

附件 1:

湖南科技大学计算机科学与工程学院

软件测试实验报告

专业班级: 计算机科学与技术七班

姓 名: 陈琪琪

学 号: 2102010629

指导教师: 何庭钦

时 间: 2024.04.08-2024.04.29

地 点: 逸夫楼 401

指导教师评语:

成绩:

等级:

签名: _____

年 月 日

实验名称	实验一、BMI 计算问题		
实验性质 (必修、选修)	选修	实验类型（验证、设计、创新、综合）	验证
实验课时	2	实验日期	2024-04-08
实验仪器设备以及实验软硬件要求	1. 硬件环境：PC 机一台 2. 软件环境：Java 编程环境：Java SDK + Eclipse + Junit4 3. 待测程序：计算机程序		
实验目的	1. 学习并了解 Junit 框架。 2. 掌握 Junit 的常见方法。 3. 学习测试用例的书写。		
实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法）			
一、实验原理			
JUnit 是一个用于编写和运行可重复的自动化测试的开源测试框架，它能够保证代码按预期工作。JUnit 被广泛用于工业和作为支架（从命令行）或 IDE（如 Eclipse）内单独的 Java 程序中。			
1. JUnit 的主要功能：			
<ul style="list-style-type: none">● 断言测试预期结果：JUnit 允许开发人员编写测试用例来验证代码输出是否符合预期。● 测试功能共享通用的测试数据：通过 JUnit，开发人员可以编写通用的测试数据，以支持多个测试用例。● 测试套件轻松地组织和运行测试：JUnit 允许开发人员将多个测试用例组织成测试套件，以便于管理和执行。● 图形和文本测试运行：JUnit 提供了图形和文本两种方式来运行和查看测试结果，使得开发人员可以直观地了解测试的进展和结果。			
2. JUnit 的特点：			
<ul style="list-style-type: none">● JUnit 可以与各种 IDE 集成，如 Eclipse、NetBeans 等。● JUnit 可以进行白盒测试和黑盒测试。● JUnit 可以测试任何 Java 应用程序，包括 J2EE 应用程序。● JUnit 支持参数化的测试和测试套件。● JUnit 可以测量测试用例的代码覆盖率。● JUnit 对于测试结果的判断是自动的，用户不需要手动进行判断。			
3. Junit 的架构：			
JUnit 最基本的结构是 Test Case，即一个测试用例。一个测试用例应该包括以下几个部分：			
<ul style="list-style-type: none">● 测试类：测试用例的标准 Java 类，包含测试用例的方法。● Fixture：为测试用例进行预设的环境。包括 SetUp 和 TearDown 方法。● Test Method：测试用例类中的测试方法，通过调用被测程序的方法来检查预期结果和实际结果是否一致。			
二、实验理论知识			
1. 固定片断法（Fixture）：用于创建和维护测试环境。它确保测试的可重复性和一致性，通过在测试开始前设置预定义状态，执行测试后恢复到原始状态。固定片断法包括设置阶段（setup）、执行阶段（exercise）、断言阶段（assertion）和清理阶段（teardown），帮助			

测试人员构建可靠的测试用例并准确评估软件功能和性能。

2. 异常处理：用于确保程序在面对异常情况时能够正确、可靠地处理。通过测试各种异常情况，包括输入无效数据、网络中断等，可以验证系统的健壮性和可靠性。
3. 条件测试：用于验证程序在不同条件下的行为是否符合预期。测试涵盖各种条件，如边界情况、边界外情况和特殊情况，以确保程序在各种条件下均能正确运行。
4. 参数注入测试：用于验证程序对输入参数的处理是否正确。通过注入各种不同类型和大小的参数，包括正常参数、异常参数和边界参数，测试程序对参数的接受、解析和处理能力，以确保程序的稳健性和安全性。

三、实验步骤与方法

1. 实验步骤

- (1) 搭建 Junit 单元测试环境,包括配置 JDK 和 Java 开发环境,使用 IDEA 集成开发平台完成 Junit 的安装和集成。
- (2) 根据《Junit 教程》学习 Junit 的用法。

2. 实验方法

(1) 固定片断法 (Fixture)

注解说明：

- **@BeforeClass**：在所有测试方法执行之前执行，通常用于设置一次性的资源或者初始化静态变量。
- **@AfterClass**：在所有测试方法执行完毕后执行，通常用于释放一次性资源或者进行清理工作。
- **@Before**：在每个测试方法执行之前执行，通常用于初始化测试所需的对象或者设置测试环境。
- **@After**：在每个测试方法执行完毕后执行，通常用于释放测试所需的资源或者进行清理工作。

程序代码：

```
1.      @Before
2.      public void setUp() throws Exception {
3.          testObj = new BMI(0, 0);
4.          System.out.println("Run @Before method");
5.      }
6.
7.      @After
8.      public void tearDown() throws Exception {
9.          testObj = null;
10.         System.out.println("Run @After method");
11.     }
12.
13.     @BeforeClass
14.     public static void prepareEnvironment(){
15.         System.out.println("Run @BeforeClass method");
16.     }
17.
```

```

18.     @AfterClass
19.     public static void RestoreEnvironment(){
20.         System.out.println("Run @AfterClass method");
21.     }

