JAVAGURU INTRODUCTION TO JAVA

LESSON 11

CODE TESTING
OVERVIEW

THE PURPOSE OF SOFTWARE TESTS

- A test is a piece of software, which executes another piece of software in order to confirm that code operates as expected
- A test can check
  - Expected state (state testing)
  - Expected sequence of events (behaviour testing)
- Having high test coverage allows to develop new features without being afraid to break existing code

PRIMARY SOFTWARE TESTING SCOPES

- **Unit tests**
  - Targets a small unit of code (e.g. a method or a class)
  - External class dependencies should be replaced with test implementation objects (mocks)
- **Integration tests**
  - Aims to test the behaviour of a component or the integration between a set of components
  - Check that the whole system works as intended

UNIT TESTING APPROACH: MANUAL TESTING

- Executing a test cases manually without any tool support is known as manual testing
- **Time-consuming and tedious**
  - Since test cases are executed by human resources, it is very slow and tedious
- **Huge investment in human resources**
  - As test cases need to be executed manually, more testers are required in manual testing
- **Less reliable**
  - Manual testing is less reliable, as it has to account for human errors
- **Non-programmable**
  - No programming can be done to write sophisticated tests to fetch hidden information

UNIT TESTING APPROACH: AUTOMATED TESTING

- Taking tool support and executing the test cases by using an automation tool is known as automation testing
- **Fast**
  - Automation runs test cases significantly faster than human resources
- **Less investment in human resources**
  - Test cases are executed using automation tools, so less number of testers are required in automation testing
- **More reliable**
  - Automation tests are precise and reliable
- **Programmable**
  - Testers can program sophisticated tests to bring out hidden information

JUNIT FRAMEWORK

JUNIT TESTING APPROACH

- **JUnit** is a unit testing framework for Java programming language
- **JUnit test** is a method contained in a class which is only used for testing (also called a test class)
- **Formally written** unit test case is characterised by:
  - Known input
  - Expected output

1. JUNIT TEST EXAMPLE: SYSTEM UNDER TEST

```
public class Calculator {

    public int sum(int a, int b) {
        return a + b;
    }

}
```

2. JUNIT TEST EXAMPLE: TEST CLASS

```
public class CalculatorTest {

    private Calculator victim;

    @Before
    public void setUp() {
        victim = new Calculator();
    }

    @Test
    public void shouldCalculateSum() {
        int result = victim.sum(3, 5);
        assertEquals(8, result);
    }

}
```

MOST COMMON JUNIT ANNOTATIONS: TEST DECLARATION

Annotation	Description
`@Test` `public void testCase() {}`	The @Test annotation indicates the following method as a test method
`@Test(expected = Exception.class)` `public void testCase() {}`	If the method does not throw the given exception, the test will fail
`@Test(timeout = 500)` `public void testCase() {}`	If the method takes longer than 500 milliseconds, the test will fail
`@Ignore` `public void testCase() {}`	This annotation is useful when you want temporarily disable the execution of a specific test

