



JUnit Tutorial

Kuan-Ting Lai
2020/3/29

JUnit – Automatic Testing

- Unit-testing framework
- "test a little, code a little, test a little, code a little."
- Annotations to identify test methods
- Assertions to test results
- Test runners to run tests
- Run automatically

What is Unit Test Case?

- Part of code that ensures other part of code work as expected
- Define input and expected output
- At least two unit tests – one positive and one negative tests

Environment Setup

- Download JUnit archive
 - Download the latest JUnit jar file from <http://www.junit.org>
 - We are using <https://github.com/downloads/junit-team/junit/junit-4.10.jar> in this tutorial
- (Optional) Set JUnit environment
 - **Windows:** Set the environment variable JUNIT_HOME to C:\JUNIT
 - **Linux:** export JUNIT_HOME = /usr/local/JUNIT
- (Optional) Set CLASSPATH Variable
 - **Windows:** Set the environment variable CLASSPATH to %CLASSPATH%;%JUNIT_HOME%\junit4.12.jar;.

Running with Jar

- **Windows:** `java -cp ".;\junit-4.10.jar" YourClass`
- **Linux:** `java -cp " ../junit-4.10.jar" YourClass`

JUnit Test Framework

- **Fixtures**
 - `setup()` – runs before every test
 - `tearDown()` – runs after every test
- **Test Suites**
 - `@RunWith`
 - `@Suite`
- **Test Runners**
 - Run test cases
- **JUnit Classes**
 - `Assert`
 - `TestCase`
 - `TestResult`

Create a Class to be Tested

```
/* * This class prints the given message on console. */  
public class MessageUtil {  
    private String message;  
  
    //Constructor //@param message to be printed  
    public MessageUtil(String message){  
        this.message = message;  
    }  
  
    // prints the message  
    public String printMessage() {  
        System.out.println(message);  
        return message;  
    }  
}
```

Create Unit Tests (TestJUnit.java)

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class TestJUnit {

    String message = "Hello World";
    MessageUtil messageUtil = new MessageUtil(message);

    @Test
    public void testPrintMessage() {
        assertEquals(message, messageUtil.printMessage());
    }
}
```


Create Test Runner Class (TestRunner.java)

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

public class TestRunner {

    public static void main(String[] args) {
        Result result = JUnitCore.runClasses(TestJunit.class);

        for (Failure failure : result.getFailures()) {
            System.out.println(failure.toString());
        }

        System.out.println(result.wasSuccessful());
    }
}
```

Compile & Run JUnit

- `javac -cp ".;\junit-4.10.jar" MessageUtil.java TestJUnit.java TestRunner.java`
- `java -cp ".;\junit-4.10.jar" TestRunner`

Annotations

- @Test
- @Before
- @After
- @BeforeClass
- @AfterClass
- @Ignore

Create Unit Testing using IntelliJ

Select Your Class and Press “Alt + Enter”

Calculator [C:\Users\User\OneDrive\Teaching\107-2物件導向程式設計\homework\hw2\Calculator] - ...src\CalculatorForm.java [Calculator] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Calculator > src > CalculatorForm

Project: Calculator C:\Users\User\OneDrive\Teaching\107-2物件導向程式設計\homework\hw2\Calculator

- .idea
- out
 - artifacts
 - Calculator_jar
 - Calculator.jar
 - production
 - test
- src
 - META-INF
 - CalculatorFormTest
 - CalculatorForm
 - CalculatorForm.form
- External Libraries
 - hamcrest-core-1.3.jar library root
 - junit-4.12.jar library root
 - < 11 > C:\Program Files\Java\jdk-11.0.2\bin\java.exe
 - JUnit4
 - JUnit5.3
 - testng

