

单元与集成测试实验报告

- 姓名：孙旭
- 学号：161930127
- 日期：2021-10-14

实验环境

- jdk: 1.8
- maven: 4.0.0
- maven引入: apache commons-lang3: 3.11
- 源代码下载: apache commons-lang3: 3.12
- JUnit: 5.4

实验过程

1、按照实验手册对LinkedList进行测试，学习使用JUnit进行单元测试

复制jdk中的LinkedList，并修改类名称：

编写main方法驱动测试类：

```
1  import util.MyLinkedList;
2
3  public class TestMyLinkedList {
4      public static void main(String[] args){
5          TestMyLinkedList myTest = new TestMyLinkedList();
6          myTest.testAdd();
7      }
8
9      public void testAdd(){
10
11         MyLinkedList list = new MyLinkedList();
12         Integer i1 = new Integer(1);
13         Integer i2 = new Integer(2);
14         list.add(i1);
15         list.add(i2);
16         if(2== list.size() && list.contains(i1) && list.contains(i2)){
17             System.out.println("OK!");
18         }else {
19             System.out.println("Error int Add()!");
20         }
21     }
22 }
```

```
23 }  
24
```

使用JUnit进行单元测试:

```
1  package util;  
2  
3  import static org.junit.jupiter.api.Assertions.*;  
4  
5  class MyLinkedListTest extends Object {  
6  
7      @org.junit.jupiter.api.BeforeEach  
8      void setUp() throws Exception {  
9      }  
10  
11     @org.junit.jupiter.api.AfterEach  
12     void tearDown() throws Exception{  
13     }  
14  
15     @org.junit.jupiter.api.Test  
16     void add() {  
17         MyLinkedList list = new MyLinkedList();  
18         Integer i1 = new Integer(1);  
19         Integer i2 = new Integer(2);  
20         list.add(i1);  
21         list.add(i2);  
22         assertEquals(2,list.size());  
23     }  
24  
25     @org.junit.jupiter.api.Test  
26     void remove() {  
27         MyLinkedList list = new MyLinkedList();  
28         Integer i1 = new Integer(1);  
29         Integer i2 = new Integer(2);  
30         list.add(i1);  
31         list.add(i2);  
32         list.remove(0);  
33         assertEquals(1,list.size());  
34         //assertEquals(2,list.size());  
35     }  
36  
37     @org.junit.jupiter.api.Test  
38     void push() {  
39         MyLinkedList list = new MyLinkedList();  
40         Integer i1 = new Integer(1);  
41         Integer i2 = new Integer(2);  
42         list.add(i1);  
43         list.add(i2);  
44         assertEquals(2,list.size());  
45     }  
46 }
```

2、引入maven和Apache commons类

maven配置文件: pom.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5         http://maven.apache.org/xsd/maven-4.0.0.xsd">
6     <modelVersion>4.0.0</modelVersion>
7
8     <groupId>groupId</groupId>
9     <artifactId>UnitTest</artifactId>
10    <version>1.0-SNAPSHOT</version>
11    <dependencies>
12        <dependency>
13            <groupId>org.apache.commons</groupId>
14            <artifactId>commons-lang3</artifactId>
15            <version>3.11</version>
16        </dependency>
17    </dependencies>
18
19    <properties>
20        <maven.compiler.source>8</maven.compiler.source>
21        <maven.compiler.target>8</maven.compiler.target>
22    </properties>
23 </project>
```

3、复制待测类并注入缺陷

测试类: lang3中的StringEscapeUtils类

用途: 将字符串进行各种转码到其他形式

修改该类的类名称为MyStringEscapeUtils

注入的缺陷:

```
1 public static final CharSequenceTranslator ESCAPE_HTML4 =
2     new AggregateTranslator(
3         //new LookupTranslator(EntityArrays.BASIC_ESCAPE()),
4         new LookupTranslator(EntityArrays.ISO8859_1_ESCAPE()),
5         new LookupTranslator(EntityArrays.HTML40_EXTENDED_ESCAPE())
6     );
```

注释掉该方法中的一个函数调用过程, 会导致该方法转码错误, 不会进行转码过程

4、编写main方法驱动测试类

测试类源代码：