MOST COMMON JUNIT ANNOTATIONS: TEST PREPARATION

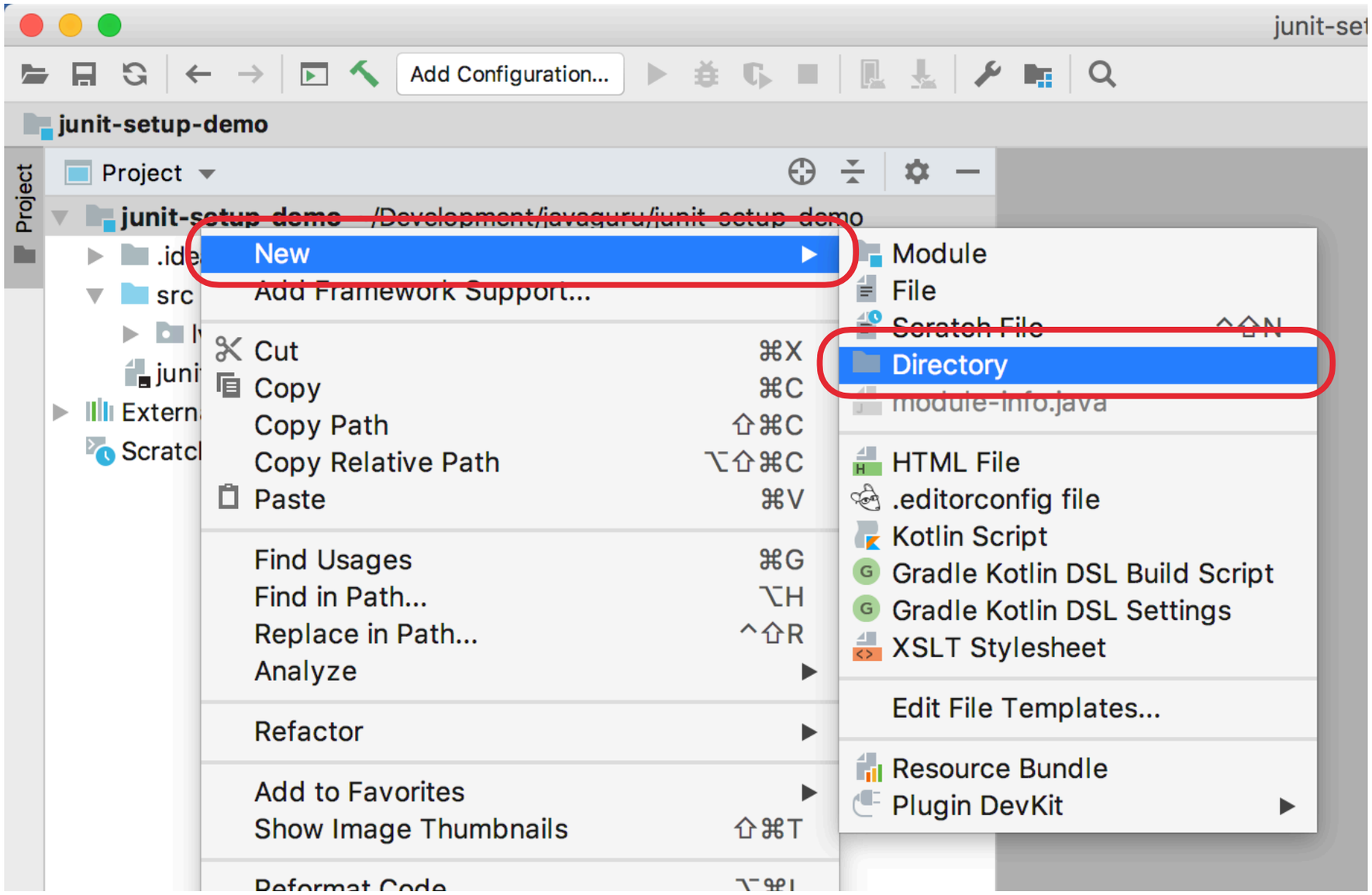
Annotation	Description
`@Before` `public void setUp() {}`	This method is executed before each test
`@After` `public void tearDown() {}`	This method is executed after each test
`@BeforeClass` `public static void setUp() {}`	The following static method is executed once, before the start of all tests
`@AfterClass` `public static void tearDown() {}`	The following static method is executed once after all tests have been completed

