit

下载JUnit包，如junit3.8.1

<http://nchc.sourceforge.net/project/junit/junit/3.8.1/junit3.8.1.zip>

解压缩后如下：

Name	Size	Type
doc		File folder
javadoc		File folder
junit		File folder
cpl-v10.html	15 KB	HTML Document
junit.jar	119 KB	JAR File
README.html	22 KB	HTML Document
src.jar	57 KB	JAR File

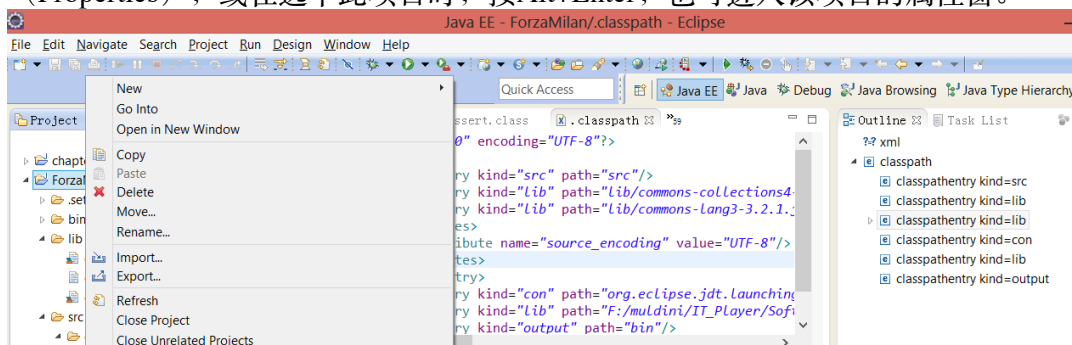
JUnit4.10.jar下载路径为：<http://sourceforge.net/projects/junit/>

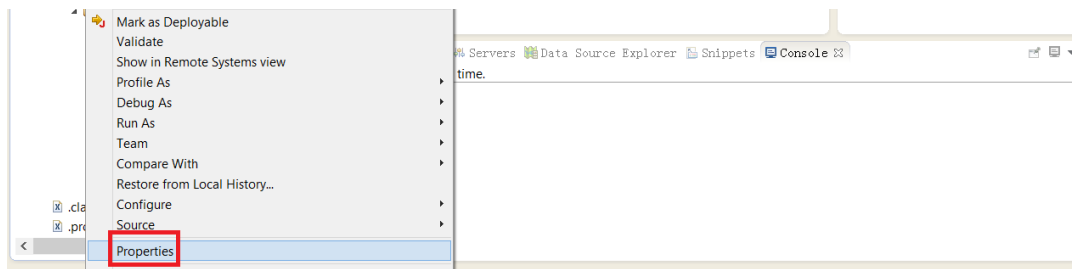
将JUnit配置入eclipse

@ 导入JUnit

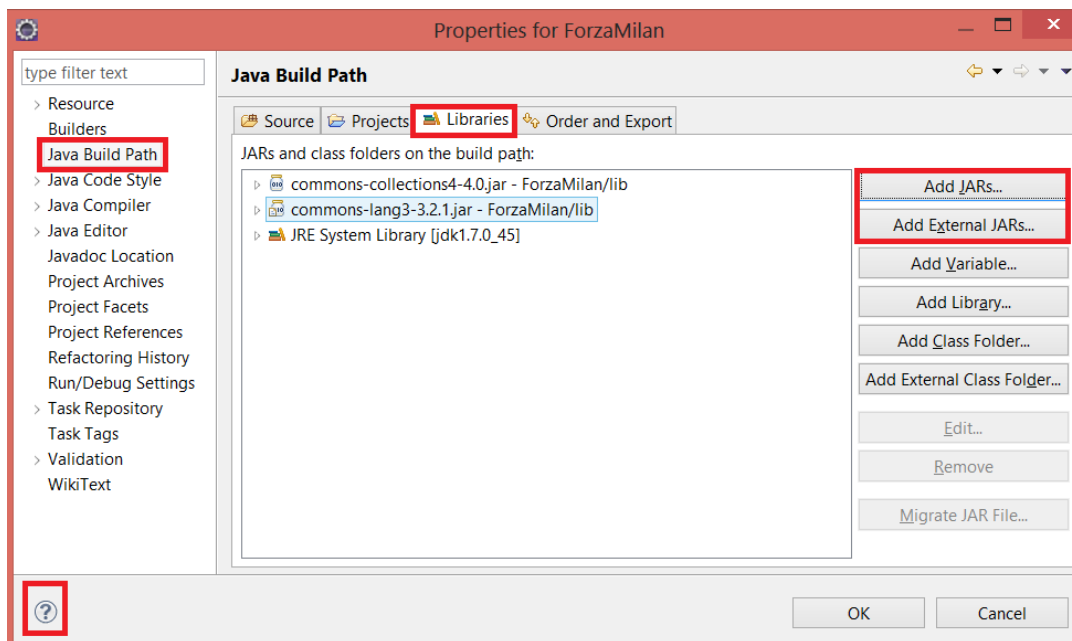
运行eclipse java SE或EE版，选择workspace后即可进入eclipse工作界面

鼠标移到某个项目（project）上点右键，在弹出菜单中选择该项目的属性（Properties），或在选中此项目时，按Alt+Enter，也可进入该项目的属性窗。

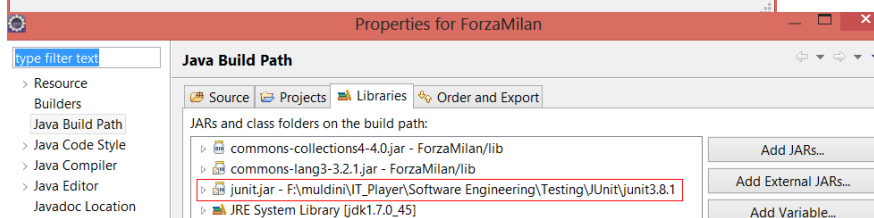
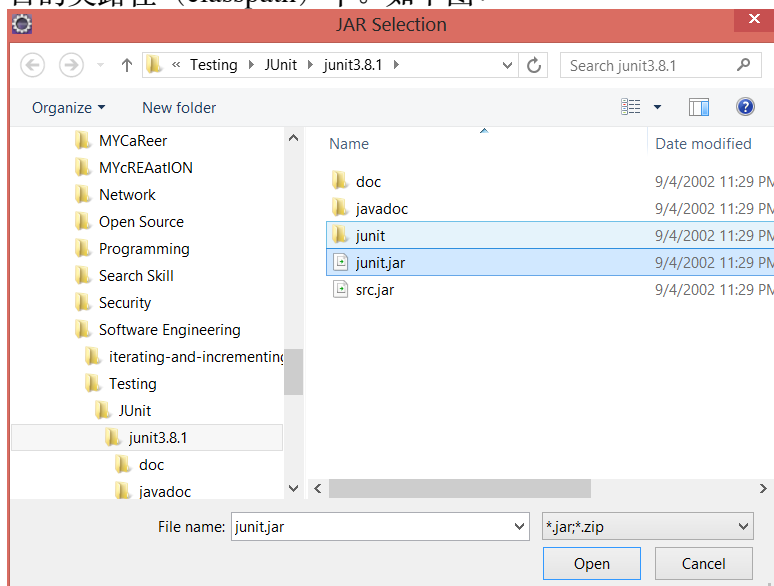


