单元与集成测试实验报告

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：

```java
import util.MyLinkedList;

public class TestMyLinkedList {
    public static void main(String[] args){
        TestMyLinkedList myTest = new TestMyLinkedList();
        myTest.testAdd();
    }

    public void testAdd(){

        MyLinkedList list = new MyLinkedList();
        Integer i1 = new Integer(1);
        Integer i2 = new Integer(2);
        list.add(i1);
        list.add(i2);
        if(2== list.size()&&list.contains(i1)&&list.contains(i2)){
            System.out.println("OK!");
        }else {
            System.out.println("Error int Add()!");
        }
    }

}
```

使用JUnit进行单元测试：

```java
package util;

import static org.junit.jupiter.api.Assertions.*;
```

```java
class MyLinkedListTest extends Object {

    @org.junit.jupiter.api.BeforeEach
    void setUp()throws Exception {
    }

    @org.junit.jupiter.api.AfterEach
    void tearDown() throws Exception{
    }

    @org.junit.jupiter.api.Test
    void add() {
        MyLinkedList list = new MyLinkedList();
        Integer i1 = new Integer(1);
        Integer i2 = new Integer(2);
        list.add(i1);
        list.add(i2);
        assertEquals(2,list.size());
    }

    @org.junit.jupiter.api.Test
    void remove() {
        MyLinkedList list = new MyLinkedList();
        Integer i1 = new Integer(1);
        Integer i2 = new Integer(2);
        list.add(i1);
        list.add(i2);
        list.remove(0);
        assertEquals(1,list.size());
        //assertEquals(2,list.size());
    }

    @org.junit.jupiter.api.Test
    void push() {
        MyLinkedList list = new MyLinkedList();
        Integer i1 = new Integer(1);
        Integer i2 = new Integer(2);
        list.add(i1);
        list.add(i2);
        assertEquals(2,list.size());
    }
}
```

2、引入maven和Apache commons类

maven配置文件：pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>groupId</groupId>
    <artifactId>UnitTest</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.11</version>
        </dependency>
    </dependencies>

    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
    </properties>

</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：

```java
public static final CharSequenceTranslator ESCAPE_HTML4 =
        new AggregateTranslator(
            //new LookupTranslator(EntityArrays.BASIC_ESCAPE()),
            new LookupTranslator(EntityArrays.ISO8859_1_ESCAPE()),
            new LookupTranslator(EntityArrays.HTML40_EXTENDED_ESCAPE())
        );
```