MOST COMMON ASSERT STATEMENTS

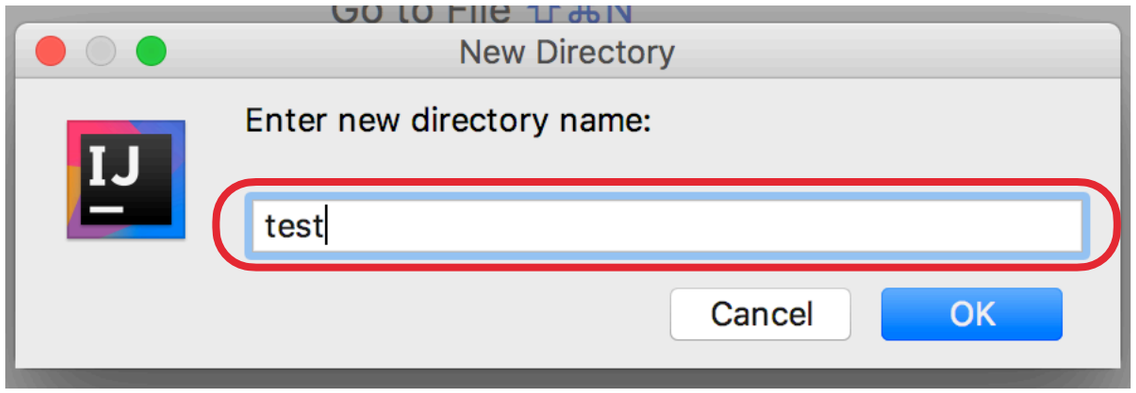
Assertion	Description
`Assert.assertEquals(expected, actual);` `Assert.assertNotEquals(expected, actual);`	This method is executed before each test
`Assert.assertTrue(actual);` `Assert.assertFalse(actual);`	This method is executed after each test
`Assert.assertNull(actual);` `Assert.assertNotNull(actual);`	The following static method is executed once, before the start of all tests
`Assert.assertSame(expected, actual);` `Assert.assertNotSame(expected, actual);`	The following static method is executed once after all tests have been completed

