**《软件测试》**

**实验报告五 ——基路径测试方法**

**姓 名： 徐静 学 号： 2020112825**

**院 系： 计算机与信息学院 专 业：数字媒体技术\_\_\_\_\_**

**实 验 室： 实验日期： 2021/5/07**

**总评成绩： 审阅教师：**

### 一、实验目的：

1. 能够运用功能测试、基于控制流和数据流的测试等软件测试的核心技术和原理，结合相关文献，对测试问题及其影响因素进行分析或计算，得出有效结论（设计出的测试用例要保证在测试中程序的语句覆盖100%，条件覆盖100%。）；
2. 学习逻辑覆盖准则；
3. 学习绘制程序图；
4. 学习基路径测试；
5. 学习测试用例评估方法。

### 二、实验环境：

IDEA2020,JUnit

### 三、基路径测试步骤

1. 根据程序代码创建程序图
2. 计算圈复杂度
3. 获取基路径集
4. 构造测试数据，覆盖基路径集

### 四、实验要求：

1. 对上面的remove方法，设计尽可能少的测试用例，要求达到 100% 语句覆盖率
2. 创建remove方法的程序图；
3. 根据创建的程序图，计算圈复杂度
4. 采用基路径测试方法，构造基路径集
5. 根据基路径方法，构造测试数据
6. 使用Junit5框架实现自动化测试，提交到码云仓库
7. 配置插件Pitest，评估测试用例有效性
8. 针对实验中的经验和教训，编写实验总结
9. 按照[实验报告模板](https://star.jmhui.com.cn/u/cms/www/202203/06160105hi3m.docx) 编写实验报告，以“学号-姓名-软件测试实验五”命名，提交到雨课堂“软件测试实验五

### **五、实验步骤与内容**

1. **题目：**

针对LinkedList的remove方法实现基本路径测试。

1. **实验准备：**

配置编译测试插件pitest：在build.gradle 中添加pitest插件，并配置pitest，如针对包”mimic”执行变异测试

plugins {

id "info.solidsoft.pitest" version "1.7.4"

}

pitest {

targetClasses = ['mimic.\*']

threads = 4

outputFormats = ['HTML']

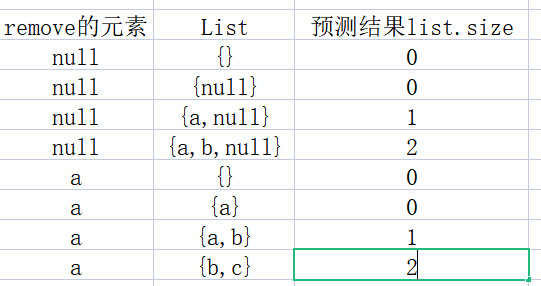
timestampedReports = true

junit5PluginVersion = '0.15'

