重 庆 交 通 大 学

学生实验报告

实验课程名称：《软件测试》

开 课 实验室：软件实验室（南岸）

学　　院：信息学院

专　　业：计算机科学与技术

班　　级：2014 级 一 班

学　　号：631406010109

学 生 姓 名：郭文浩

指 导 教 师：何　伟

开 课 时 间：2016 至 2017 学年第 2 学期

总 成 绩	
教师签名	

实验项目名称		JUnit 测试			
姓名	郭文浩	学号	631406010109	实验日期	2017 年 6 月 14 日

教师评阅：

1:实验目的明确□A□B□C□D;　2:操作步骤正确□A□B□C□D;

3:实验报告规范□A□B□C□D;　4:实验结果符合要求□A□B□C□D

5:实验过程原始记录（数据、图表、计算等）符合要求□A□B□C□D;

6:实验分析总结全面□A□B□C□D;

签名：　　年　月　日

实验成绩：

一、实验目的

二、实验主要内容及原理

三、测试代码、测试方法及测试用例（可以以表格形式说明）

四、发现程序缺陷及修改方案

五、测试结果及分析(包括心得体会，本部分为重点，不能抄袭，不能复制)

一、实验目的

1、掌握 Junit 测试框架的使用。

2、掌握测试用例的编写。

二、实验内容及原理

1、实验内容

（1）学习 Junit 框架的使用

（2）使用 Junit 框架对程序设计实训 II 中所承担的代码进行单元测试。要求设计合理测试用例，用 Junit 进行测试，分析测试结果，并对错误代码进行修改。

2、实验原理

利用 Java 自身的 JUnit.jar 提供的方法进行单元测试。

三、测试代码、测试方法及测试用例

1、测试代码（简单计算器和字符串转换为实验指导书的例子，自我评价修改为我在实训 II 中的一个小功能的代码）

简单计算器

```
package com.gwh.JUnit;
/**
 * 2017-6-14 19:14:58<br>
 * <b>JUnit测试一</b><br>
 * 简单计算器，用于练习JUnit
 *
 * @author guowenhao
 *
 */
public class SampleCaculator {
    /**
     * 加法
     *
     * @param firstAddNum
     *            第一个加数
     * @param lastAddNum
     *            第二个加数
     * @return 和
     */
    public int add(int firstAddNum, int lastAddNum) {
        return firstAddNum + lastAddNum;
    }
```

```
    /**
     * 减法
     *
     * @param firstSubNum
     *            被减数
     * @param lastSubNum
     *            减数
     * @return 差
     */
    public int sub(int firstSubNum, int lastSubNum) {
        return firstSubNum - lastSubNum;
    }
}
```

Java 对象名转换为数据库格式

```
package com.gwh.JUnit;
import java.util.regex.Pattern;
import java.util.regex.Matcher;
/**
 * 2017-6-14 19:33:56<br>
 * <b>JUnit测试二</b><br>
 * 用于转换字符串格式<br>
 * 将Java对象字符串转换为数据库中的格式
 *
 * @author guowenhao
 *
 */
public class WordDealUtil {
    /**
     * 将Java对象名称（每个单词的头字母大写）<br>
     * 按照数据库命名的习惯进行格式化<br>
     * 格式化后的数据为小写字母<br>
     * 并且使用下划线分割命名单词
     *
     * @param name
     *            Java对象名称
     * @return 该名称的数据库格式
     */
    public static String wordFormat4DB(String name) {
        if (name == null) {
            return null;
        }

        Pattern p = Pattern.compile("[A-Z]");
```

```
        Matcher m = p.matcher(name);
        StringBuffer strbuff = new StringBuffer();
        while (m.find()) {
            if (m.start() != 0)
                m.appendReplacement(strbuff, "_" + m.group());
        }
        return m.appendTail(strbuff).toString().toLowerCase();
    }
}
```

