JUnit 4 on one page

JBRAINS.CA


```
package ca.jbrains.junit4.test;

import static org.hamcrest.Matchers.equalTo;
import static org.junit.Assert.*;

import java.util.*;
import java.util.regex.Pattern;

import org.junit.*;

public class SellOneItemTest {
    private Display display;
    private Sale sale;

    @Before
    public void wireSaleToDisplay() {
        display = new Display();
        sale = new Sale(display);
    }

    @Test
    public void productFound() throws Exception {
        sale.onBarcode("12345");
        assertEquals("$7.95", display.getText());
    }

    @Test
    public void productNotFound() throws Exception {
        sale.onBarcode("99999");
        assertThat(display.getText(),
                equalTo("No product found for 99999"));
    }

    @Test
    public void tryUsingGroupsBeforeInvokingMatches()
            throws Exception {
        try {
            Pattern.compile("(\\d\\d)(\\d)").matcher("762")
                    .group(1);
            fail("You read a match group before matching the regex?!");
        } catch (IllegalStateException expected) {
        }
    }

    @Test
    public void regularExpressionsMatch() throws Exception {
        assertTrue(Pattern.matches("\\d\\d\\d", "762"));
        assertFalse(Pattern.matches("\\d\\d\\d", "jbrains"));
    }

    @Test
    public void findItemInCollection() throws Exception {
        List<Display> displays = new ArrayList<Display>();

        displays.add(display);

        assertSame(display, displays.get(0));
    }

    @Test
    public void listCanStoreNullReferences()
            throws Exception {
        List<Object> list = Collections.<Object> singletonList(null);
        assertNotNull(list);
        assertNull(list.get(0));
    }

    public void thisIsNotATestBecauseItHasNoAnnotation()
            throws Exception {
        fail("The test runner won't run me.");
    }

    @Test
    public void thisIsNotATestBecauseItHasAParameter(
            @SuppressWarnings("unused") Object parameter)
            throws Exception {
        fail("This method results in an error in the test runner.");
    }

    @Test
    public String thisIsNotATestBecauseItReturnsAValue()
            throws Exception {
        fail("This method results in an error in the test runner.");
        return "fail";
    }
}
```

```
package ca.jbrains.junit4.test;

import junit.framework.TestCase;

import org.junit.internal.runners.JUnit38ClassRunner;
import org.junit.runner.RunWith;

@RunWith(JUnit38ClassRunner.class)
public class RunLegacyTestsTest extends TestCase {
    public void testAnotherProductFound() throws Exception {
        final Display display = new Display();
        final Sale sale = new Sale(display);
        sale.onBarcode("23456");
        assertEquals("$12.50", display.getText());
    }
}
```

No need to extend TestCase

Naming convention for test case classes

Fixture objects used by many tests

executes before each test to set up common test data

a test method

check the *expected* value against the *computed* value

JUnit reports unexpected exceptions

more flexible assertions

check that code throws the desired exception

naming convention for expected exception

check any boolean condition

check that the expected and computed result are the *same physical objects*

these are not tests

1. create some objects ("Arrange")

2. invoke a method ("Act")

3. check the result ("Assert")

check that references are null or not null

Run JUnit 3 tests, if you already have them