MANUAL
PROJECT SETUP

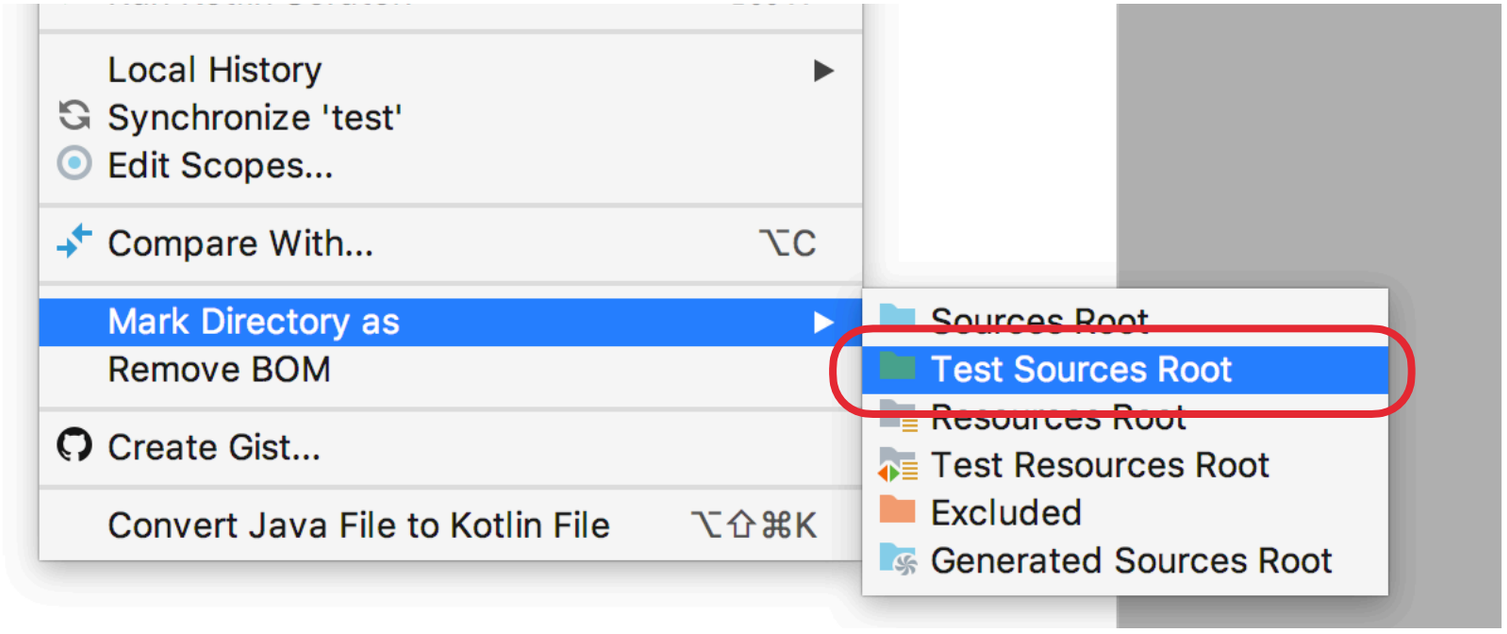
1. MANUAL JUNIT SETUP: CREATE NEW DIRECTORY



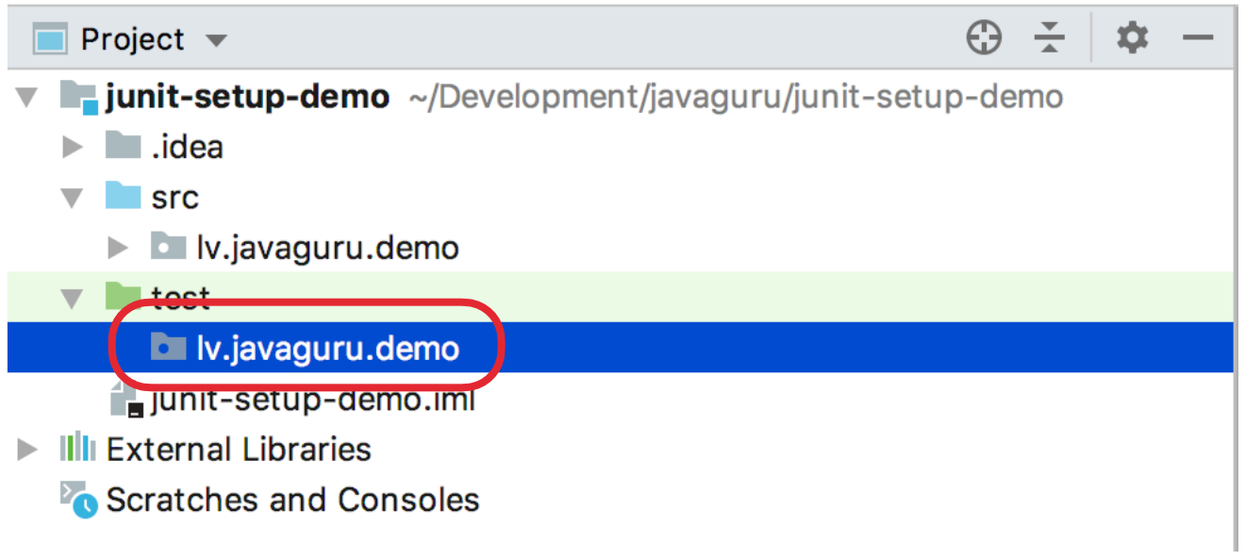
2. MANUAL JUNIT SETUP: NAME IT A TEST



