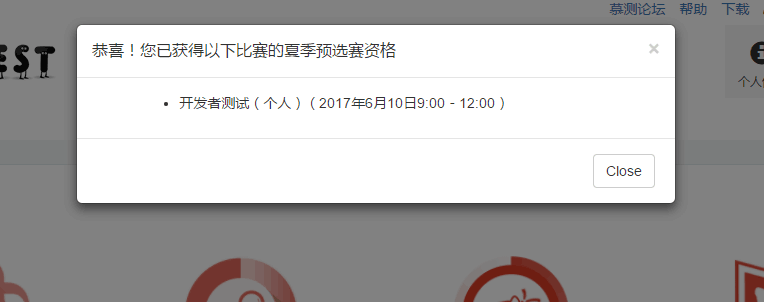
软件测试加分题

具体就是写一下覆盖测试吧，主要其他两项也不太会，但还是成功过了



代码如下

import static org.junit.Assert.\*;

import java.util.Random;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

public class NextdayTest {

@Before

public void setUp() throws Exception {

}

@After

public void tearDown() throws Exception {

}

@Test

public void testNextday() {

Date a = new Date(0,0, 0);

Nextday.nextDay(a);

}

@Test

public void testYear() {

try{

Year year = new Year(0);

fail("error");

}

catch( Exception e ){

assertTrue(true);

}

}

@Test

public void testYear2() {

Year a = new Year(2000);

assertTrue(a.isLeap());

}

@Test

public void testYear3() {

Year a = new Year(-1);

assertTrue(a.isLeap());

}

// @Test

// public void testYear11() {

// Year a = new Year(-2);

// assertTrue(a.isLeap());

// }

@Test

public void testYear4() {

Year a = new Year(2);

assertFalse(a.isLeap());

}

@Test

public void testYear5() {

Year a = new Year(2);

Year b = new Year(2);

assertTrue(a.equals(b));

}

@Test

public void testYear6() {

Year a = new Year(2);

Year b = new Year(3);

assertFalse(a.equals(b));

}

@Test

public void testYear7() {

Year a = new Year(-1);

a.increment();

}

@Test

public void testYear8() {

Year a = new Year(-1);

a.currentPos = 0;

a.isValid();

}

@Test

public void testYear9() {

Year a = new Year(2);

int b = 1;

assertFalse(a.equals(b));

}

@Test

public void testYear10() {

Year a = new Year(2);

a.getYear();

}

@Test

public void testMonth() {

try{

Month a = new Month(15, new Year(2000));

fail("");

}

catch( Exception e ){

assertTrue(true);

}

}

@Test

public void testMonth2() {

Month a = new Month(12, new Year(2000));

assertEquals(31,a.getMonthSize());

}

@Test

public void testMonth3() {

Month a = new Month(12, new Year(2001));

assertEquals(31,a.getMonthSize());

}

@Test

public void testMonth4() {

Month a = new Month(12, new Year(2000));

assertFalse(a.increment());

}

@Test

public void testMonth5() {

Month a = new Month(11, new Year(2000));

assertTrue(a.increment());

}

@Test

public void testMonth6() {

Month a = new Month(11, new Year(2000));

Month b = new Month(11, new Year(2000));

assertTrue(a.equals(b));

}

@Test

public void testMonth7() {

Month a = new Month(11, new Year(2000));

Month b = new Month(11, new Year(2001));

assertFalse(a.equals(b));

}

@Test

public void testMonth8() {

Month a = new Month(11, new Year(2000));

int b = 1;

assertFalse(a.equals(b));

}

@Test

public void testMonth9() {

Month a = new Month(12, null);

a.isValid();

}

@Test

public void testMonth11() {

Month a = new Month(0, new Year(2000));

a.isValid();

}

@Test

public void testMonth10() {

Month a = new Month(11, new Year(2000));

assertEquals(11,a.getMonth());

}

@Test

public void testDay() {

Day a = new Day(30, new Month( 11, new Year(2000)));

assertFalse(a.increment());

}

@Test

public void testDay2() {

Day a = new Day(11, new Month( 10, new Year(2000)));

assertTrue(a.increment());

}

@Test

public void testDay3() {

Day a = new Day(11, new Month( 10, new Year(2000)));

assertTrue(a.increment());

}

@Test

public void testDay4() {

try{

Day a = new Day(33, new Month( 10, new Year(2000)));

}

catch( Exception e ){

assertTrue(true);

}

}

@Test

public void testDay5() {

Day a = new Day(28, new Month( 2, new Year(1999)));

a.increment();

assertFalse(a.isValid());

}

@Test

public void testDay6() {

Day a = new Day(28, new Month( 2, new Year(2000)));

a.increment();

assertTrue(a.isValid());

}

@Test

public void testDay7() {

Day a = new Day(30, new Month( 10, new Year(2000)));

Day b = new Day(30, new Month( 10, new Year(2000)));

assertTrue(a.equals(b));

}

@Test

public void testDay8() {

Day a = new Day(30, new Month( 10, new Year(2000)));

Day b = new Day(29, new Month( 10, new Year(2000)));

assertFalse(a.equals(b));

}

@Test

public void testDay9() {

Day a = new Day(30, new Month( 10, new Year(2000)));

int b = 1;

assertFalse(a.equals(b));

}

@Test

public void testDay10() {

Day a = new Day(30, null);

int b = 1;

assertFalse(a.equals(b));

}

@Test

public void testDate() {

Date a = new Date(12,31, 1999);

a.increment();

}

public void testDate7() {

Date a = new Date(11,30, 1999);

a.increment();

}

@Test

public void testDate2() {

Date a = new Date(12,30, 1999);

a.increment();

}

@Test

public void testDate3() {

Date a = new Date(12,30, 1999);

Date b = new Date(12,30, 1999);

assertTrue(a.equals(b));

}

@Test

public void testDate4() {

Date a = new Date(12,30, 1999);

Date b = new Date(12,31, 1999);

assertFalse(a.equals(b));

}

@Test

public void testDate5() {

Date a = new Date(12,30, 1999);

a.toString();

}

@Test

public void testDate6() {

Date a = new Date(12,30, 1999);

int b = 1;

assertFalse(a.equals(b));

}

}