```

(2) 异常处理

注解说明：注解 `@Test(expected = IllegalArgumentException.class)` 为 junit4 版本中的调用方式（junit5 中使用 `assertThrows(IllegalArgumentException.class)`），表示该测试方法预期会抛出 `IllegalArgumentException` 异常。如果方法确实抛出了该异常，测试将被视为通过。

程序代码：

```

1.     @Test(expected = IllegalArgumentException.class)
2.     public void testNegative(){
3.         System.out.println("Run testNegative");
4.         testObj.setParams(-1, -1);
5.         testObj.getBMIType();
6.     }

```

(3) 条件测试

注解说明：通过给测试方法添加 `@Ignore` 注解，在运行测试时可以对有该注解的测试方法进行无视，以替代注释掉 `@Test` 注解的形式。

程序代码：

```

1.     @Ignore
2.     @Test
3.     public void testIgnore() {
4.         System.out.println("Run testIgnore");
5.         testObj.setParams(52.0, 1.58);
6.         String actualResult = testObj.getBMIType();
7.         String expectResult = "正常";
8.         assertTrue(expectResult == actualResult);
9.     }

```

(4) 参数注入测试

注解说明：注解 `@Parameterized.Parameters` 在 JUnit 中的主要使用场景是允许在测试方法中运行相同的测试逻辑，但使用不同的输入参数。这对于需要对不同输入值进行测试的情况非常有用，可以大大简化测试代码的编写。

程序代码：

```

1.     @Parameterized.Parameters(name="{index}:getBMIType[{0},{1}]=[{2}]")
2.     public static Collection testDataset(){
3.         return Arrays.asList(
4.             new Object[][]{
5.                 {45.0, 1.6, "偏瘦"},
6.                 {55.0, 1.6, "正常"},
7.                 {68.0, 1.6, "偏胖"},
8.                 {80.0, 1.6, "肥胖"}

```

```
9.         }  
10.    );  
11. }
```

实验结果与分析

1. 固定片断方法

运行: BMI Test ×

测试 已通过: 5, 已忽略: 3 共 8 个测试 - 1 毫秒

测试名称	耗时	执行步骤
BMI Test (BMI)	1 毫秒	Run @Before method
getBMIType_Normal	1 毫秒	Run getBMIType_Normal
testGetBMIType_SlightlyFat	0 毫秒	Run @After method
getBMIType_SlightlyFat	0 毫秒	
test	0 毫秒	
testGetBMIType_Fat	0 毫秒	
testIgnore	0 毫秒	
getBMIType_Thin	0 毫秒	
testNegative	0 毫秒	

2. 异常处理

运行: BMI Test.testNegative ×

测试 失败: 1 共 1 个测试 - 4 毫秒

测试名称	耗时	执行步骤
BMI Test (BMI)	4 毫秒	Run @BeforeClass method
testNegative	4 毫秒	Run @Before method Run testNegative Run @After method java.lang.AssertionError: Expected exception: java.lang.IllegalArgumentException <21 个内部行> Run @AfterClass method

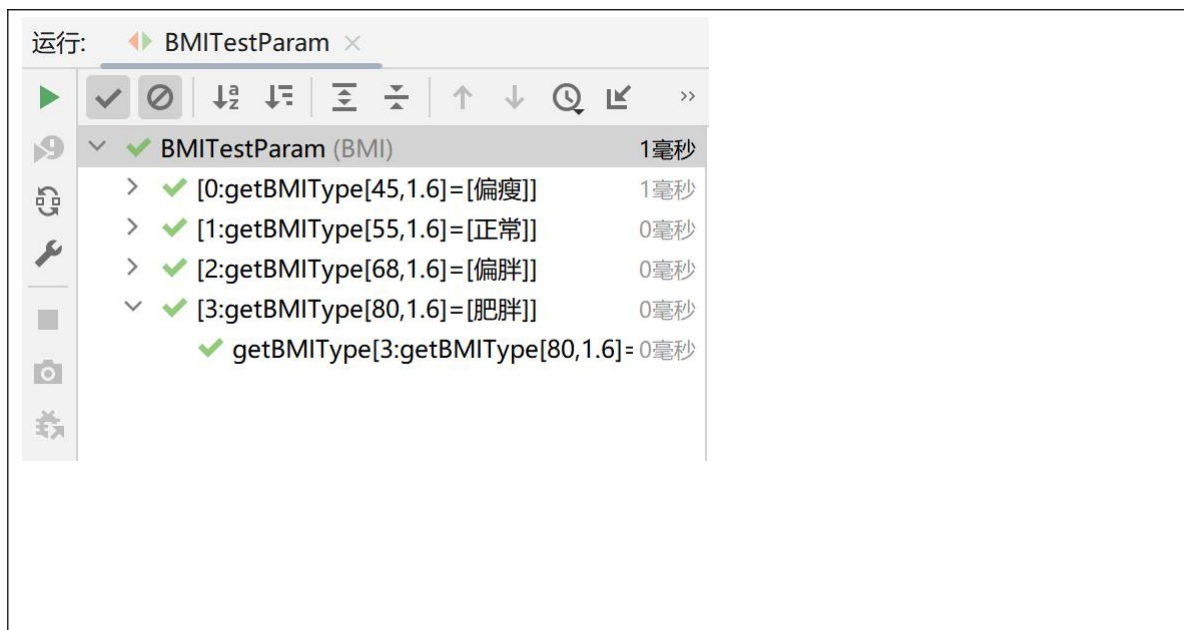
3. 条件测试

运行: BMI Test.testIgnore ×

测试 已通过: 1 共 1 个测试 - 0 毫秒

测试名称	耗时	执行步骤
BMI Test (BMI)	0 毫秒	Run @Before method
testIgnore	0 毫秒	Run testIgnore Run @After method