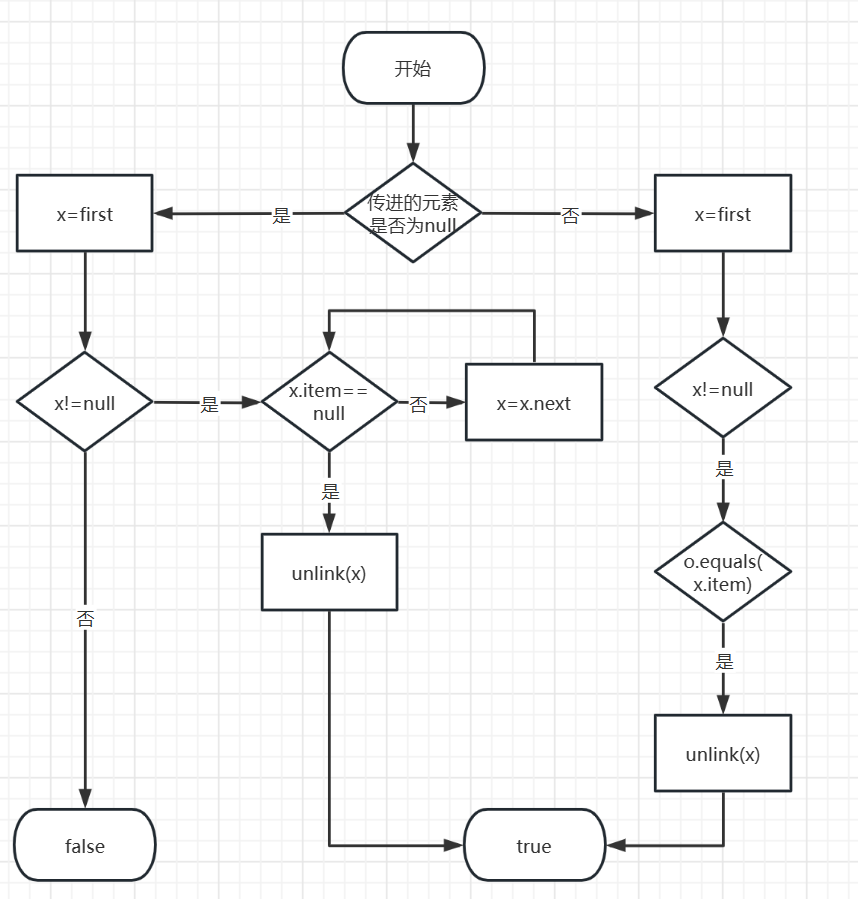
}

1. **测试用例：**

考虑到这是为remove函数设计测试用例，在参考别人的意见之后，这里采用list.size函数返回List的长度来判断元素有没有被删除成功：



流程图如下图所示：



1. **代码实现：**

|  |
| --- |
| **实现代码：**   1. public boolean remove(Object o) { 2. 01. if (o == null) { 3. 02. for (Node<E> x = first; x != null; x = x.next) { 4. 03. if (x.item == null) { 5. 04. unlink(x); 6. 05. return true; 7. } 8. } 9. 06. } else { 10. 07. for (Node<E> x = first; x != null; x = x.next) { 11. 08. if (o.equals(x.item)) { 12. 09. unlink(x); 13. 10. return true; 14. } 15. } 16. } 17. 11. return false; 18. }   测试代码：  package inittaskdemo3.app;  import org.junit.jupiter.api.DisplayName;  import org.junit.jupiter.api.Test;  import java.util.LinkedList;  import java.util.List;  import static org.junit.jupiter.api.Assertions.\*;  public class LinkedListTest {  @Test  @DisplayName("remove的元素为null并且list为{}")  void remove\_find\_null1() {  List<String> list = new LinkedList<>();  list.remove(null);  assertEquals(0,list.size());  }  @Test  @DisplayName("remove的元素为null并且list为{null}")  void remove\_find\_null2() {  List<String> list = new LinkedList<>();  list.add(null);  list.remove(null);  assertEquals(0,list.size());  }  @Test  @DisplayName("remove的元素为null并且list为{a,null}")  void remove\_find\_null3() {  List<String> list = new LinkedList<>();  list.add("a");  list.add(null);  list.remove(null);  assertEquals(1,list.size());  }  @Test  @DisplayName("remove的元素为null并且list为{a,b}")  void remove\_find\_null4() {  List<String> list = new LinkedList<>();  list.add("a");  list.add("b");  list.remove(null);  assertEquals(2,list.size());  }  @Test  @DisplayName("remove的元素为a并且list为{}")  void remove\_find\_a1() {  List<String> list = new LinkedList<>();  list.remove("a");  assertEquals(0,list.size());  }  @Test  @DisplayName("remove的元素为a并且list为{a}")  void remove\_find\_a2() {  List<String> list = new LinkedList<>();  list.add("a");  list.remove("a");  assertEquals(0,list.size());  }  @Test  @DisplayName("remove的元素为a并且list为{a,b}")  void remove\_find\_a3() {  List<String> list = new LinkedList<>();  list.add("a");  list.add("b");  list.remove("a");  assertEquals(1,list.size());  }  @Test  @DisplayName("remove的元素为a并且list为{b,c}")  void remove\_find\_a4() {  List<String> list = new LinkedList<>();  list.add("b");  list.add("c");  list.remove("a");  assertEquals(2,list.size());  }  }  测试结果：    覆盖率：    Pitest使用： |

### **六、结论分析与体会**

本次实验我了解了基本路径测试的实验方法，也对Pitest插件有了一个新的了解。通过这几次学习插件，我见识到了GRADLE配合各个插件工具的强大功能，同时也明白了计算机是一个学无止境的学科。

### **七、仓库地址**

https://github.com/cdwvcf/rjce1.git