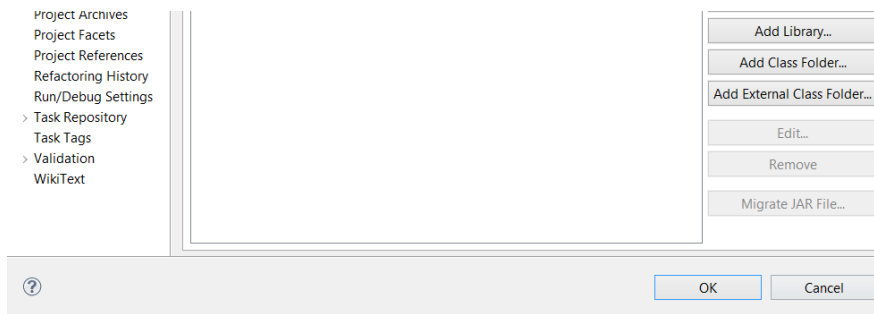


在弹出的属性面板内，选择左列的Java Build Path，再选Libraries，若JUnit包所在路径是当前workspace，则可点击Add JARs，否则点击Add External JARs，<欲知Java Build Path的使用说明详情，可点击面板左下角的问号，在弹出窗口中搜索“Java Build Path”>。



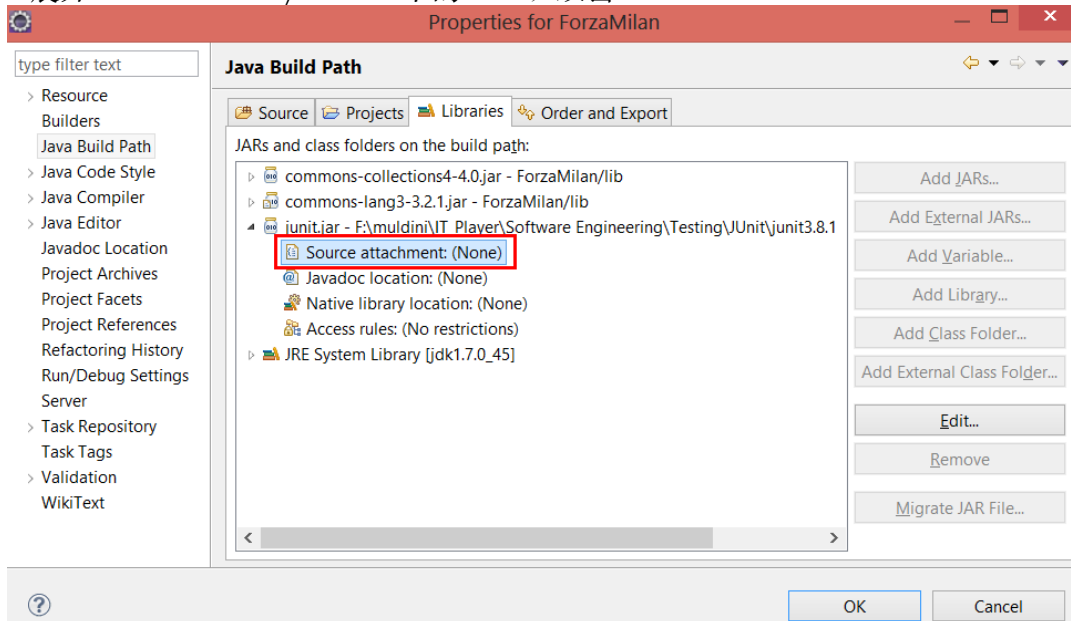
在点击Add External JARs的弹出窗中，选择junit.jar，确定即可将junit.jar配置到项目的类路径（classpath）中。如下图：



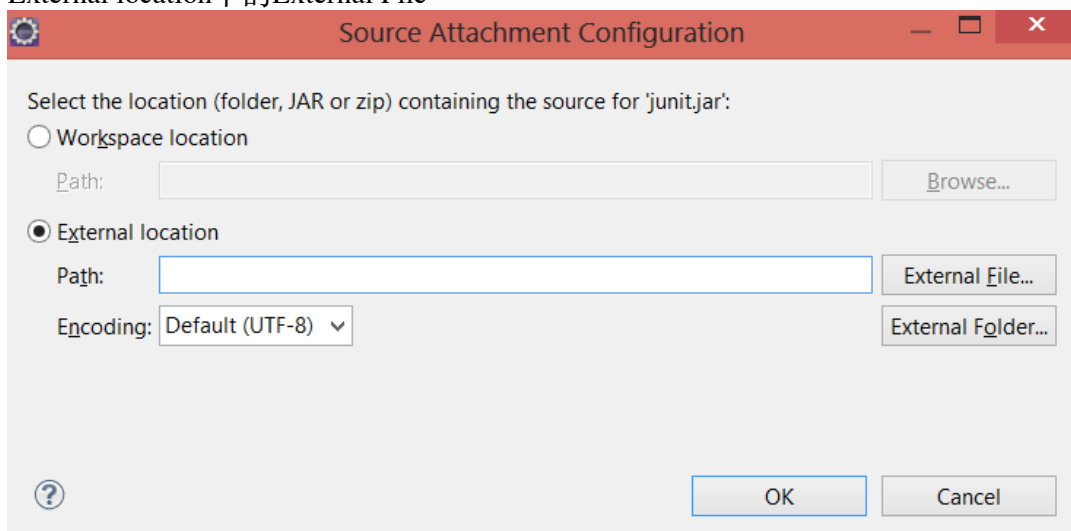


@ 配置JUnit源码 (可选)

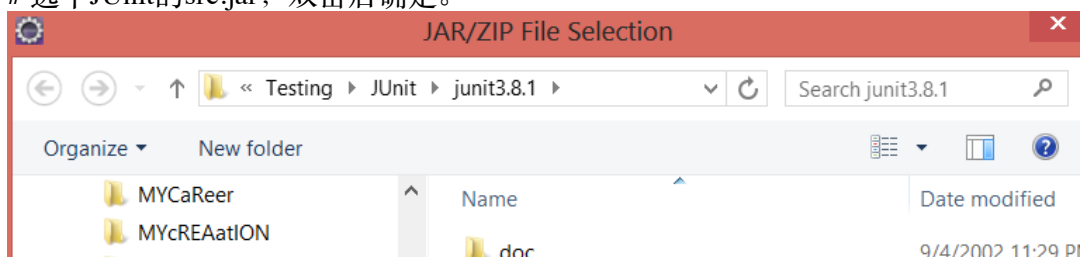
展开Java Build Path \Libraries下的JUnit，双击Source attachment

