

# 实验报告

课程名称：\_\_\_\_软件工程\_\_\_\_

实验名称：\_\_\_\_白盒测试\_\_\_\_

专业班级：\_\_\_\_软件工程 2 班\_\_\_\_

学    号：\_\_\_\_1813075\_\_\_\_

姓    名：\_\_\_\_刘茵\_\_\_\_

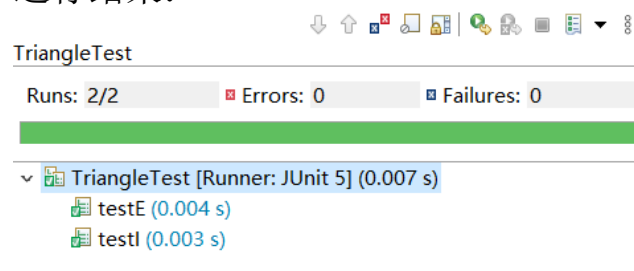
2020 年 11 月 12 日

## 实验三

实验名称	白盒测试实验		
实验地点	泰达五区	实验时间	2020/11/12
实验目的和要求			
<p>找一个经典算法题目及其 Java 代码（函数代码 50~100 行且不少于 3 个判断语句，不少于 2 个循环），作为被测函数。</p> <ol style="list-style-type: none"><li>1. 安装配置 JUnit，编写测试驱动代码，利用 JUnit 对被测函数进行单元测试。</li><li>2. 在被测函数的分支中增加 assert 语句，用于统计覆盖率，验证执行 JUnit 的结果</li><li>3. 对于其中的模块、函数单元，分别依据语句覆盖准则和分支覆盖准则，设计测试用例。利用 JUnit 批量执行这些测试用例，记录测试时间、用例、覆盖率，验证所设计的测试用例是否输出了所要求的覆盖准则的完全覆盖、是否发现缺陷。</li><li>4、（选做题）对于其中的模块、函数单元，依据原子谓词覆盖准则，设计测试用例。利用 JUnit 批量执行这些测试用例，记录测试时间、用例、覆盖率，验证所设计的测试用例是否输出了所要求的覆盖准则的完全覆盖、是否发现缺陷。</li></ol>			
实验环境			
<p>操作系统: Windows10 JUnit: 4.13.1 IDE: Eclipse IDE 2020 -09</p>			
实验过程			
<p>一、</p> <ol style="list-style-type: none"><li>1、安装配置 JUnit，编写测试驱动代码，利用 JUnit 对被测函数进行单元测试。</li></ol> <p>安装配置：将 Junit.jar 放入项目下的 lib 问价夹中，通过 build path 添加。并后期配置 JUnit 4/JUnit 5。</p> <p>测试驱动代码：</p>			

```
Triangle.java TriangleTest.java
1 package softtest1;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Before;
6 import org.junit.Test;
7
8 public class TriangleTest {
9     private Triangle tri=null;
10    @Before // 先运行构造函数，创建三角形
11    public void testBeforeClass() {
12        tri = new Triangle(2, 2, 3);
13    }
14    @Test
15    public void testE() { //检查是否是等边三角形
16        assertFalse(tri.isEquilatera());
17    }
18    @Test
19    public void testI() { //检查是否是等腰三角形
20        assertTrue(tri.isIsosceles());
21    }
22
23 }
```