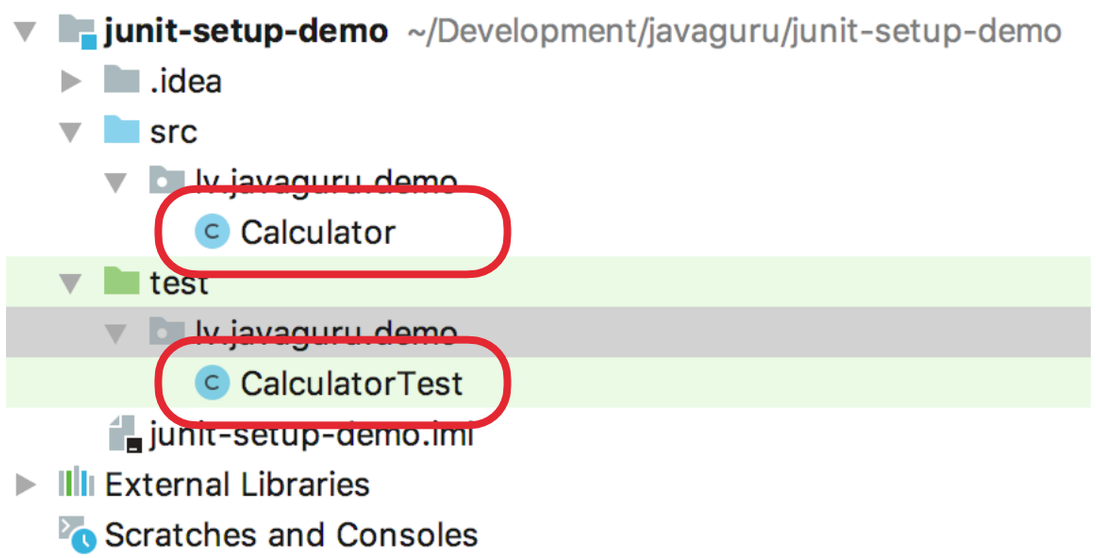
3. MANUAL JUNIT SETUP: MARK IT AS A TEST SOURCE ROOT



4. MANUAL JUNIT SETUP: CREATE DUPLICATED PACKAGE NAME

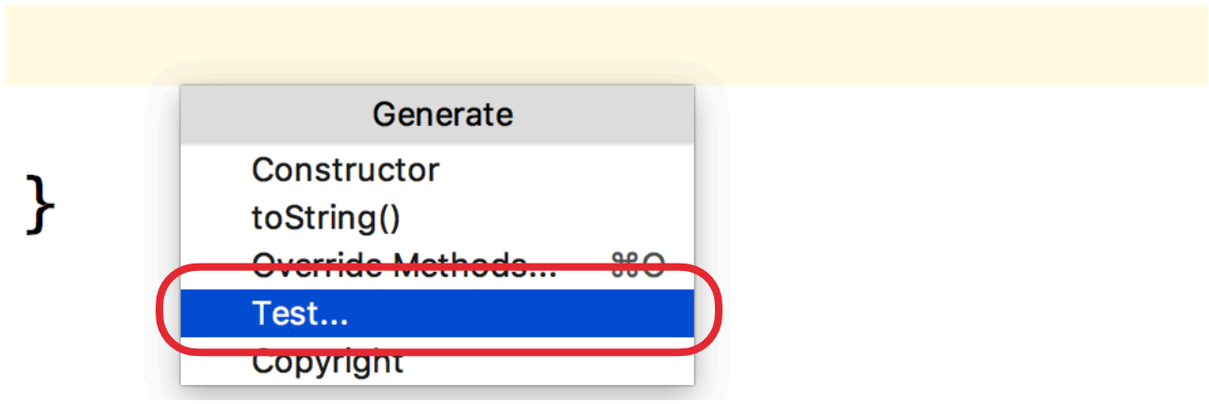