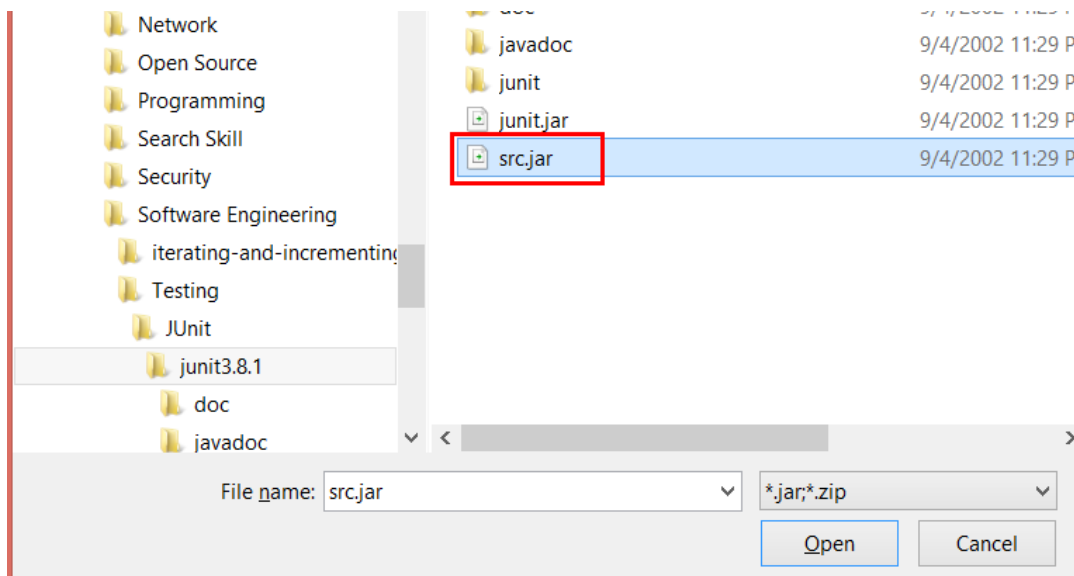


若JUnit包所在路径是当前workspace，则可点击Workspace location，否则点击External location下的External File