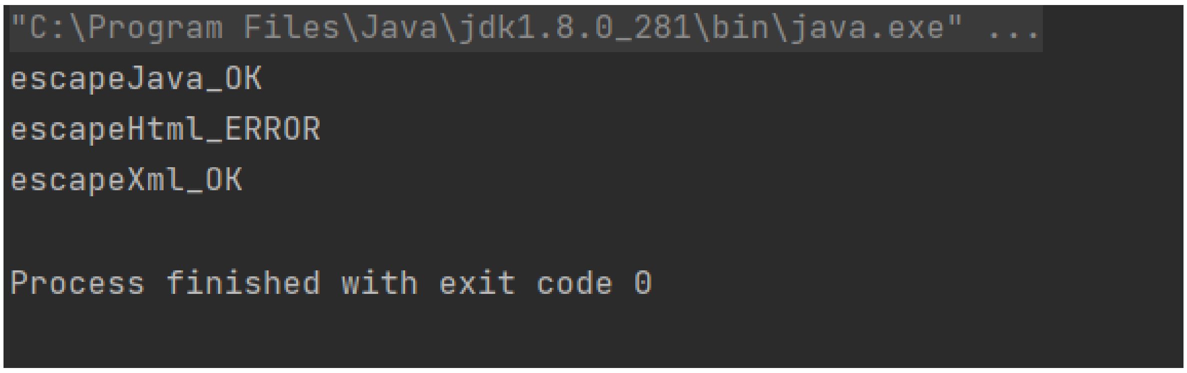
注释掉该方法中的一个函数调用过程，会导致该方法转码错误，不会进行转码过程

4、编写main方法驱动测试类

测试类源代码：

```java
import apache.MyStringEscapeUtils;

public class TestMyStringEscapeUtils {
    public static void main(String[] args){
        TestMyStringEscapeUtils myTest = new TestMyStringEscapeUtils();
        myTest.testEscapeJava();
        myTest.testEscapeHtml();
        myTest.testEscapeXml();
    }

    public static void testEscapeJava(){
        String testString = "测试字符串";
        String outString = MyStringEscapeUtils.escapeJava(testString);
        //System.out.println(outString);
        //System.out.println("\\u6D4B\\u8BD5\\u5B57\\u7B26\\u4E32");
        if(outString.equals("\\u6D4B\\u8BD5\\u5B57\\u7B26\\u4E32")){
            System.out.println("escapeJava_OK");
        }else{
            System.out.println("escapeJava_ERROR");
        }
    }
    public static void testEscapeHtml(){
        String testString = "<test>测试String</test>";
        String outString = MyStringEscapeUtils.escapeHtml4(testString);
        //System.out.println(outString);
        //System.out.println("&lt;test&gt;测试String&lt;/test&gt;");
        if(outString.equals("&lt;test&gt;测试String&lt;/test&gt;")){
            System.out.println("escapeHtml_OK");
        }else{
            System.out.println("escapeHtml_ERROR");
        }
    }
    public static void testEscapeXml(){
        String testString = "<test>测试String</test>";
        String outString = MyStringEscapeUtils.escapeXml10(testString);
        //System.out.println(outString);
        //System.out.println("&lt;test&gt;测试String&lt;/test&gt;");
        if(outString.equals("&lt;test&gt;测试String&lt;/test&gt;")){
            System.out.println("escapeXml_OK");
        }else{
            System.out.println("escapeXml_ERROR");
        }
    }
}
```

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
```

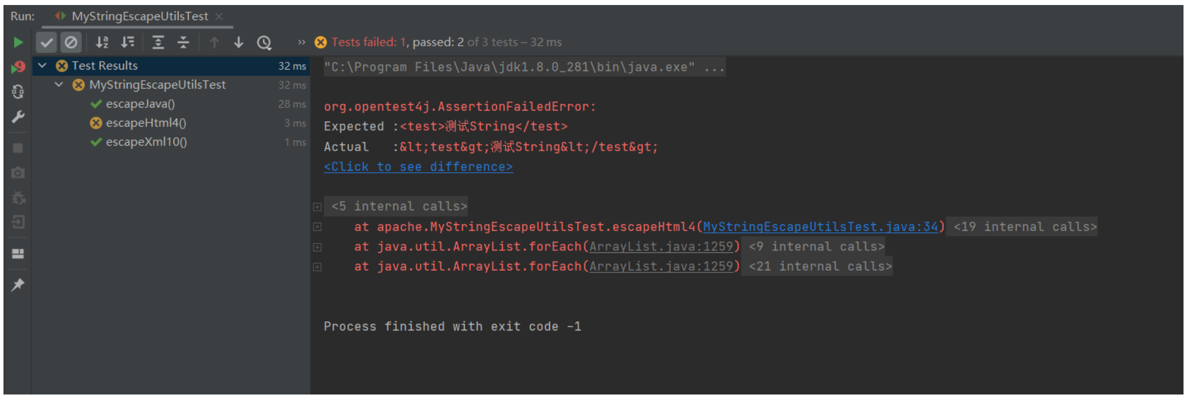
5、使用JUnit进行单元测试

单元测试源代码

```
package apache;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
```

```java
import static org.junit.jupiter.api.Assertions.*;

class MyStringEscapeUtilsTest extends Object {

    @BeforeEach
    void setUp() {
    }

    @AfterEach
    void tearDown() {
    }

    @Test
    void escapeJava() {
        String testString = "测试字符串";
        String outString = MyStringEscapeUtils.escapeJava(testString);
        //System.out.println(outString);
        //System.out.println("\\u6D4B\\u8BD5\\u5B57\\u7B26\\u4E32");
        assertEquals(outString,"\\u6D4B\\u8BD5\\u5B57\\u7B26\\u4E32");
    }

    @Test
    void escapeHtml4() {
        String testString = "<test>测试String</test>";
        String outString = MyStringEscapeUtils.escapeHtml4(testString);
        //System.out.println(outString);
        //System.out.println("&lt;test&gt;测试String&lt;/test&gt;");
        assertEquals(outString,"&lt;test&gt;测试String&lt;/test&gt;");
    }

    @Test
    void escapeXml10() {
        String testString = "<test>测试String</test>";
        String outString = MyStringEscapeUtils.escapeXml10(testString);
        //System.out.println(outString);
        //System.out.println("&lt;test&gt;测试String&lt;/test&gt;");
        assertEquals(outString,"&lt;test&gt;测试String&lt;/test&gt;");
    }
}
```

单元测试运行结果：



```
Tests failed: 1, passed: 2 of 3 tests – 32 ms

Test Results                  32 ms
  MyStringEscapeUtilsTest     32 ms
    ✓ escapeJava()            28 ms
    escapeHtml4()              3 ms
    ✓ escapeXml10()            1 ms
```

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

org.opentest4j.AssertionFailedError: 
Expected :<test>测试String</test>
Actual   :&lt;test&gt;测试String&lt;/test&gt;
<Click to see difference>

<5 internal calls>
    at apache.MyStringEscapeUtilsTest.escapeHtml4(MyStringEscapeUtilsTest.java:34) <19 internal calls>
    at java.util.ArrayList.forEach(ArrayList.java:1259) <9 internal calls>
    at java.util.ArrayList.forEach(ArrayList.java:1259) <21 internal calls>


Process finished with exit code -1
```

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。