4. 参数注入方法



实验名称	实验二、黑盒测试和成对测试工具使用		
实验性质 (必修、选修)	选修	实验类型（验证、设计、创新、综合）	验证
实验课时	2	实验日期	2024-04-15
实验仪器设备以及实验软硬件要求	1. 硬件环境：PC 机一台 2. 软件环境：Java 编程环境：Java SDK + Eclipse + Junit4 3. 待测程序：JunitBMI 健康值计算判断程序 4. 软件环境：成对测试用例生成工具 PICT		
实验目的	1. 能熟练应用黑盒测试中的等价类划分方法设计测试用例 2. 能熟练应用黑盒测试中的边界值分析方法设计测试用例 3. 能熟练综合使用等价类划分和边界值分析解决黑盒测试需求 4. 熟练使用 Pairwise 测试工具 5. 能综合使用正交表法设计测试用例；熟练使用正交表查询工具		
实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法）			
一、实验原理与理论知识			
1. 等价类划分法			
等价类划分法是一种软件测试设计技术，用于通过将输入数据划分为若干等价类来减少测试用例的数量。每个等价类代表一组具有相似特性的输入数据，这些数据应当在相同的条件下产生类似的结果。通过测试每个等价类中的一个或几个代表值，可以有效地覆盖整个输入空间，减少冗余测试，提高测试效率。			
(1) 划分规则			
输入数据类型		划分等价类规则	
数据个数		1 个有效等价类：正确数据个数 2 个无效等价类：大于和小于数据个数	
集合		1 个有效等价类：正确数据集合 1 个或多个无效等价类	
符合某些规则的输入		多个有效等价类 若干个无效等价类	
取值范围		1 个有效等价类：正确取值范围 2 个无效等价类：大于和小于取值范围	
布尔值		1 个有效等价类：TRUE 1 个无效等价类：FALSE	
(2) 有效等价类：指输入数据符合系统规范或要求的情况下的等价类。代表了系统预期的正常操作条件。测试这些有效等价类有助于验证系统在正常输入条件下的功能和行为是否正确。根据本次实验用例设计的有效等价类如下：			
结果		体重（kg）	身高（m）
偏瘦		45	1.75
		45	1.80
		40	1.75
		40	1.80

正常	60	1.80
	60	1.70
	65	1.80
	65	1.70
偏胖	70	1.59
	70	1.63
	65	1.59
	65	1.63
肥胖	95	1.75
	95	1.80
	100	1.75
	100	1.80

(3) 无效等价类：指输入数据不符合系统规范或要求的情况下的等价类。它们代表了系统预期之外的异常或边界情况。测试这些无效等价类有助于验证系统在处理异常输入条件时的健壮性和容错能力。

根据本次实验用例设计的有效等价类如下：

结果	体重	身高
无效	-60	1.70
	60	-1.70
	-60	-1.70

2. 边界值分析法

边界值分析法是一种软件测试设计技术，用于确定测试用例的边界条件。它基于一种假设，即在输入值的边界处，程序可能会出现错误。因此，通过测试边界值，可以有效地发现潜在的错误，提高测试覆盖率和效率。

(1) 边界值分析法主要关注于对输入和输出的边界条件进行测试。通常，将输入值的边界情况划分为以下几类进行测试：

- 最小边界：最小可接受值的边界情况。
- 最大边界：最大可接受值的边界情况。
- 边界内部：在最小边界和最大边界之间的值。
- 边界外部：小于最小边界或大于最大边界的值。

(2) 应用场景：通常应用于数值型输入和输出的测试，例如整数、浮点数、日期等。通过测试边界值，可以覆盖各种情况，并发现可能存在的边界相关错误，如越界、溢出等。

(3) 方法优点：优点包括简单易懂、高效性和有效性。通过测试边界条件，可以在相对较少的测试用例下达到较高的测试覆盖率，帮助发现潜在的软件缺陷。

3. 成对测试

成对测试（Pairwise Testing）是一种软件测试设计技术，旨在通过测试输入参数的所有可能组合中的成对组合来有效地发现缺陷。这种方法基于这样的观察：大多数软件缺陷是由两个参数之间的交互引起的，因此，通过测试所有可能的参数对，可以在较少的测试用例下覆盖大部分潜在的缺陷。

(1) 基本概念

- 输入参数：软件功能通常依赖于多个输入参数，每个参数可能具有多个取值。

- 参数组合：将所有参数的取值进行组合，以生成测试用例。
- 成对组合：通过组合每两个参数的所有可能取值，生成测试用例，而不是所有参数的全组合。

(2) 应用场景

- 参数众多：当输入参数数量较多时，成对测试可以有效减少测试用例。
- 资源有限：在测试资源（如时间、人力）有限的情况下，成对测试可以确保高覆盖率。
- 交互复杂：当缺陷主要由参数间的交互引起时，成对测试能够更好地发现这些缺陷。