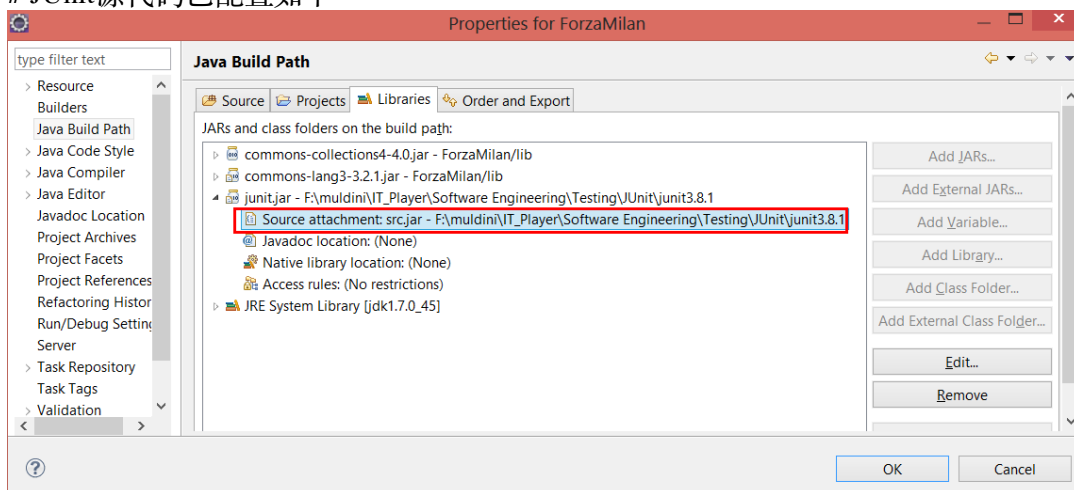


选中JUnit的src.jar，双击后确定。



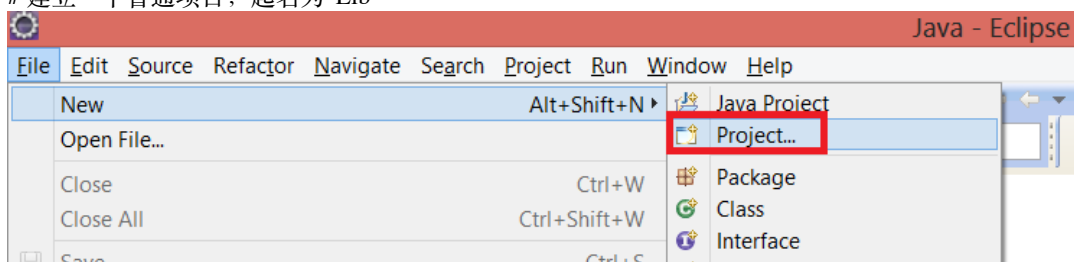


JUnit源代码已配置如下

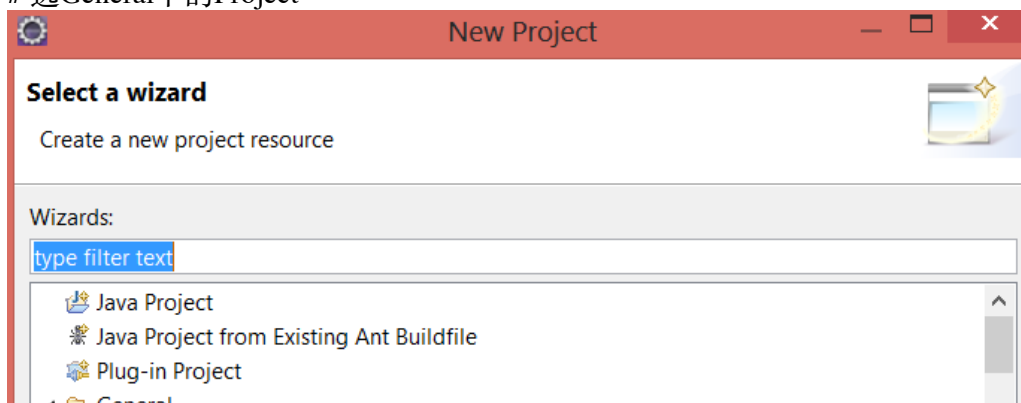


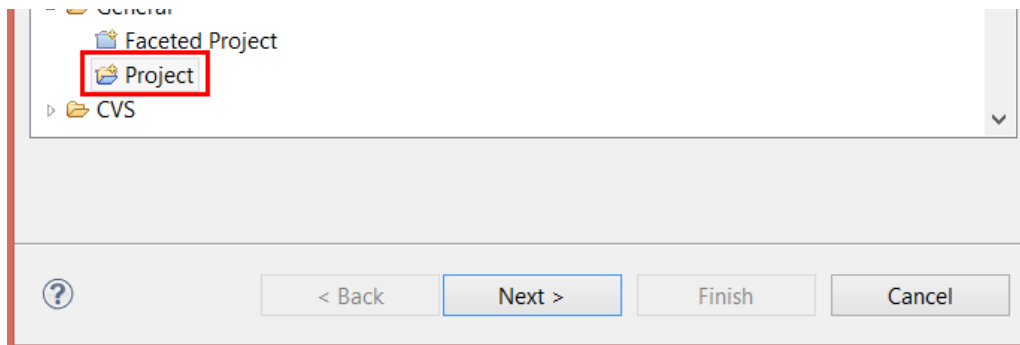
@ 另一种导入JUnit并配置其源码的方法（即从workspace中导入和配置）

建立一个普通项目，起名为“Lib”