5. MANUAL JUNIT SETUP: CREATE TEST CLASS

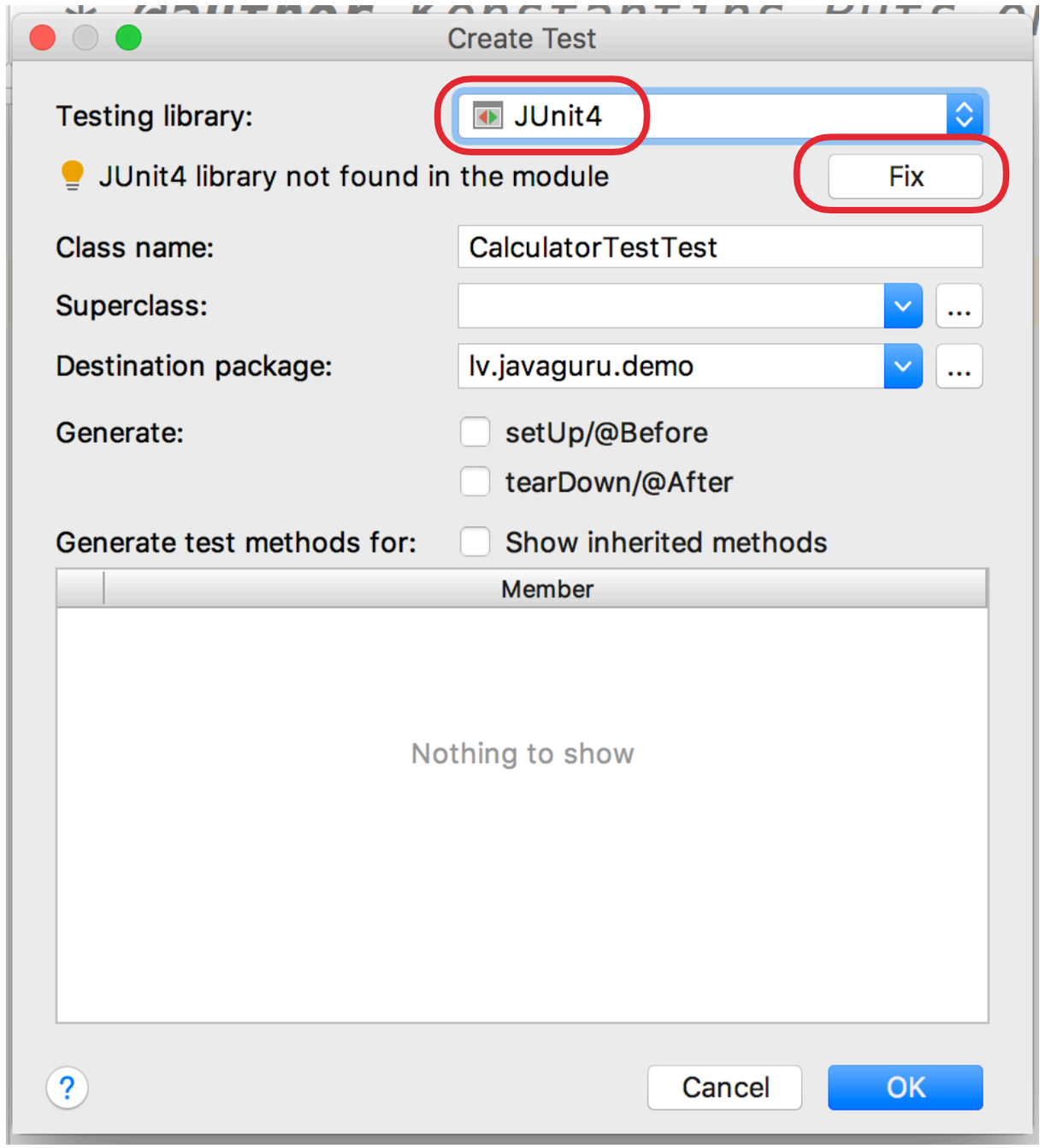


6. MANUAL JUNIT SETUP: GENERATE TEST IN TEST CLASS

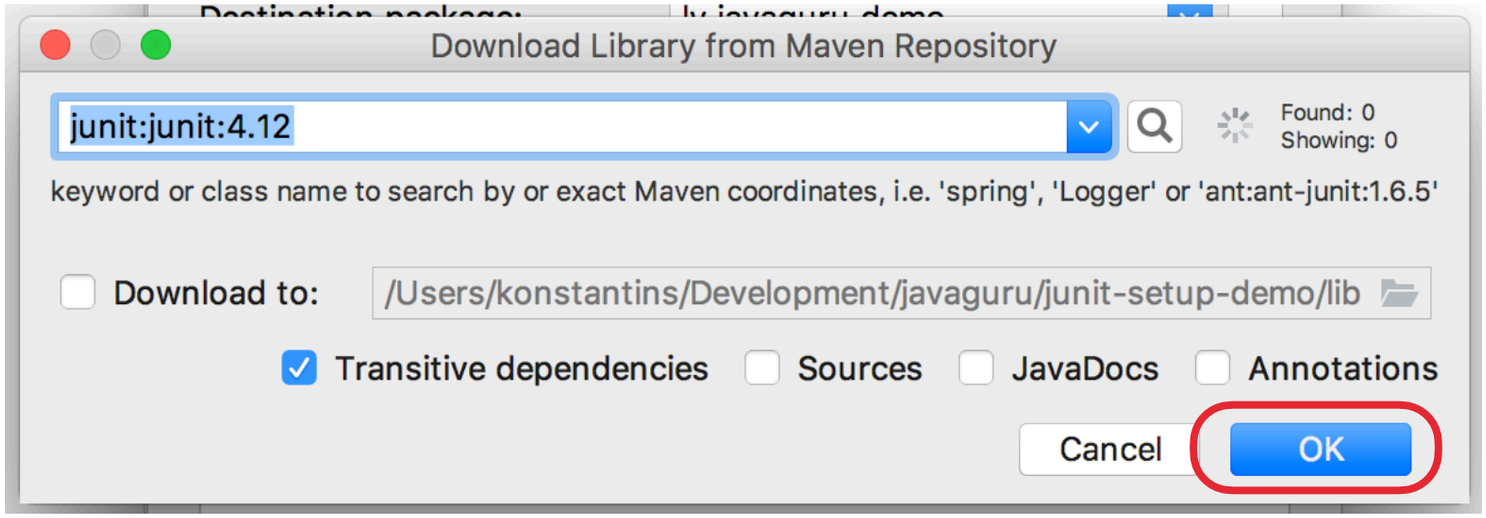
```
public class CalculatorTest {

}
```