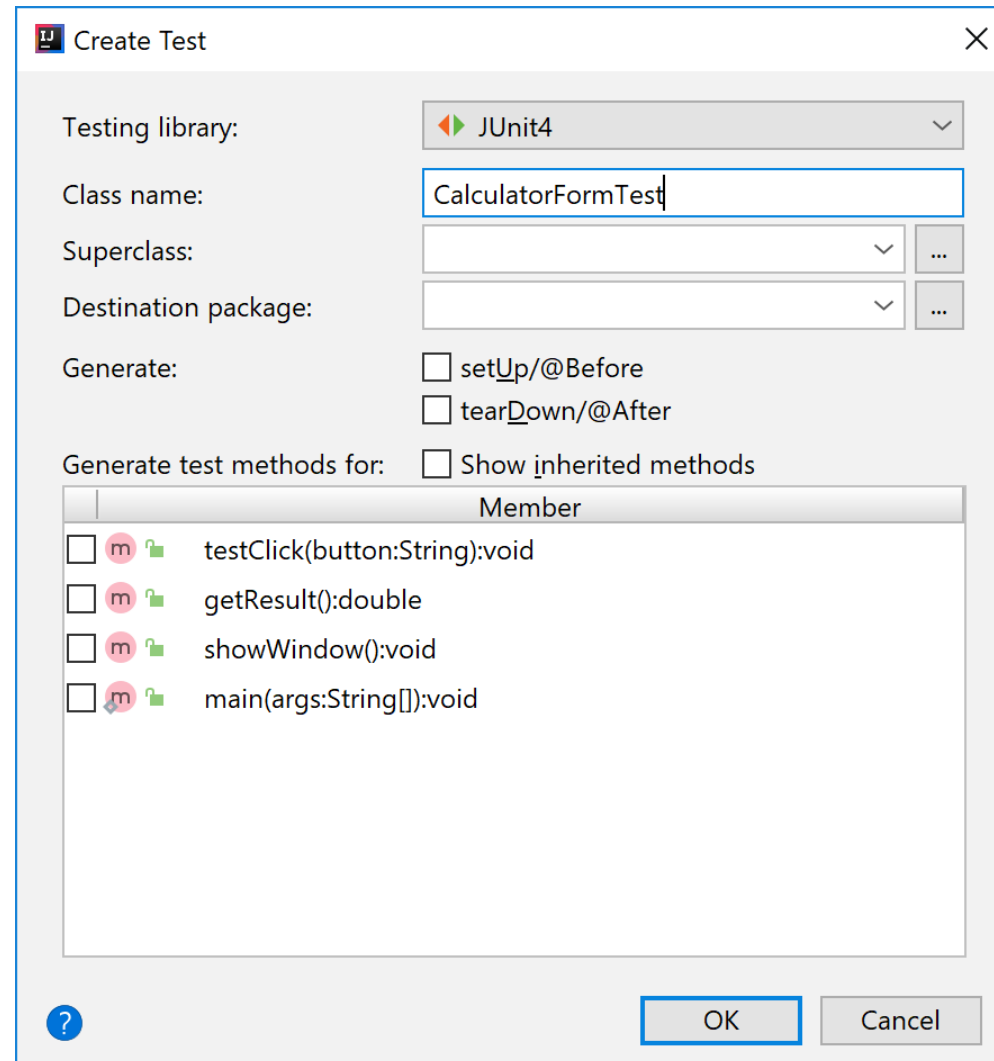
```
1 import ...
2
3
4
5
6 public class CalculatorForm {
7     private JTextField textField;
8     private JPanel calculatorForm;
9     private JButton button1;
10    private JButton button2;
11    private JButton button3;
12    private JButton button4;
13    private JButton button5;
14    private JButton button6;
15    private JButton button7;
16    private JButton button8;
17    private JButton button9;
18    private JButton buttonEqual;
19    private JButton buttonAdd;
20    private JButton buttonMultiply;
21    private JButton buttonSub;
22    private JButton buttonDivide;
23    private JButton buttonDot;
24    private JButton buttonMinusPlus;
25    private JButton buttonCLS;
26
27
28
29    enum CalcOP {NONE, ADD, SUB, MULTIPLY, DIVIDE};
30
```

Context menu options:

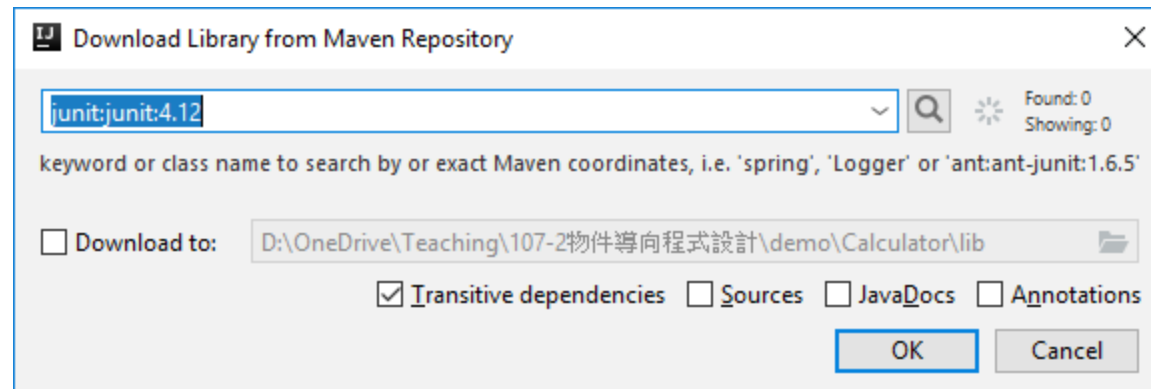
- Run 'CalculatorForm.main()' Ctrl+Shift+F10
- Debug 'CalculatorForm.main()'
- Run 'CalculatorForm.main()' with Coverage
- Create Test**
- Create subclass
- Add Javadoc
- Make package-private

Select Your Test Framework

- We use JUnit4 here.