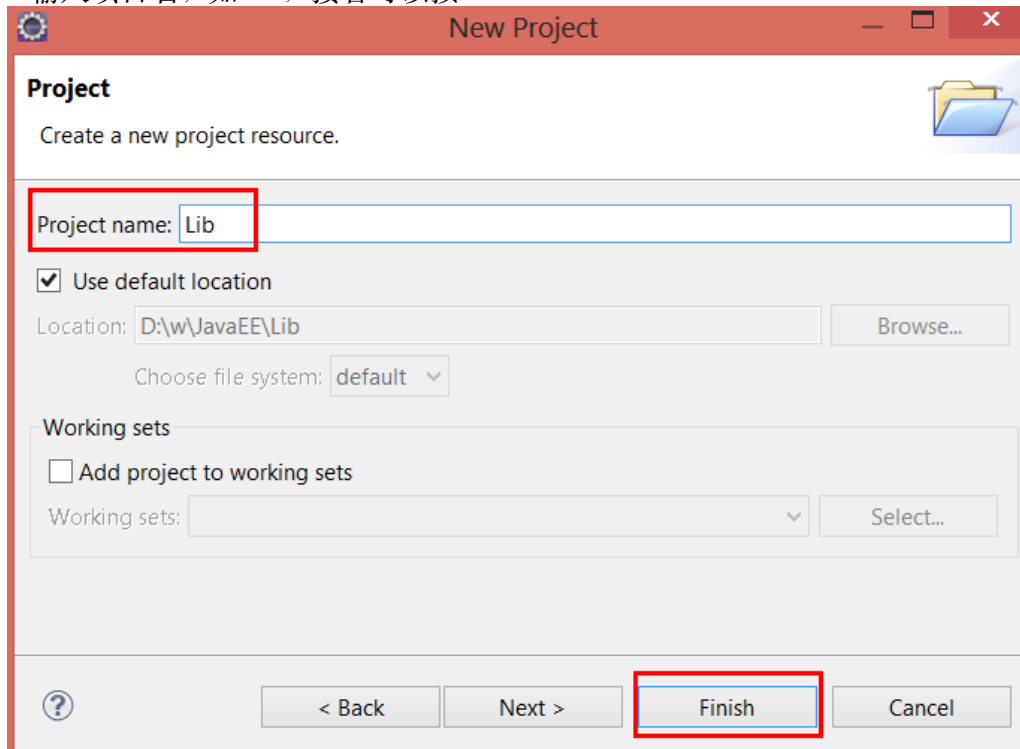


选General下的Project

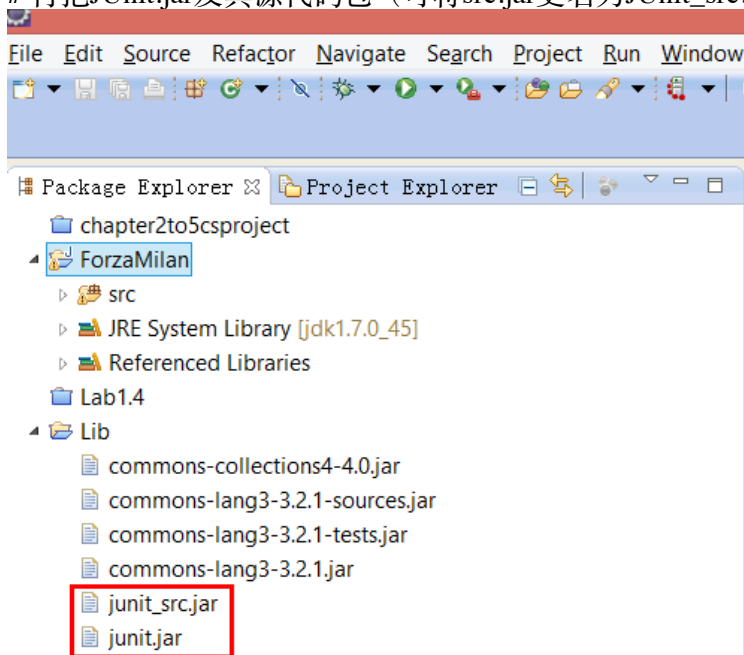




输入项目名，如Lib，接着可以按Finish

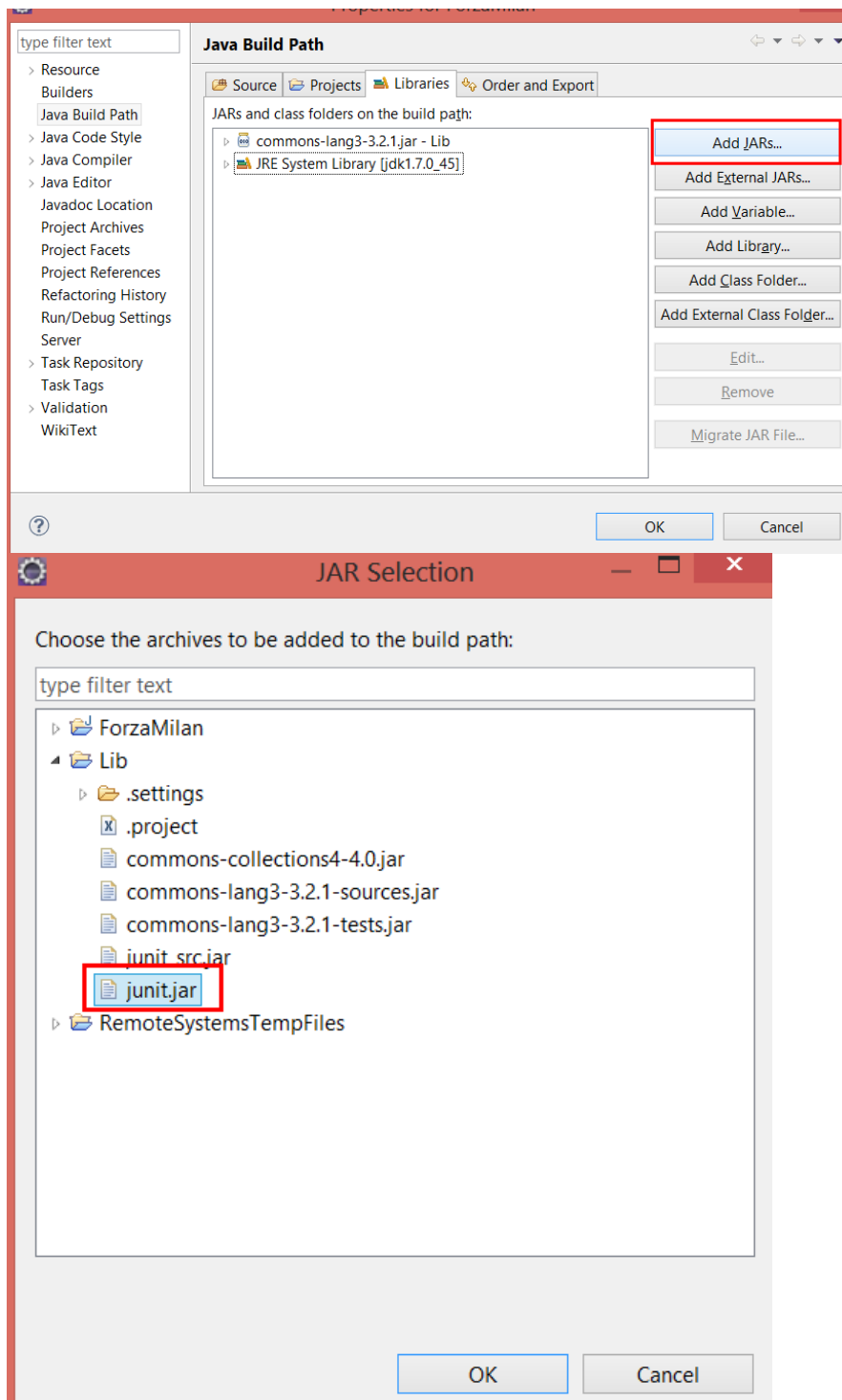


再把JUnit.jar及其源代码包（可将src.jar更名为JUnit_src.jar）拷贝到Lib，如下：

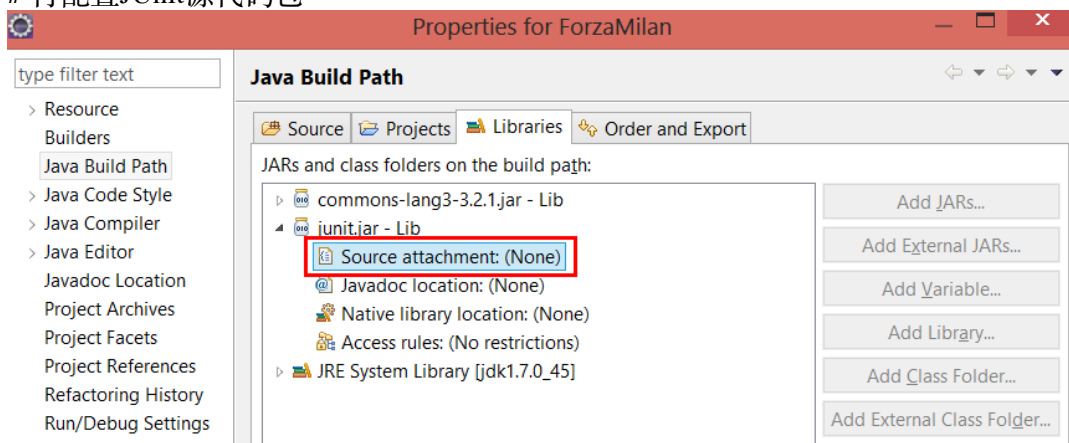


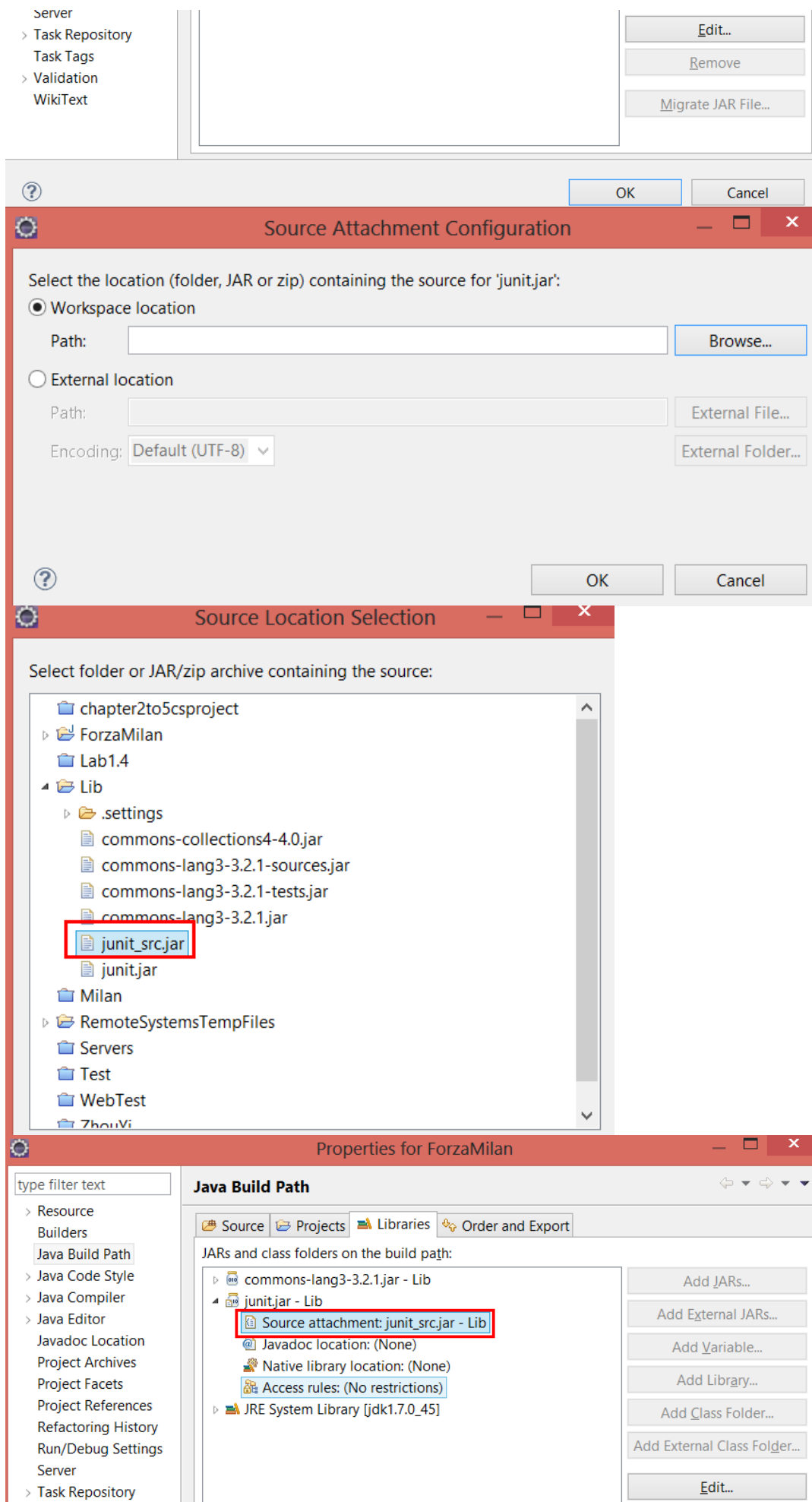
再把JUnit.jar配置到java项目中，选Add JARs





再配置JUnit源代码包

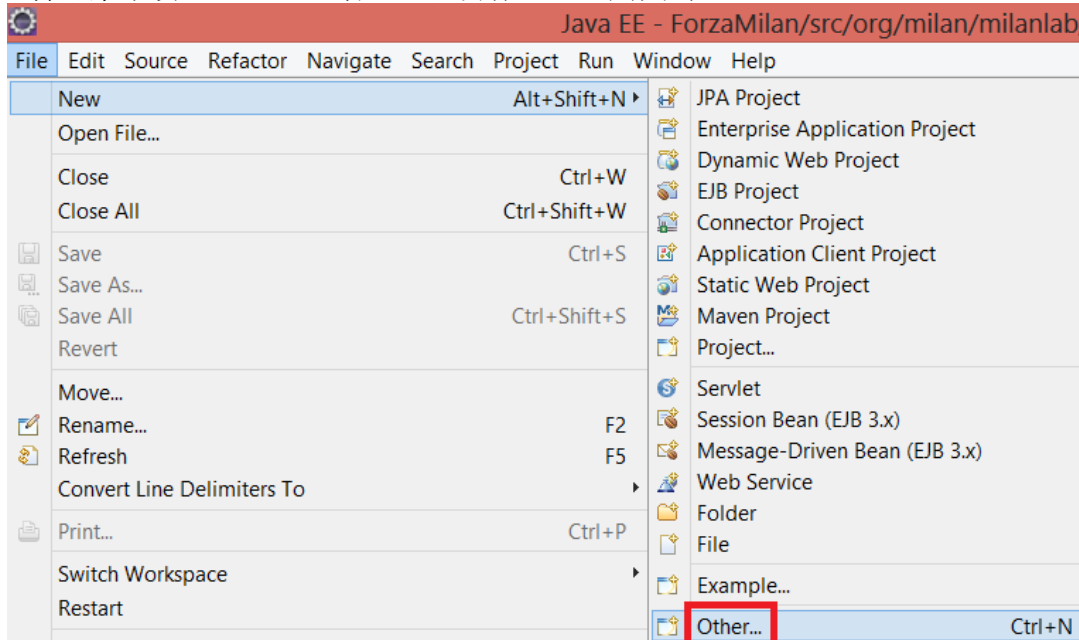




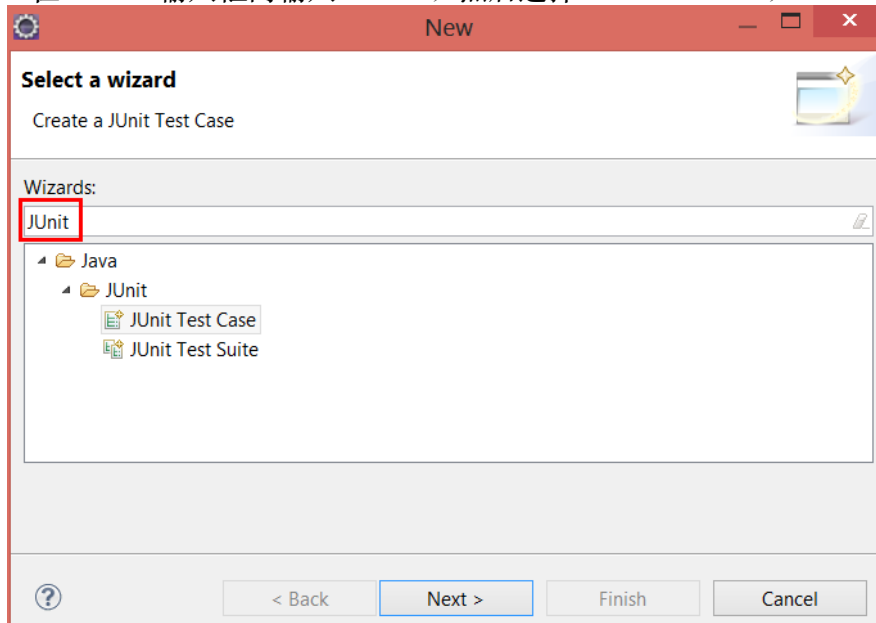


创建JUnit 测试类 (Test Case类)

打开菜单项File->New，若New下没有JUnit，则点击Other

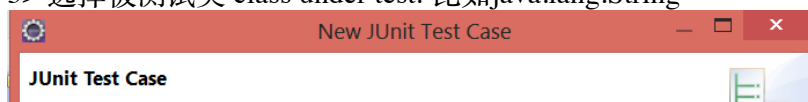


在Wizards输入框内输入“JUnit”，然后选择JUnit Test Case, Next



确定Test Case属性，然后Next

- 1>因为导入的JUnit包是3.8.1，所以应选择New JUnit 3 Test
- 2>通常是测试类 (Test Case) 和被测试类不在一个包 (package) 内，所以创建一个“test”包
- 3> 通常测试类的命名=被测试类的名+“Test”
- 4> JUnit3需要测试类继承junit.framework.TestCase
- 5> 选择被测试类 class under test: 比如java.lang.String



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:

Package:

Name:

Superclass:

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:

选择待测的方法，接着Finish

New JUnit Test Case

Test Methods

Select methods for which test method stubs should be created.

