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 | | | |
| 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。  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。 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 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。 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。 | | | |
| 实验环境 | | | |
| 操作系统：Windows10 Junit：4.13.1  IDE: Eclipse IDE 2020 -09 | | | |
| 实验过程 | | | |
| **一、** **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。** 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。 测试驱动代码：  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, 如果它不是会回文数，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** (l > 99999 || l < 10000) {  System.***out***.println("Input error!");  **return** "error";  }  //拆分  **for** (**int** i = 4; i >= 0; i--) {  a[i] = (**int**) (l / (**long**) Math.*pow*(10, i));  l = (l % (**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    3.2测试用例：   |  |  |  | | --- | --- | --- | | 语句覆盖测试用例 | | | | 测试用例 | l | 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 = || || × 100% ,  = {𝑛0, 𝑛1, 𝑛2, 𝑛3, 𝑛4, 𝑛5, 𝑛6, 𝑛7, 𝑛8, 𝑛9},  = {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  3.5缺陷：没有发现缺陷    四、分支覆盖测试  4.1测试代码  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 = || || × 100%  = {𝑒1, 𝑒2, 𝑒3, 𝑒4, 𝑒5, 𝑒6, 𝑒7, 𝑒8, 𝑒9, 𝑒10, 𝑒11, 𝑒12, 𝑒13, 𝑒14}  = {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%。  4.42 覆盖率为100%    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    5.2测试用例   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 原子谓词覆盖测试用例 | | | | | | | 测试用例 | l | 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%  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EclEmma。 | | | |