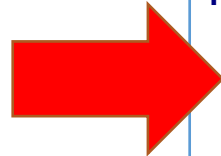


Auto Download Library

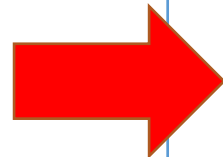


Using Junit Test & Runner

Junit Test Unit



Junit Runner

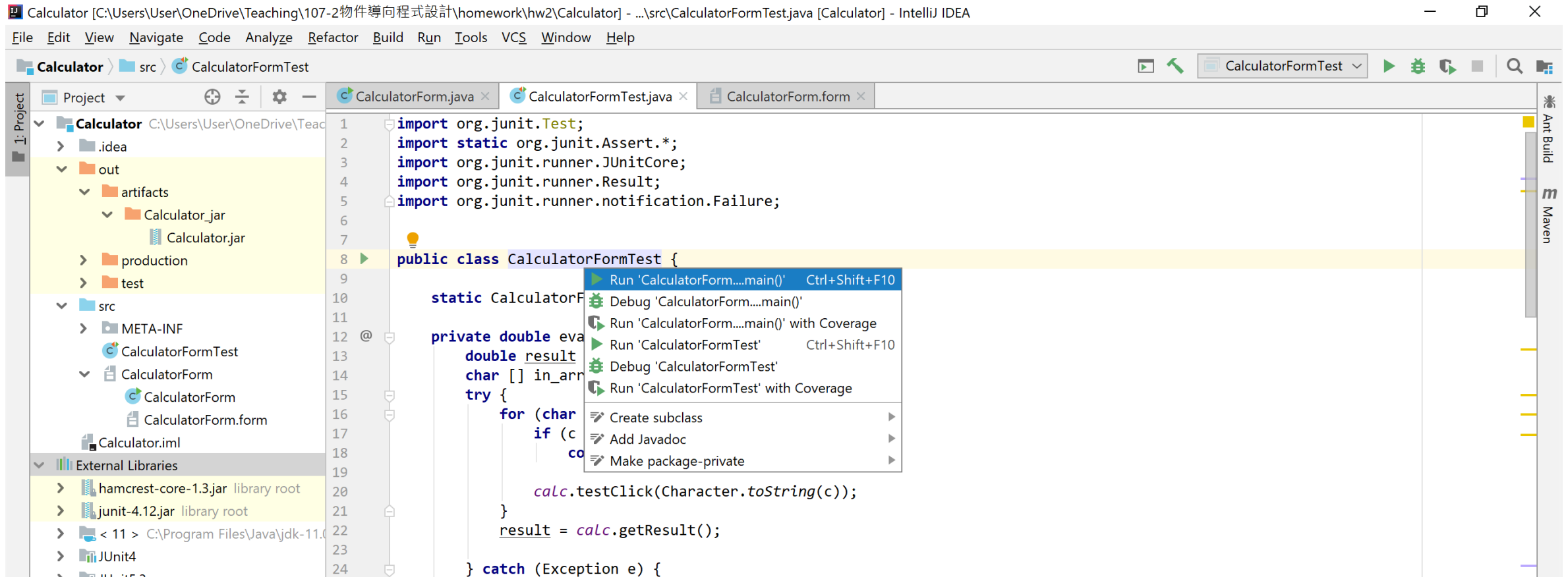


```
import org.junit.Test;
import static org.junit.Assert.*;
import org.junit.runner.RunWith;
import org.junit.runner.RunWith;
import org.junit.runner.notification.Failure;

public class CalculatorFormTest {
    static CalculatorForm calc = new CalculatorForm();
    @ Test
    public void testAddSub() {
        try {
            calc.testClick("CLEAR");
            calc.testClick("1"); calc.testClick("+");
            calc.testClick("2"); calc.testClick("=");
        }
        catch (Exception e) {
            System.out.println(e.getMessage());
        }
        double result = calc.getResult();
        assertEquals(3, result, 0);
    }
    public static void main(String[] args) {
        calc.showWindow();
        Result result = JUnitCore.runClasses(CalculatorFormTest.class);
        for (Failure failure : result.getFailures()) {
            System.out.println(failure.toString());
        }
        System.out.println(result.wasSuccessful());
    }
}
```


Run Your Test

- Press “Alt + Enter” on your test class



Reference

- <https://junit.org/junit5/docs/current/user-guide/>
- <https://www.tutorialspoint.com/junit/>