(3) 方法优点

- 高效覆盖：相比于测试所有参数的全组合，成对测试大幅减少了测试用例数量，但仍能有效发现大多数缺陷。
- 简化测试：减少了测试工作量和复杂度，节省了测试资源和时间。
- 发现交互缺陷：专注于参数间的交互，有助于发现由于参数组合引起的缺陷。

(4) 工具使用（PICT）

PICT（Pairwise Independent Combinatorial Testing）是由微软开发的一款用于成对测试的工具。它专门设计用于生成高效的测试用例集合，通过测试输入参数的成对组合来覆盖尽可能多的交互情况。PICT 工具能够显著减少测试用例的数量，同时保持较高的测试覆盖率，是成对测试的一个有力工具。

PICT 工具的特点：

- 高效生成测试用例：PICT 可以快速生成覆盖所有参数对组合的测试用例集，减少了全组合测试所需的用例数量。
- 支持复杂约束：允许用户指定参数间的约束条件，从而生成满足特定需求的测试用例。
- 易于使用：PICT 使用简单的文本文件作为输入，定义参数及其取值，并输出生成的测试用例。
- 灵活扩展：除了成对测试，PICT 还支持更高阶的组合测试，可以生成三元组合或更复杂的组合。

PICT 工具的优点：

- 节省时间和资源：通过生成最小化的测试用例集，PICT 工具显著减少了测试所需的时间和资源。
- 覆盖率高：尽管测试用例数量减少，PICT 工具能够确保高覆盖率，发现参数间的交互缺陷。
- 使用方便：简单的文本输入和命令行操作使得 PICT 工具易于集成到现有的测试流程中。

二、实验步骤和方法

1. 等价类划分法

(1) 实验步骤

- a. 分析需求，确定输入数据类型
- b. 使用规则划分有效、无效等价类
- c. 设计用例，覆盖有效等价类
- d. 设计用例，覆盖无效等价类

(2) 程序代码

```
1. // 有效等价类
2. @Test
3. public void testValidEquivalenceClass() {
4.     for (Object[] data : bmi.Eq_dataset_valid) {
```

```

5.         double weight = (double) data[0];
6.         double height = (double) data[1];
7.         String expected = (String) data[2];
8.         testGetBMIType(weight, height, expected);
9.     }
10. }
11.
12. // 无效等价类
13. @Test
14. public void testInvalidEquivalenceClass() {
15.     for (Object[] data : bmi.Eq_dataset_invalid) {
16.         double weight = (double) data[0];
17.         double height = (double) data[1];
18.         String expected = (String) data[2];
19.         testGetBMIType(weight, height, expected);
20.     }
21. }

```

2. 边界值分析法

(1) 实验步骤

- 分析需求，确定输入数据类型
- 使用规则划分有效、无效等价类
- 确定上点、离点和内点
- 设计用例，覆盖有效等价类
- 设计用例，覆盖无效等价类

(2) 程序代码

```

1. // 最小值测试
2. @Test
3. public void test_min() {
4.     for (Object[] data : bmi.Bound_dataset_min) {
5.         double weight = (double) data[0];
6.         double height = (double) data[1];
7.         String expected = (String) data[2];
8.         testGetBMIType(weight, height, expected);
9.     }
10. }
11. // 最大值测试
12. @Test
13. public void test_max() {
14.     for (Object[] data : bmi.Bound_dataset_max) {
15.         double weight = (double) data[0];
16.         double height = (double) data[1];
17.         String expected = (String) data[2];
18.         testGetBMIType(weight, height, expected);

```

```

19.     }
20. }
21. // 临界值测试
22. @Test
23. public void test_critical() {
24.     for (Object[] data : bmi.Bound_dataset_critical) {
25.         double weight = (double) data[0];
26.         double height = (double) data[1];
27.         String expected = (String) data[2];
28.         testGetBMIType(weight, height, expected);
29.     }
30. }
31. // 异常值测试
32. @Test
33. public void test_abnormal() {
34.     for (Object[] data : bmi.Bound_dataset_abnormal) {
35.         double weight = (double) data[0];
36.         double height = (double) data[1];
37.         String expected = (String) data[2];
38.         testGetBMIType(weight, height, expected);
39.     }
40. }
41. // 正常值测试
42. @Test
43. public void test_normal() {
44.     for (Object[] data : bmi.Bound_dataset_normal) {
45.         double weight = (double) data[0];
46.         double height = (double) data[1];
47.         String expected = (String) data[2];
48.         testGetBMIType(weight, height, expected);
49.     }
50. }

```

3. 成对测试工具使用

(1) 实验步骤

- a. 定义参数和取值：使用文本文件定义测试参数及其取值。例如：
 Parameter1: Value1, Value2, Value3
 Parameter2: Value1, Value2
 Parameter3: Value1, Value2, Value3, Value4
- b. 指定约束条件（可选）：如果存在参数间的特定约束条件，可以在文件中定义。例如：
 IF [Parameter1] = "Value1" THEN [Parameter2] ≠ "Value2"
- c. 运行 PICT 工具：使用命令行运行 PICT 工具，指定输入文件和输出文件路径。例如：
 pict input.txt > output.txt
- d. 查看生成的测试用例：PICT 会在输出文件中生成测试用例，每一行代表一个测试用例，