运行结果：



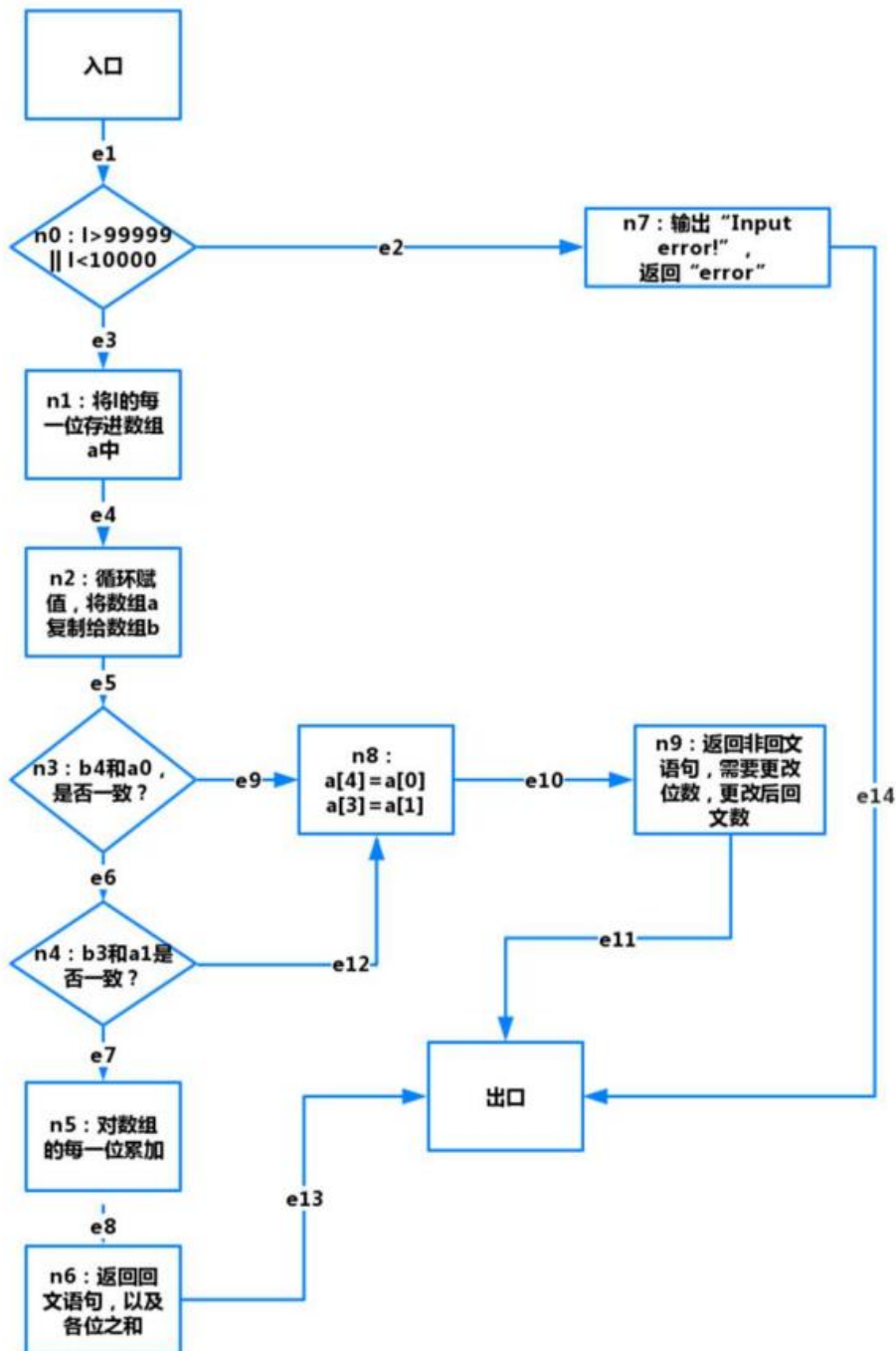
（测试成功）

2、在被测函数的分支中增加 assert 语句，用于统计覆盖率，验证执行 JUnit 的结果。

## 2.1 经典题目

一个 5 位数，判断它是不是回文数，如果是回文数，输出该回文数的个位的和，如果它不是回文数，输出将其改为回文数的需要的位数，将其改为回文数。

## 2.2 控制流图



## 2.3 程序代码

```

package softtest1;

import static org.junit.Assert.assertEquals;

import java.util.Scanner;

public class huiwen {
    public static String check(long l) {
        int[] a = new int[5];
        int[] b = new int[5];
        boolean is = false;
    }
  