Available methods:

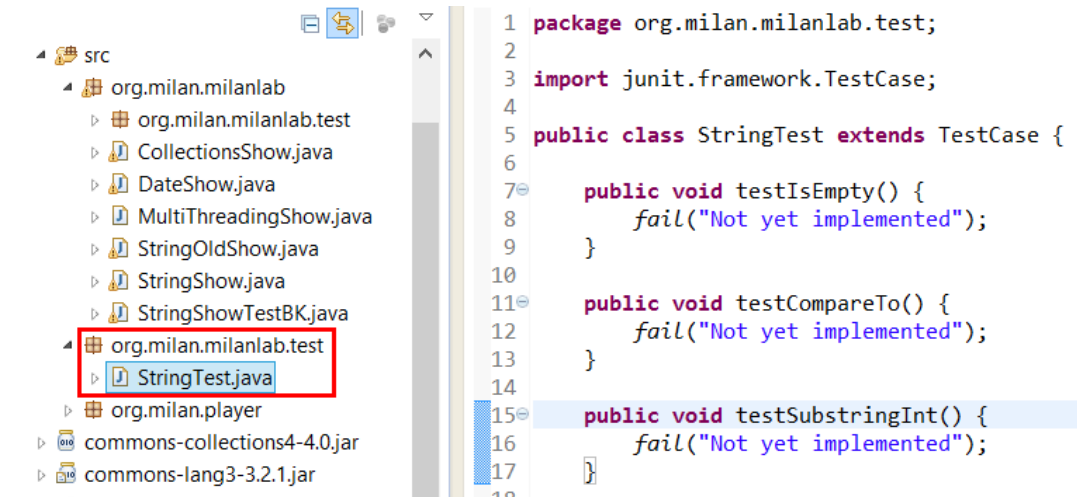
<input type="checkbox"/>	lastIndexOf(String)
<input type="checkbox"/>	lastIndexOf(String, int)
<input type="checkbox"/>	lastIndexOf(char[], int, int, char[], int, int, int)
<input checked="" type="checkbox"/>	substring(int)
<input type="checkbox"/>	substring(int, int)
<input type="checkbox"/>	subSequence(int, int)
<input type="checkbox"/>	concat(String)
<input type="checkbox"/>	replace(char, char)
<input type="checkbox"/>	matches(String)
<input type="checkbox"/>	contains(CharSequence)
<input type="checkbox"/>	replaceFirst(String, String)
<input type="checkbox"/>	replaceAll(String, String)
<input type="checkbox"/>	replace(CharSequence, CharSequence)
<input type="checkbox"/>	split(String, int)
<input type="checkbox"/>	split(String)

3 methods selected.

☐ Create final method stubs
☐ Create tasks for generated test methods

JUnit测试类（TestCase）被创建

Project Exp... ☒ Navigator ☐ StringTest.java



编写JUnit测试类

```
package org.milan.milanlab.test;
```

```
import junit.framework.TestCase;
```

```
public class StringTest extends TestCase {
```

```
    public void testIsEmpty() {
```

```
        String singer = "";
```

```
        boolean result = singer.isEmpty();
```

```
        assertTrue(result);
```

```
    }
```

```
    public void testCompareTo() {
```

```
        String singer1 = "GEM";
```

```
        String singer2 = "ZhangJie";
```

```
        boolean result = singer1.compareTo(singer2) >= 0;
```

```
        assertTrue("GEM should be prior to ZhangJie.", result);
```

```
    }
```

```
    public void testSubstringInt() {
```

```
        String singer = "I am a singer";
```

```
        String result = singer.substring(7, singer.length());
```

```
        assertEquals("singer", result);
```

