**JUnit 单元测试代码示例**

# TestCase示例

简单的计算类Calcuator类和它的JUnit测试类。

## 被测试类：Calcuator

package jtzen9.util;

/\*\*

\* 实现加减乘除的简单计算类

\*/

public class Calcuator {

public int add(int a,int b){

return a+b;

}

public int subtract(int a,int b){

return a-b;

}

public int multiply(int a,int b){

return a\*b;

}

public int divide(int a,int b){

return a/b;

}

}

## 简单测试类

package jtzen9.util;

import org.junit.Assert;

import org.junit.Test;

public class CalcuatorTest {

Calcuator calcuator;

@Test

public void testAdd() {

Calcuator = new Calcuator();

int result = Calcuator.add(2, 3);

Assert.assertEquals("加法有问题", 5, result);

/\*

\* "加法有问题"：期望值和实际值不一致时，显示的信息

\* 5 ：期望值

\* result ：实际值

\*/

}

@Test

public void testSubtract() {

Calcuator = new Calcuator();

int result = Calcuator.subtract(12, 2);

Assert.assertEquals("减法有问题", 10000, result); //故意设置减法期望值为10000

}

@Test

public void testMultiply() {

Calcuator = new Calcuator();

int result = Calcuator.multiply(2, 3);

Assert.assertEquals("乘法有问题", 6, result);

}

@Test

public void testDivide() {

Calcuator = new Calcuator();

int result = Calcuator.divide(6, 3);

Assert.assertEquals("除法有问题", 2, result);

}

}

## 参数化测试类

1. **import** **static** org.junit.Assert.\*;
3. **import** java.util.Arrays;
4. **import** java.util.Collection;
5. **import** org.junit.Test;
6. **import** org.junit.runner.RunWith;
7. **import** org.junit.runners.Parameterized;
8. **import** org.junit.runners.Parameterized.Parameters;
10. /\*\*
11. \* 测试类
12. \*/
13. @RunWith(Parameterized.**class**)//注意指定运行器
14. **public** **class** CalculatorTest {
15. **private** **static** Calculator calculator = **new** Calculator();
17. **private** **int** param;
18. **private** **int** paramm;
19. **private** **int** result;
21. //测试数据集合，方法名可以随意定义，返回类型可变，但是必须用@Parameters标注
22. @Parameters
23. **public** **static** Collection data() {
24. // 数组中,包含了传入参数和期望结果，数组参数顺序与构造函数参数顺序一致即可
25. **return** Arrays.asList(**new** Object[][] { { 2, 3, 6 }, { 0, 1, 0 },{ -3, 2, -6 }});
26. }
27. //构造函数，参数赋值顺序与测试数据集合一致
28. **public** CalculatorTest(**int** param, **int** paramm, **int** result) {
29. **this**.param = param;
30. **this**.paramm = paramm;
31. **this**.result = result;
32. }
34. @Test
35. **public** **void** square() {
36. calculator.multiply(param, paramm);
37. assertEquals(result, calculator.getResult());
38. }
39. }

# TestSuite示例

## 被测试类：Student

|  |
| --- |
| package com.phicomme.hu;  public class Student  {   private String name;   private String sex;   private int high;   private int age;   private String school;   public Student(String name, String sex ,int high, int age, String school)   {    this.name = name;    this.sex = sex;    this.high = high;    this.age = age;    this.school = school;   }   public String getName()   {    return name;   }   public void setName(String name)   {    this.name = name;   }   public String getSex()   {    return sex;   }   public void setSex(String sex)   {    this.sex = sex;   }   public int getHigh()   {    return high;   }   public void setHigh(int high)   {    this.high = high;   }   public int getAge()   {    return age;   }   public boolean setAge(int age)   {    if (age >25)    {     return false;    }    else    {     this.age = age;     return true;    }   }   public String getSchool()   {    return school;   }   public void setSchool(String school)   {    this.school = school;   }  } |

## 测试类

**测试类1：**

|  |
| --- |
| package com.phicomme.test;  import com.phicomme.hu.Student;  import junit.framework.TestCase;  public class StudentTest01 extends TestCase  {   Student testStudent;   //此方法在执行每一个测试方法之前（测试用例）之前调用   @Override   protected void setUp() throws Exception   {    // TODO Auto-generated method stub    super.setUp();    testStudent = new Student("djm", "boy", 178, 24, "华东政法");    System.out.println("setUp()");   }   //此方法在执行每一个测试方法之后调用   @Override   protected void tearDown() throws Exception   {    // TODO Auto-generated method stub    super.tearDown();    System.out.println("tearDown()");   }   //测试用例，测试Person对象的getSex()方法   public void testGetSex()   {    assertEquals("boy", testStudent.getSex());    System.out.println("testGetSex()");   }   //测试Person对象的getAge()方法   public void testGetAge()   {    assertEquals(24, testStudent.getAge());    System.out.println("testGetAge()");   }  } |

**测试类2：**

|  |
| --- |
| package com.phicomme.test;  import junit.framework.TestCase;  import com.phicomme.hu.Student;  public class StudentTest extends TestCase  {   private Student testStudent;   @Override   protected void setUp() throws Exception   {    // TODO Auto-generated method stub    super.setUp();    testStudent = new Student("steven\_hu", "boy", 170 , 23, "上海理工");   }   @Override   protected void tearDown() throws Exception   {    // TODO Auto-generated method stub    super.tearDown();   }   public void testSetage()   {    assertTrue(testStudent.setAge(21));   }   public void testGetSchool()   {    //预期值和实际值不一样，测试时出现失败(Failure)    assertEquals("南昌大学", testStudent.getSchool());   }   public void testGetName()   {    assertEquals("hdy", testStudent.getName());   }  } |

**两个测试用例一起测：**

当然，如果同时需要一起测试以上这两个测试类，可以通过TestSuite类实现，它相当于是一个套件，可以把所有测试类添进来一起运行测试；

代码如下：

|  |
| --- |
| package com.phicomme.test;  import com.phicomme.hu.StudentTest02;  import junit.framework.Test;  import junit.framework.TestSuite;  public class AllTest  {   //static PersonTest p = new PersonTest();   //static PersonTest p1 = new PersonTest();   public static Test suite()   {    TestSuite suite = new TestSuite("Test for com.phicomme.test");    //suite.addTest(p);    //suite.addTest(p1);    suite.addTestSuite(StudentTest.class);    suite.addTestSuite(StudentTest01.class);    return suite;   }  } |