包含所有参数的取值组合。

实验结果与分析

1. 等价类划分法

(1) 有效等价类

运行: BMItest2.testValidEquivalenceClass	
» 测试 已通过: 1共 1 个测试 - 4毫秒	
✓ BMItest2 (BMI) 4毫秒	Pass. 体重:45.0kg, 身高:1.75m, result:偏瘦
✓ testValidEquivalenceClass 4毫秒	Pass. 体重:45.0kg, 身高:1.80m, result:偏瘦
	Pass. 体重:40.0kg, 身高:1.75m, result:偏瘦
	Pass. 体重:40.0kg, 身高:1.80m, result:偏瘦
	Pass. 体重:60.0kg, 身高:1.80m, result:正常
	Pass. 体重:60.0kg, 身高:1.70m, result:正常
	Pass. 体重:65.0kg, 身高:1.80m, result:正常
	Pass. 体重:65.0kg, 身高:1.70m, result:正常
	Pass. 体重:70.0kg, 身高:1.59m, result:偏胖
	Pass. 体重:70.0kg, 身高:1.63m, result:偏胖
	Pass. 体重:65.0kg, 身高:1.59m, result:偏胖
	Pass. 体重:65.0kg, 身高:1.63m, result:偏胖
	Pass. 体重:95.0kg, 身高:1.75m, result:肥胖
	Pass. 体重:95.0kg, 身高:1.80m, result:肥胖
	Pass. 体重:100.0kg, 身高:1.75m, result:肥胖
	Pass. 体重:100.0kg, 身高:1.80m, result:肥胖

(2) 无效等价类

运行: BMItest2.testInvalidEquivalenceClass	
» 测试 已通过: 1共 1 个测试 - 3毫秒	
✓ BMItest2 (BMI) 3毫秒	Fail. 体重:-60.0kg, 身高:1.70m, Expected:无效, Actual:错误
✓ testInvalidEquivalenceClass 3毫秒	Fail. 体重:60.0kg, 身高:-1.70m, Expected:无效, Actual:错误
	Fail. 体重:-60.0kg, 身高:-1.70m, Expected:无效, Actual:错误

2. 边界值分析法

(1) 最小值

✓ BMItest2 (BMI) 2毫秒	Pass. 体重:0.0kg, 身高:0.00m, result:错误
✓ test_min 2毫秒	

(2) 最大值

✓ BMItest2 (BMI) 2毫秒	Pass. 体重:200.0kg, 身高:2.50m, result:肥胖
✓ test_max 2毫秒	

(3) 临界值

» 测试 已通过: 1共 1 个测试 - 2毫秒	
✓ BMItest2 (BMI) 2毫秒	"C:\Program Files\Java\jdk1.8.0_171\bin\java.exe" ...
✓ test_critical 2毫秒	Fail. 体重:18.5kg, 身高:25.00m, Expected:错误, Actual:偏瘦

(4) 异常值

▼	✔ BMITest2 (BMI)	1毫秒	Pass. 体重:-1.0kg, 身高:25.00m, result:错误
	✔ test_abnormal	1毫秒	Pass. 体重:-1.0kg, 身高:-1.50m, result:错误

(5) 正常值

▼	✔ BMITest2 (BMI)	2毫秒	Pass. 体重:65.0kg, 身高:1.75m, result:正常
	✔ test_normal	2毫秒	

3. 成对测试工具使用

(1) 文本文件参数定义及其取值

操作系统类型:	win7, mac, win8
打印机类型:	EP, HP
打印类型:	打印双面, 打印单面

(2) 测试用例

	A	B	C
1	操作系统类型	打印机类型	打印类型
2	win7	HP	打印单面
3	win7	EP	打印双面
4	win8	HP	打印双面
5	mac	EP	打印单面
6	mac	HP	打印双面
7	win8	EP	打印单面
8			