```
1  import apache.MyStringEscapeUtils;
2
3  public class TestMyStringEscapeUtils {
4      public static void main(String[] args){
5          TestMyStringEscapeUtils myTest = new TestMyStringEscapeUtils();
6          myTest.testEscapeJava();
7          myTest.testEscapeHtml();
8          myTest.testEscapeXml();
9      }
10
11     public static void testEscapeJava(){
12         String testString = "测试字符串";
13         String outString = MyStringEscapeUtils.escapeJava(testString);
14         //System.out.println(outString);
15         //System.out.println("\u6D4B\u8BD5\u5B57\u7B26\u4E32");
16         if(outString.equals("\u6D4B\u8BD5\u5B57\u7B26\u4E32")){
17             System.out.println("escapeJava_OK");
18         }else{
19             System.out.println("escapeJava_ERROR");
20         }
21     }
22     public static void testEscapeHtml(){
23         String testString = "<test>测试String</test>";
24         String outString = MyStringEscapeUtils.escapeHtml4(testString);
25         //System.out.println(outString);
26         //System.out.println("&lt;test&gt;测试String&lt;/test&gt;");
27         if(outString.equals("&lt;test&gt;测试String&lt;/test&gt;")){
28             System.out.println("escapeHtml_OK");
29         }else{
30             System.out.println("escapeHtml_ERROR");
31         }
32     }
33     public static void testEscapeXml(){
34         String testString = "<test>测试String</test>";
35         String outString = MyStringEscapeUtils.escapeXml10(testString);
36         //System.out.println(outString);
37         //System.out.println("&lt;test&gt;测试String&lt;/test&gt;");
38         if(outString.equals("&lt;test&gt;测试String&lt;/test&gt;")){
39             System.out.println("escapeXml_OK");
40         }else{
41             System.out.println("escapeXml_ERROR");
42         }
43     }
44 }
45
```

运行结果：

```
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" ...
escapeJava_OK
escapeHtml_ERROR
escapeXml_OK

Process finished with exit code 0
```

5、使用JUnit进行单元测试

单元测试源代码

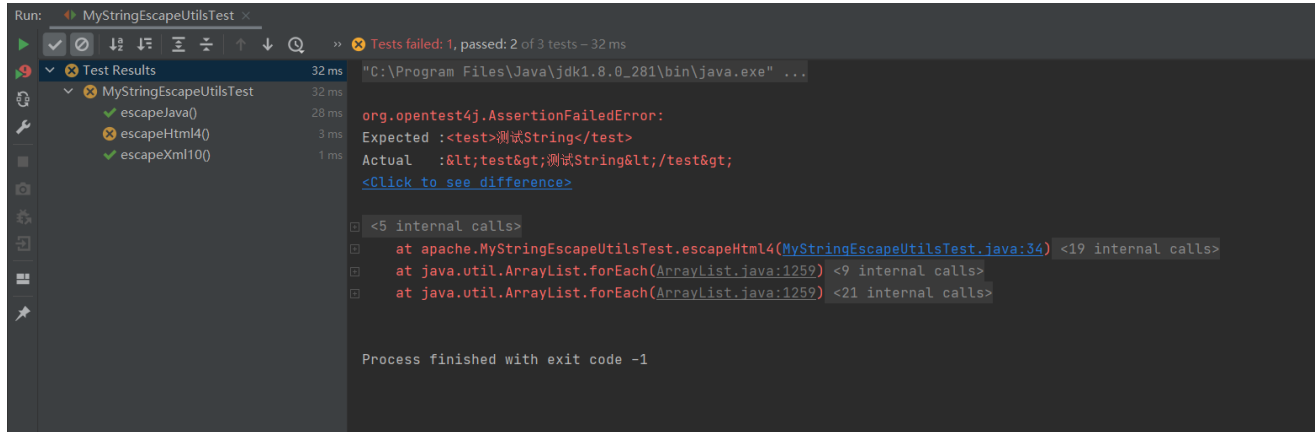
```
1 package apache;
2
3 import org.junit.jupiter.api.AfterEach;
4 import org.junit.jupiter.api.BeforeEach;
5 import org.junit.jupiter.api.Test;
6
7 import static org.junit.jupiter.api.Assertions.*;
8
9 class MyStringEscapeUtilsTest extends Object {
10
11     @BeforeEach
12     void setUp() {
13     }
14
15     @AfterEach
16     void tearDown() {
17     }
18
19     @Test
20     void escapeJava() {
21         String testString = "测试字符串";
22         String outString = MyStringEscapeUtils.escapeJava(testString);
23         //System.out.println(outString);
24         //System.out.println("\u6D4B\u8BD5\u5B57\u7B26\u4E32");
25         assertEquals(outString, "\u6D4B\u8BD5\u5B57\u7B26\u4E32");
26     }
27
28     @Test
29     void escapeHtml4() {
30         String testString = "<test>测试String</test>";
31         String outString = MyStringEscapeUtils.escapeHtml4(testString);
32         //System.out.println(outString);
33         //System.out.println("&lt;test&gt;测试String&lt;/test&gt;");
34         assertEquals(outString, "&lt;test&gt;测试String&lt;/test&gt;");
35     }
36
37     @Test
38     void escapeXml10() {
39         String testString = "<test>测试String</test>";
```

```

40     String outString = MyStringEscapeUtils.escapeXml10(testString);
41     //System.out.println(outString);
42     //System.out.println("&lt;test&gt;测试String&lt;/test&gt;");
43     assertEquals(outString, "&lt;test&gt;测试String&lt;/test&gt;");
44 }
45 }

```

单元测试运行结果：



实验结果与总结

无论是直接编写然后调用测试类还是使用JUnit进行单元测试，都可以获得测试结果。但是编写测试类相对更加繁琐，如果需要改变测试的方法，需要对源代码进行修改，不利于软件维护。而单元测试不仅可以直接单独测试每一个方法，和可以方便地进行打包测试。