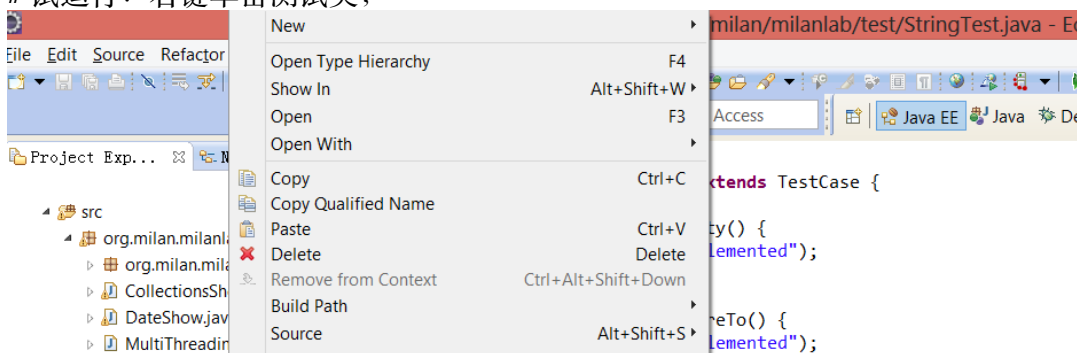
```
    }
```

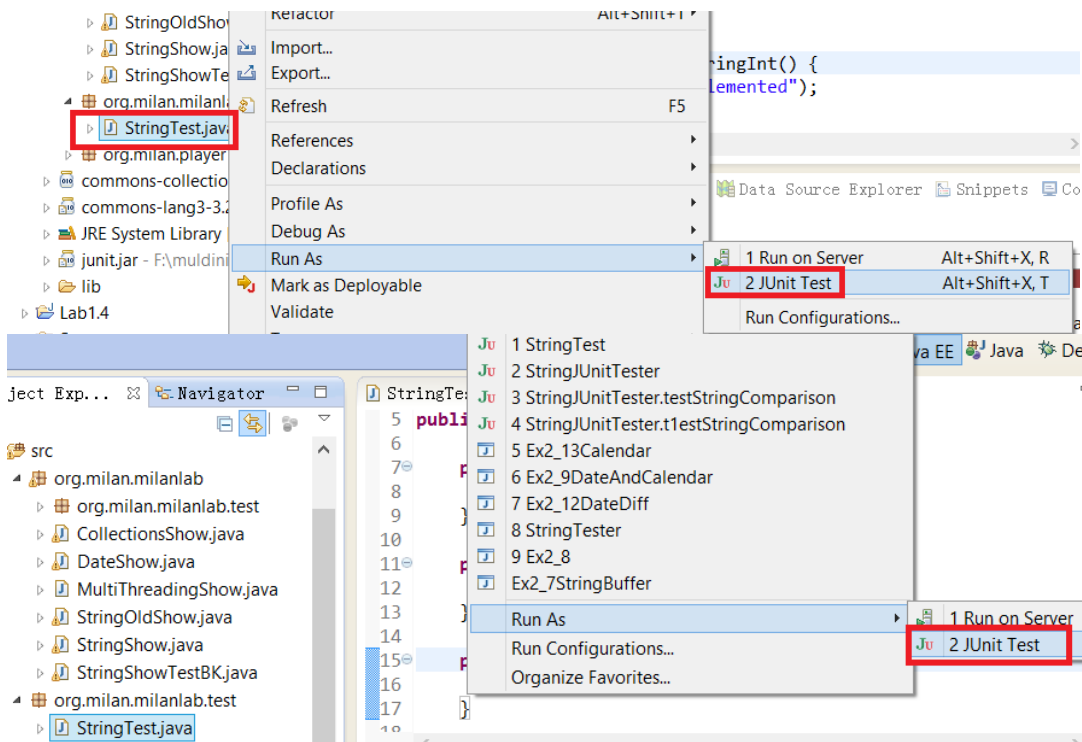
```
}
```

运行JUnit测试类

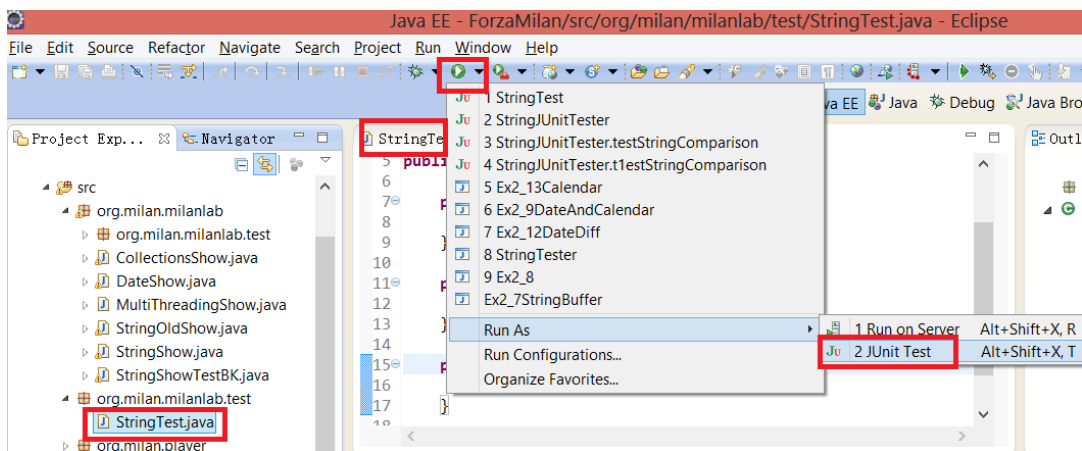
@ 测试整个测试类

试运行：右键单击测试类，

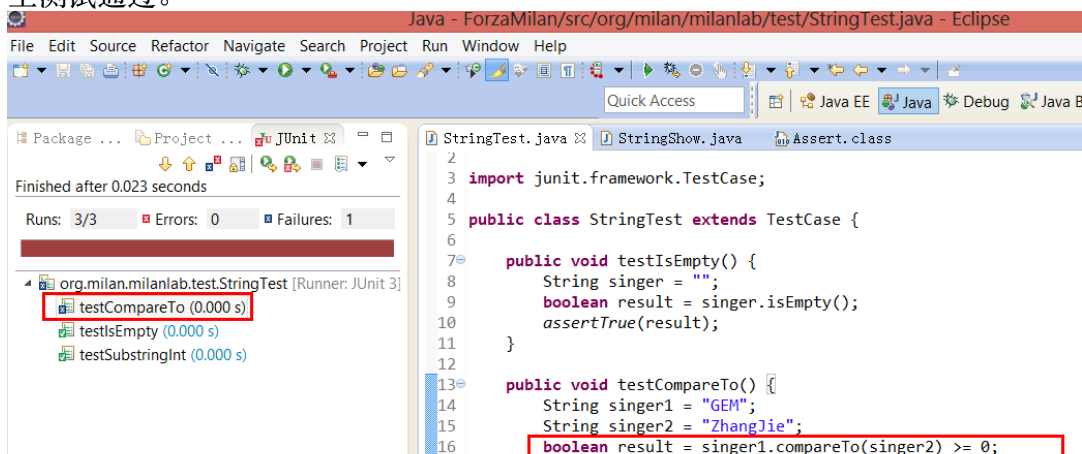


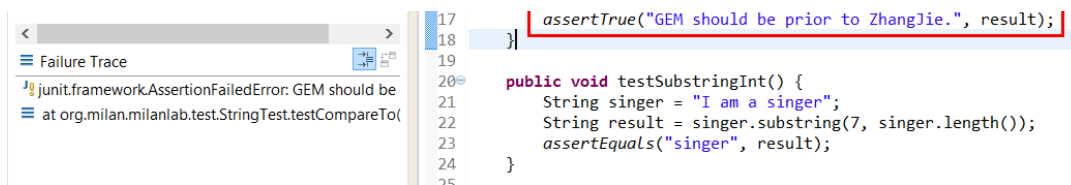


或者，在当前打开文件是StringTest.java的状态下，点击如下图所示的Run快捷符号，选择Run As > JUnit Test



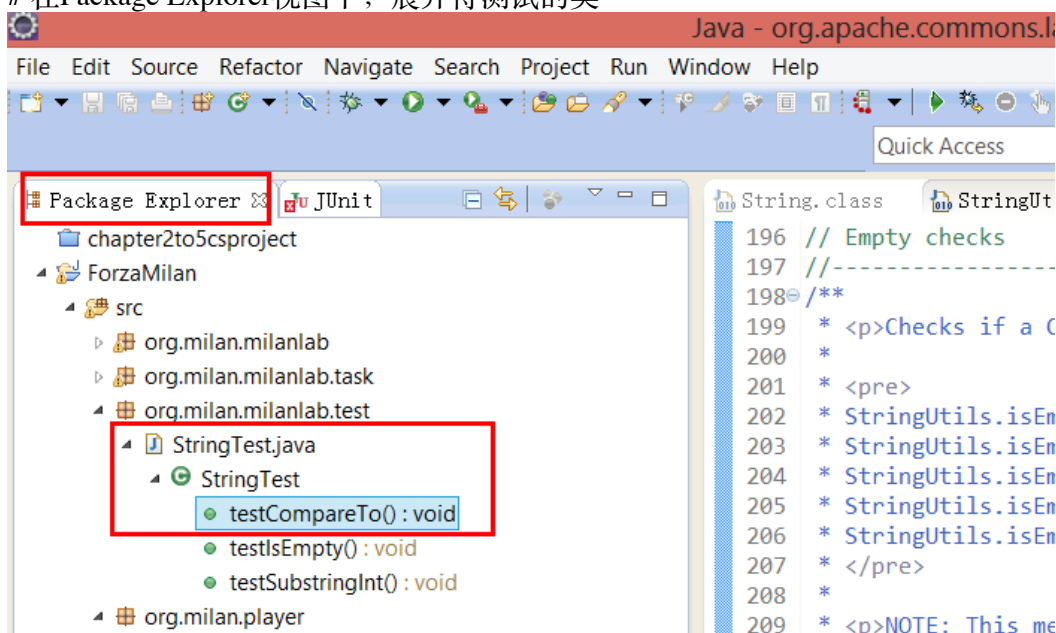
结果应该如下，JUnit视图会被自动打开，显示第2个测试方法失败，则修改它，直至测试通过。





@ 测试JUnit测试类内的一个方法

在Package Explorer视图下，展开待测试的类



鼠标在待测试类的待测试方法上单击右键，然后选择Debug as JUnit Test，即可。