实验名称	实验三、白盒测试 实验四、接口测试		
实验性质 (必修、选修)	选修	实验类型（验证、 设计、创新、综合）	验证
实验课时	2	实验日期	2024-04-22
实验仪器设备以及实验软硬件要求	1. 硬件环境：PC 机一台 2. 软件环境：Java 编程环境：Java SDK + Eclipse + Junit4 3. 编程环境：Visual Studio 4. 自动生成工具：Soot、Visustin v7 等 5. 待测程序：Demo		
实验目的	1. 巩固基于控制流白盒测试知识，对于给定的待测程序，能熟练应用基本控制流覆盖方法设计测试用例 2. 通过绘制程序控制流程图，实现对程序源代码的逻辑描述 3. 掌握逻辑短路对测试的影响 4. 培养严谨和系统的测试精神，学习测试用例的设计和分析 5. 掌握常用的静态测试工具		
实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法）			
一、实验原理与理论知识			
1. 白盒测试			
白盒测试是一种软件测试方法，也称为结构测试或逻辑驱动测试，它主要关注于内部结构和逻辑的测试。与黑盒测试不同，白盒测试是基于了解被测试软件的内部实现原理来设计测试用例的。			
(1) 白盒测试的特点			
<ul style="list-style-type: none">● 了解内部结构：测试人员需要了解被测试软件的内部结构、代码逻辑和程序控制流程。● 基于代码覆盖：通常依赖于代码覆盖标准，如语句覆盖、分支覆盖、路径覆盖等。● 深入测试：白盒测试可以深入到程序的每个执行路径，包括正常情况和异常情况。			
(2) 白盒测试的方法			
<ul style="list-style-type: none">a. 语句覆盖：程序中每一条语句至少被执行一次。b. 判定覆盖：每个判定的“真”或“假”都至少被执行一次，即程序中的每个分支至少执行一次。c. 条件覆盖：判定中的每个条件至少有一次取真值，有一次取假值。d. 判定条件覆盖：每个判定本身的判定结果（真假）至少满足一次。每个逻辑条件的可能值（真假）也至少被满足一次。e. 条件组合覆盖：被测试程序中的每个判定中的条件结果的所有可能组合至少执行一次。f. 路径覆盖：覆盖程序中所有可能的路径。			
(3) 白盒测试的应用场景			
<ul style="list-style-type: none">● 关键业务逻辑：对于涉及重要业务逻辑的模块或功能，白盒测试可以提供更深入的覆盖和测试。● 错误修复验证：在修复程序错误后，通过白盒测试可以验证修复是否完全生效。● 代码审查辅助：白盒测试可以作为代码审查的辅助手段，帮助审查人员发现潜在问题。			
(4) 白盒测试的优点			
<ul style="list-style-type: none">● 高效发现错误：通过深入测试内部结构，可以更容易地发现程序中的潜在缺陷。● 高度可控：测试人员可以精确地控制测试用例的设计和执行。			

- 提高代码质量：白盒测试可以帮助开发人员发现并修复代码中的问题，提高代码的质量和可靠性

2. 接口测试

接口测试是一种软件测试方法，用于验证软件系统中接口的正确性、功能性和性能。接口是不同软件组件之间相互通信的一种方式，通过接口，软件组件可以交换信息、调用功能和共享资源。

(1) 接口测试的特点

- 独立性：接口测试通常独立于用户界面，专注于测试接口的输入和输出。
- 自动化：由于接口测试不涉及用户界面，可以很容易实现自动化测试，提高测试效率。
- 灵活性：接口测试可以在开发过程任何阶段进行，从单元测试到集成测试和系统测试。
- 跨平台：接口测试可以跨越不同的平台和技术栈，例如，测试 Web 服务、RESTful API、SOAP API 等。
- 高效性能：接口测试可以测试软件的性能、负载和稳定性，发现潜在的性能问题。

(2) 接口测试的内容

- 功能测试：验证接口的功能是否符合预期，包括输入验证、参数验证、状态转换等。
- 数据验证：验证接口的数据传输是否正确，包括数据格式、数据类型、数据完整性等。
- 性能测试：测试接口的响应时间、吞吐量和并发处理能力，评估接口的性能指标。
- 安全测试：验证接口的安全性，包括身份验证、授权、加密等方面的测试。
- 兼容性测试：验证接口在不同环境、不同设备和不同浏览器下的兼容性。

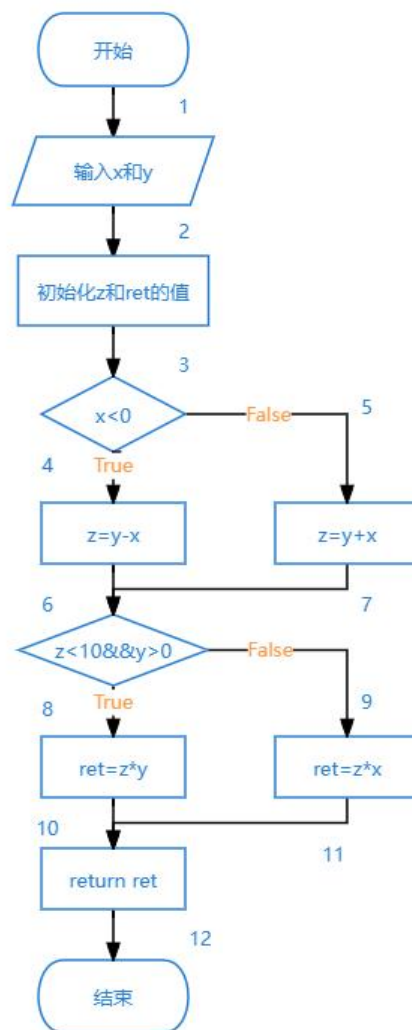
(3) 接口测试的重要性

- 保证系统稳定性：接口是系统不同模块之间的通信桥梁，确保接口的正确性可以保证系统整体的稳定性。
- 提高开发效率：通过自动化接口测试可以及早发现和修复问题，提高开发效率和质量。
- 保护数据安全：接口测试可以验证数据传输和处理的安全性，保护用户数据不受损害。
- 优化用户体验：保证接口的性能和可靠性可以提升用户体验，降低用户投诉和流失率。