```

```

//判断范围
if (1 > 99999 || 1 < 10000) {
    System.out.println("Input error!");
    return "error";
}
//拆分
for (int i = 4; i >= 0; i--) {
    a[i] = (int) (1 / (long) Math.pow(10, i));
    l = (1 % (long) Math.pow(10, i));
}
//反
for (int i = 0, j = 0; i < 5; i++, j++) {
    b[j] = a[i];
}
//判断
for (int i = 0, j = 4; i < 2; i++, j--) {
    if (a[i] != b[j]) {
        is = false;
        break;
    } else {
        is = true;
    }
}
if (!is) {
    int count = 0;
    for (int i = 0, j = 4; i < 2; i++, j--) {
        if (a[i] != b[j]) {
            count++;
        }
    }
    a[3] = a[1];
    a[4] = a[0];
    StringBuffer str = new StringBuffer();
    for (int s : a) {
        str.append(s);
    }
    System.out.println("is not a Palindrom!" + "需要改变的位数是: " + count + ", 更改过后为: " + str);
    return ("is not a Palindrom!" + "需要改变的位数是: " + count + ", 更改过后为: " + str);
} else {
    int sum = 0;
    for (int i = 0; i < 5; i++) {
        sum = sum + a[i];
    }
}

```

```

        System.out.println("is a Palindrom!" + "the
sum is " + sum);
        return ("is a Palindrom!" + "the sum is " +
sum);
    }
}
}

```

### 3.语句覆盖测试

#### 3.1 测试代码

```

@Test
public void check() {
    huiwen hui = new huiwen();
    assertEquals("is a Palindrom!" + "the sum is 9",hui.check(12321));
    assertEquals("is not a Palindrom!需要改变的位数是: 1. 更改过后为: 34143",hui.check(32143));
    assertEquals("error", hui.check(8937926));
}

```

#### 3.2 测试用例:

语句覆盖测试用例		
测试用例	I	覆盖节点
测试用例 1	{12321}	n0、n1、n2、n3、n4、n5、n6
测试用例 2	{32143}	n0、n1、n3、n4、n8、n9
测试用例 3	{8937926}	n0、n7

3.3 测试时间: 2020/11/19 22.42 花费 0.146s








3.4 覆盖率: 100%

3.4.1 语句覆盖测试的覆盖率 =  $||\frac{NODE(L_T)}{N_G}|| \times 100\%$  ,

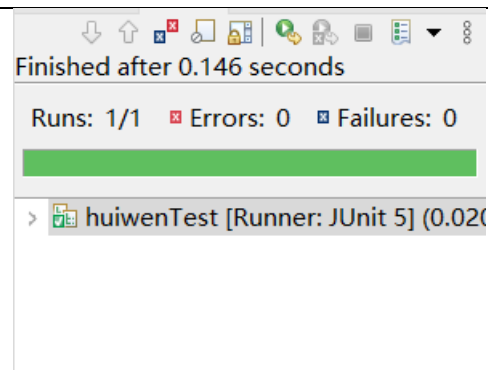
$N_G = \{n0, n1, n2, n3, n4, n5, n6, n7, n8, n9\}$ ,

$NODE(L_T) = \{n0, n1, n2, n3, n4, n5, n6, n7, n8, n9\}$ , 所以语句覆盖测试的覆盖率为 100%。

#### 3.4.2

Element	Coverage	Covered Instructions	Missed Instr...	Total Instruct...
▼ softtest1	 99.1 %	224	2	226
▼ src	 99.1 %	224	2	226
▼ softtest1	 99.1 %	224	2	226
▼ huiwen.java	 99.0 %	204	2	206
> huiwen	 99.0 %	204	2	206
▼ huiwenTest.java	 100.0 %	20	0	20
> huiwenTest	 100.0 %	20	0	20

3.5 缺陷: 没有发现缺陷



## 四、分支覆盖测试

### 4.1 测试代码

```
@Test
public void check() {
    huiwen hui = new huiwen();
    assertEquals("error", hui.check(23));
    assertEquals("is a Palindrom!" + "the sum is 9", hui.check(12321));
    assertEquals("is not a Palindrom!需要改变的位数是: 1. 更改过后为: 34143", hui.check(32143));
    assertEquals("is not a Palindrom!需要改变的位数是: 2. 更改过后为: 22422", hui.check(13422));
}
```