7. MANUAL JUNIT SETUP: SET AND INCLUDE TESTING LIBRARY



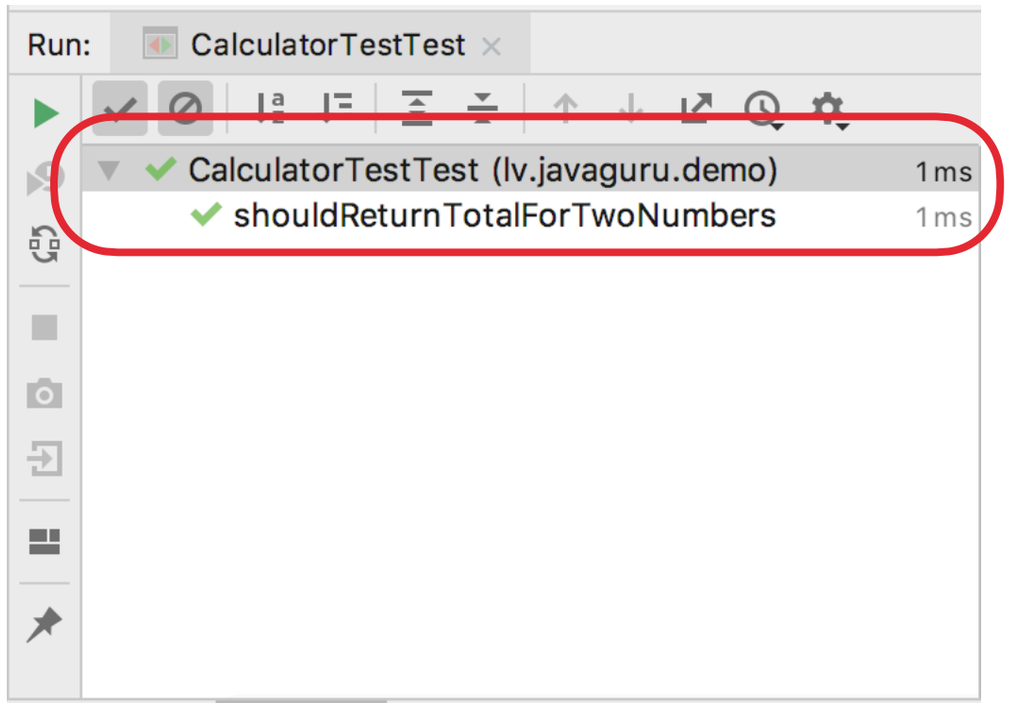
8. MANUAL JUNIT SETUP: DOWNLOAD LIBRARY



9. MANUAL JUNIT SETUP: WRITE TEST SCENARIO

```
12  public class CalculatorTestTest {
13
14      private Calculator victim;
15
16      @Before
17      public void setUp() {
18          victim = new Calculator();
19      }
20
21      @Test
22      public void shouldReturnTotalForTwoNumbers() {
23          int result = victim.sum( ...values: 2, 3);
24          Assert.assertEquals( expected: 5, result);
25      }
26
27  }
28
```

10. MANUAL JUNIT SETUP: RUN TESTS



REFERENCES

- https://junit.org/junit4/
- https://www.tutorialspoint.com/junit/junit_basic_usage.htm
- http://www.vogella.com/tutorials/JUnit/article.html
- https://www.swtestacademy.com/junit-tutorial/
- https://dzone.com/articles/7-popular-unit-test-naming