二、实验步骤和方法

1. 白盒测试

(1) 程序流程图



(2) 测试用例设计

说明：

设判定 $x < 0$ 为 P1，判定 $z < 10 \&\& y > 0$ 为 P2；

条件 $x < 0$ 为 C1，条件 $z < 10$ 为 C2，条件 $y > 0$ 为 C3。

语句覆盖

用例编号	用例输入	覆盖路径	预期输出
Test1	{x=7, y=7}	1-2-3-5-7-9-11-12	98
Test2	{x=-7, y=1}	1-2-3-4-6-8-10-12	8

判定覆盖

用例编号	用例输入	覆盖判定		覆盖路径	预期输出
		P1	P2		
Test1	{x=7, y=7}	F	F	1-2-3-5-7-9-11-12	98
Test1	{x=-7, y=1}	T	T	1-2-3-4-6-8-10-12	8

条件覆盖

用例编号	用例输入	覆盖条件			覆盖路径	预期输出
		C1	C2	C3		
Test1	{x=12, y=-1}	F	F	F	1-2-3-5-7-9-11-12	132
Test2	{x=-7, y=1}	T	T	T	1-2-3-4-6-8-10-12	8

判定条件覆盖

用例编号	用例输入	覆盖判定		覆盖条件			覆盖路径	预期输出
		P1	P2	C1	C2	C3		
Test1	{x=12, y=-1}	F	F	F	F	F	1-2-3-5-7-9-11-12	132
Test2	{x=-7, y=1}	T	T	T	T	T	1-2-3-4-6-8-10-12	8

条件组合覆盖

用例编号	用例输入	覆盖判定		覆盖条件			覆盖路径	预期输出
		P1	P2	C1	C2	C3		
Test1	{x=1, y=6}	F	T	F	T	T	1-2-3-5-7-8-10-12	42
Test2	{x=3, y=-1}	F	F	F	T	F	1-2-3-5-7-9-11-12	6
Test3	{x=-6, y=6}	T	F	T	F	T	1-2-3-4-6-9-11-12	-72
Test4	{x=15, y=-2}	F	F	F	F	F	1-2-3-5-7-9-11-12	195

(3) 基于 Junit 实现上述测试用例。并使用 **eclemma** 工具查看测试用例的覆盖情况，尽量达到全覆盖。最后导出测试结果报告。

用例编号	用例输入	执行条件	预期输出	实际输出
Test1	{x=7, y=7}	语句覆盖 判定覆盖	98	98
Test2	{x=-7, y=1}	语句覆盖 条件覆盖 判定覆盖 判定条件覆盖	8	8
Test3	{x=12, y=-1}	语句覆盖 条件覆盖 判定覆盖 判定条件覆盖	132	132
Test4	{x=1, y=6}	条件组合覆盖	42	42
Test5	{x=3, y=-1}	条件组合覆盖	6	6
Test6	{x=-6, y=6}	条件组合覆盖	-72	-72
Test7	{x=15, y=-2}	条件组合覆盖	195	195

(4) 程序代码

Demo:

```
1. public int getNum(int x, int y){
2.     int z = 1;
3.     int ret =0;
```

```
4.     if (x < 0) {
5.         z = y - x;
6.     } else {
7.         z = y + x;
8.     }
9.     if (z < 10 && y > 0) {
10.        ret = z * y;
11.    } else {
12.        ret = z * x;
13.    }
14.    return ret;
15. }
```

语句覆盖、判定覆盖、条件覆盖、判定条件覆盖和条件组合覆盖：

```
1.     @Test
2.     public void test_Statements() {
3.         for (Object[] data : demo.dataset_statement) {
4.             int x = (int) data[0];
5.             int y = (int) data[1];
6.             int expected = (int) data[2];
7.             testGetNum(x, y, expected);
8.         }
9.     }
10.
11.    @Test
12.    public void test_judge() {
13.        for (Object[] data : demo.dataset_judge) {
14.            int x = (int) data[0];
15.            int y = (int) data[1];
16.            int expected = (int) data[2];
17.            testGetNum(x, y, expected);
18.        }
19.    }
20.
21.    @Test
22.    public void test_condition() {
23.        for (Object[] data : demo.dataset_condition) {
24.            int x = (int) data[0];
25.            int y = (int) data[1];
26.            int expected = (int) data[2];
27.            testGetNum(x, y, expected);
28.        }
29.    }
30. }
```

```

31.     @Test
32.     public void test_judgeAndCondition() {
33.         for (Object[] data : demo.dataset_judgeAndCondition) {
34.             int x = (int) data[0];
35.             int y = (int) data[1];
36.             int expected = (int) data[2];
37.             testGetNum(x, y, expected);
38.         }
39.     }
40.
41.     @Test
42.     public void test_path() {
43.         for (Object[] data : demo.dataset_path) {
44.             int x = (int) data[0];
45.             int y = (int) data[1];
46.             int expected = (int) data[2];
47.             testGetNum(x, y, expected);
48.         }
49.     }

```

2. 接口测试

(1) 使用断言

```

1  // 判断响应状态码是否等于200
2  pm.test("Status code is 200", function () {
3  pm.response.to.have.status(200); });
4
5  // 判断响应体中是否包含指定的字符串
6  pm.test("Body matches string", function () {
7  pm.expect(pm.response.text()).to.include("string_you_want_to_search"); });
8
9  // 判断响应体数据是否等于指定的字符串
10 pm.test("Body is correct", function () {
11 pm.response.to.have.body("response_body_string"); });
12
13 // 校验响应的JSON数据
14 pm.test("Your test name", function () {
15 var jsonData = pm.response.json();
16 pm.expect(jsonData.value).to.eql(100); });
17
18 // 判断响应头中是否包含指定的头标签
19 pm.test("Content-Type is present", function () {
20 pm.response.to.have.header("Content-Type"); });
21