实训II中的我评价判断单元代码

```
package com.gwh.JUnit;
/**
 * 2017-6-14 20:11:13<br>
 * <b>实训二</b>中的一个样例函数<br>
 * 该函数用于对提交的自我评价进行判断<br>
 * 如果为空：提醒用户为空，是否提交。是->提交 不是->不提交<br>
 * 如果不为空：提交。<br>
 * <b>注：</b> 函数源码为JavaScript语言编写<br>
 * 现转换为Java语言描述<br>
 * 稍微改变一下语法，拿来测试使用<br>
 * <b>js函数源码：</b>
 * <textarea>
 *  function isNull() {
        if (document.getElementById("selfAssess").value.trim().length == 0) {
            var flag = confirm("当前自我评价为空，是否保存？");
            if (flag) {
                document.getElementById('selfAssForm').action =
"/UniversityOfShaft/UpdateSelfAssess.do#test3";
                alert("自我评价修改为空！");
            } else {
                document.getElementById('selfAssForm').action =
"/UniversityOfShaft/listAssess.do#test3";
                alert("未提交");
            }
        } else {
            alert("自我评价修改成功！");
            document.getElementById('selfAssForm').action =
"/UniversityOfShaft/UpdateSelfAssess.do#test3";
        }
 * </textarea>
 * @author guowenhao
 *
 */
public class SelfAsses {
```

```
    public static String isNull(String selfAssess, boolean bool) {
        if (selfAssess == null || selfAssess.trim().length() == 0) {
            if (bool) {
                return "自我评价修改成功！";
            } else {
                return "未提交！";
            }
        } else {
            return "自我评价修改成功！";
        }
    }
}
```

2、测试方法

JUnit 单元测试。

3、测试用例

简单计算器

```
int result = sampleCaculator.add(10, 20);
assertEquals(30, result);
int result = sampleCaculator.sub(10, 20);
assertEquals(-10, result);
```

Java 对象名转换为数据库格式

```
@Test
public void testWordFormat4DB() {
    String target = "employeeInfo";
    String result = WordDealUtil.wordFormat4DB(target);
    assertEquals("employee_info", result);
    // fail("Not yet implemented");
}
@Test
public void testWordFormat4DBNull() {
    String target = null;
    String result = WordDealUtil.wordFormat4DB(target);
    assertNull(result);
}
@Test
public void testWordFormat4DBEmpty() {
    String target = "";
    String result = WordDealUtil.wordFormat4DB(target);
    assertEquals("", result);
}
```

```
@Test
public void testWordFormat4DBBegin() {
    String target = "EmployeeInfo";
    String result = WordDealUtil.wordFormat4DB(target);
    assertEquals("employee_info", result);
}
@Test
public void testWordFormat4DBEnd() {
    String target = "employeeInfoA";
    String result = WordDealUtil.wordFormat4DB(target);
    assertEquals("employee_info_a", result);
}
@Test
public void testWordFormat4DBTogether() {
    String target = "employeeAInfo";
    String result = WordDealUtil.wordFormat4DB(target);
    assertEquals("employee_a_info", result);
}
```

实训II 中的我评价判断单元代码

```
// 正常测试
@Test
public void testIsNull() {
    String target = "自我评价修改成功！";
    String result = SelfAsses.isNull("我很厉害！", true);
    assertEquals(target, result);
}
// 输入为空并且确认提交
@Test
public void testIsNullStringNullTrue() {
    String target = "自我评价修改成功！";
    String result = SelfAsses.isNull(null, true);
    assertEquals(target, result);
}
// 输入为空并且取消提交
@Test
public void testIsNullStringNullFalse() {
    String target = "未提交！";
    String result = SelfAsses.isNull(null, false);
    assertEquals(target, result);
}
// 输入为空串并且确认提交
@Test
public void testIsNullStringEmptyTrue() {
    String target = "自我评价修改成功！";
```

```
        String result = SelfAsses.isNull("", true);
        assertEquals(target, result);
    }
    // 输入为空串并且取消提交
    @Test
    public void testIsNullStringEmptyFalse() {
        String target = "未提交！";
        String result = SelfAsses.isNull("", false);
        assertEquals(target, result);
    }
    // 输入为空串并且确认提交
    @Test
    public void testIsNullStringSpaceTrue() {
        String target = "自我评价修改成功！";
        String result = SelfAsses.isNull("   ", true);
        assertEquals(target, result);
    }
    // 输入为空串并且取消提交
    @Test
    public void testIsNullStringSpaceFalse() {
        String target = "未提交！";
        String result = SelfAsses.isNull("   ", false);
        assertEquals(target, result);
    }
    // 输入字符串开始为空格
    @Test
    public void testIsNullStringBeginEmpty() {
        String target = "自我评价修改成功！";
        String result = SelfAsses.isNull("   我很厉害！", true);
        assertEquals(target, result);
    }
    // 输入字符串结束为空格
    @Test
    public void testIsNullStringEndEmpty() {
        String target = "自我评价修改成功！";
        String result = SelfAsses.isNull("我很厉害！   ", true);
        assertEquals(target, result);
    }
```