### 4.2 测试用例

分支覆盖测试用例					
测试用例	l	n0?	n3?	n4?	覆盖有向边
测试用例 1	{12321}	真	真	真	e1、e3、e4、e5、e6、e7、e8、e13
测试用例 2	{32143}	真	真	假	e1、e3、e4、e5、e6、e12、e10、e11
测试用例 3	{23}	假	—	—	e1、e2、e14
测试用例 4	{13422}	真	假	—	e1、e3、e4、e5、e9、e10、e11

4.3 测试时间 2020/11/20 22: 47 花费时间: 0.192s

### 4.4 覆盖率

4.4.1 分支覆盖测试的覆盖率 =  $||\frac{EDGE(L_T)}{E_G}|| \times 100\%$

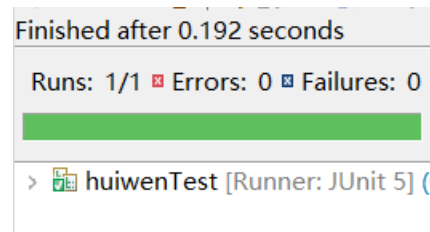
$E_G = \{e1, e2, e3, e4, e5, e6, e7, e8, e9, e10, e11, e12, e13, e14\}$   
 $EDGE(L_T) = \{e1, e2, e3, e4, e5, e6, e7, e8, e9, e10, e11, e12, e13, e14\}$

所以分支覆盖测试的覆盖率为 100%。

4.4.2 覆盖率为 100%

softtest1	100.0 %	226	0	226
src	100.0 %	226	0	226
softtest1	100.0 %	226	0	226
huiwen.java	100.0 %	202	0	202
huiwen	100.0 %	202	0	202
check(long)	100.0 %	199	0	199
huiwenTest.java	100.0 %	24	0	24
huiwenTest	100.0 %	24	0	24

## 4.5 缺陷



如图：没有发现缺陷

## 五、原子谓词覆盖测试

### 5.1 测试代码

```
@Test
public void check() {
    huiwen hui = new huiwen();
    assertEquals("error", hui.check(100000));
    assertEquals("error", hui.check(23));
    assertEquals("is a Palindrom!" + "the sum is 9", hui.check(12321));
    assertEquals("is not a Palindrom!需要改变的位数是: 1. 更改过后为: 34143", hui.check(32143));
    assertEquals("is not a Palindrom!需要改变的位数是: 2. 更改过后为: 22422", hui.check(13422));
}
```

### 5.2 测试用例










原子谓词覆盖测试用例					
测试用例	I	L>99999?	L<10000?	b4=b0?	b3=b1?
测试用例 1	{100000}	真	假	—	—
测试用例 2	{23}	假	真	—	—
测试用例 3	{12321}	假	假	真	真
测试用例 4	{32143}	假	假	真	假
测试用例 5	{13422}	假	假	假	—

5.3 测试时间 2020/11/20 23: 06 花费时间: 0.227s

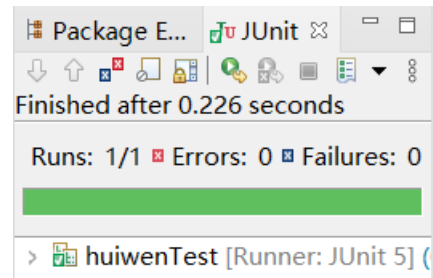
5.4 覆盖率 100%



huiwenTest (2020年11月20日 下午11:06:04)

Element	Coverage	Covered Instructions	Missed Instr...	Total Instruct...
▼ softtest1	 100.0 %	230	0	230
▼ src	 100.0 %	230	0	230
▼ softtest1	 100.0 %	230	0	230
▼ huiwen.java	 100.0 %	202	0	202
▼ huiwen	 100.0 %	202	0	202
check(long)	 100.0 %	199	0	199
▼ huiwenTest.java	 100.0 %	28	0	28
▼ huiwenTest	 100.0 %	28	0	28
check()	 100.0 %	25	0	25

## 5.5 缺陷



如图：没有缺陷

## 心得体会

掌握了基本白盒测试方法，学会使用 Junit 和 EcEmma。