四、发现程序缺陷及修改方案

1、程序缺陷

这就只分析我自己的那块代码吧。我的那块代码开始时只是判断自我评价的

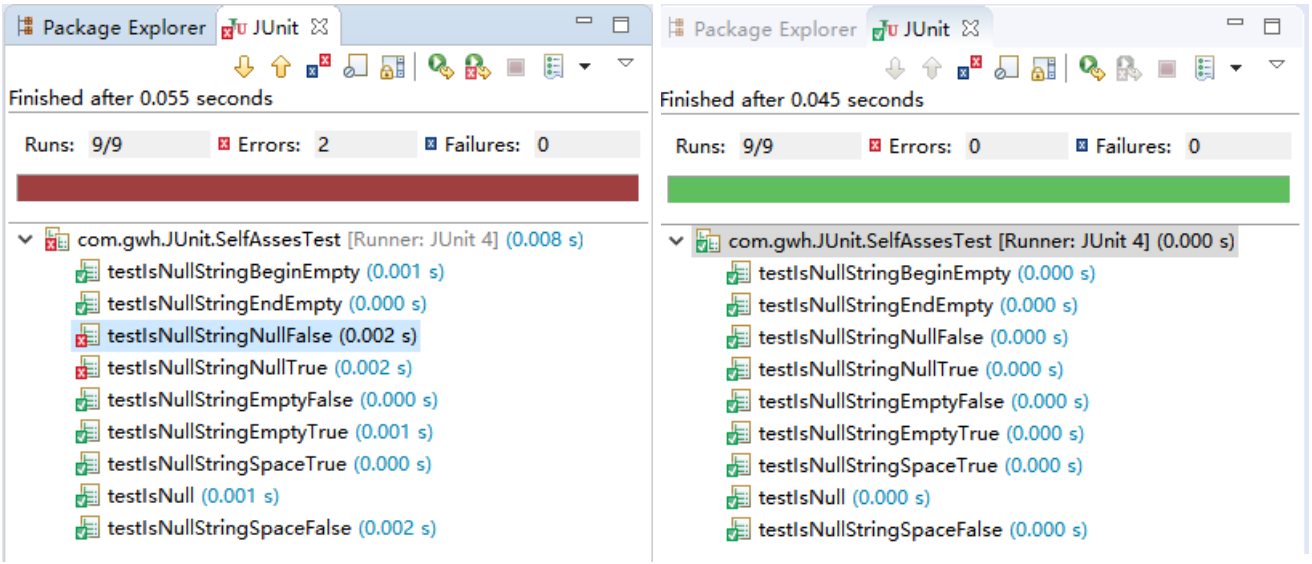
内容是否为空，而没有判断自我评价这个 string 是否为 null，如果为 null 的话程序运行时就会抛个异常。

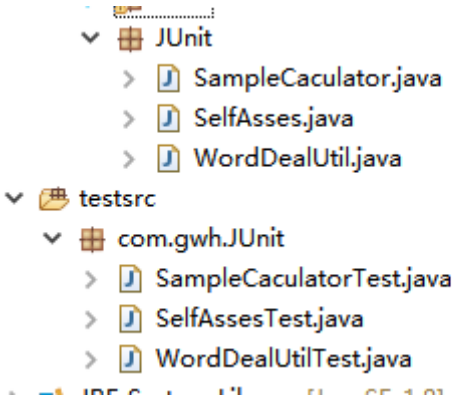
2、修改方案

```
if (selfAssess == null || selfAssess.trim().length() == 0) {
```

五、测试结果及分析

1、测试结果及分析





2、心得体会

这次实验体会就是 Java 真的强大，它的 JUnit 单元测试很好用，只要自己能想到足够完善的测试用例它就能保证程序单元的正确性。平时都是自己写个 test 测试类，在测试类中调用单元方法，但是每次这样做的时候只能保证程序是可以完成功能的，并不能保证程序是对的。以后要学着利用好 JUnit，把代码不断优化，尽可能的做到更好。