```

(2) 使用关联

GET

▼

http://www.weather.com.cn/data/sk/101010100.html

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTests●Settings

```
1 var jsonData = pm.response.json();
2 var city = jsonData.weatherinfo.city;
3 console.log("city="+city);
4
5
6 pm.globals.set("city",city);
7
```

GET

▼

http://www.baidu.com/s?wd={{city}}

Send

▼

Params●AuthorizationHeaders (8)BodyPre-request ScriptTestsSettingsCookies

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	wd	{{city}}			
	Key	Value	Description		

BodyCookies (6)Headers (12)Test Results

Status: 200 OKTime: 118 msSize: 1.85 KBSave as example

PrettyRawPreviewVisualizeHTML▼

实验结果与分析

1. 白盒测试

(1) 语句覆盖

Pass. x:7, y:7, result:98

Pass. x:-7, y:1, result:8

(2) 判定覆盖

Pass. x:7, y:7, result:98

Pass. x:-7, y:1, result:8

(3) 条件覆盖

Pass. x:12, y:-1, result:132

Pass. x:-7, y:1, result:8

(4) 判定条件覆盖

Pass. x:12, y:-1, result:132

Pass. x:-7, y:1, result:8

(5) 条件组合覆盖




Pass. x:1, y:6, result:42

Pass. x:3, y:-1, result:6

Pass. x:-6, y:6, result:-72

Pass. x:15, y:-2, result:195

(6) 使用 eclemma 工具查看测试用例的覆盖情况

Element ▲	Class, %	Method, %	Line, %
✓  org.example	100% (2/2)	100% (4/4)	100% (24/24)
 GetNum	100% (1/1)	100% (1/1)	100% (10/10)
 TestGetNum	100% (1/1)	100% (3/3)	100% (14/14)

2. 接口测试

(1) 使用断言

PASS

Status code is 200

FAIL

Body matches string | AssertionError: expected '{"weatherinfo":{"city":"北京","cityid":...}' to include 'string_you_want_to_search'

FAIL

Body is correct | AssertionError: expected response body to equal 'response_body_string' but got '{"weatherinfo":{"city":"北京","cityid":...}'


FAIL

Your test name | AssertionError: expected undefined to deeply equal 100

PASS

Content-Type is present

(2) 使用关联

Body Cookies (6) Headers (12) Test Results  Status: 200 OK Time: 118 ms Size: 1.85 KB


Pretty

Raw

Preview

Visualize

HTML ▼



1

<!DOCTYPE html>

2

<html lang="zh-CN">

3

4

<head>

5

 <meta charset="utf-8">

6

 <title>百度安全验证</title>

7

 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

8

 <meta name="apple-mobile-web-app-capable" content="yes">

9

 <meta name="apple-mobile-web-app-status-bar-style" content="black">

10

 <meta name="viewport">

实验名称	实验五&实验六、性能与自动化测试		
实验性质 (必修、选修)	选修	实验类型（验证、设计、创新、综合）	验证
实验课时	2	实验日期	2024-04-29
实验仪器设备以及实验软硬件要求	主流 PC 机一套，windows 操作系统、WEB 项目、selenium 、tomcat 、jmeter； 目标测试系统： 1. 站长工具网站(http://tool.chinaz.com/) 2. DBshop 电子商务网站 http://150.158.110.4/test01/123456 3. CSDN 官网： https://www.csdn.net		
实验目的	1. 掌握 selenium 测试工具的基本原理和方法； 2. 掌握 selenium IDE 测试； 3. 掌握性能测试的基本原理和方法； 4. 掌握 jmeter 测试工具使用方法 5. 掌握性能测试的分析方法		
实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法）			
一、实验原理和理论知识			
Selenium 是一种用于自动化 Web 应用程序测试的工具，它可以模拟用户在浏览器中的操作，如点击、输入文本、提交表单等，从而进行自动化测试。			
(1) 基本原理			
<ul style="list-style-type: none">● 浏览器控制：Selenium 通过驱动浏览器来执行测试操作。它可以与各种浏览器（如 Chrome、Firefox、Safari 等）进行交互，控制浏览器执行指定的操作。● 定位元素：Selenium 可以通过各种定位策略（如 ID、Class、XPath 等）来定位页面上的元素，例如按钮、文本框、链接等。● 模拟用户操作：一旦定位到页面元素，Selenium 可以模拟用户在浏览器中的操作，如点击、输入文本、提交表单等。● 验证结果：Selenium 可以获取页面上的元素属性、文本内容等，并进行断言验证，以确保页面行为符合预期。			
(2) 基本方法			
<ul style="list-style-type: none">● 初始化浏览器驱动：在测试脚本中初始化所需的浏览器驱动，如 ChromeDriver、FirefoxDriver 等。● 定位元素：使用定位器（Locator）来定位页面上的元素，可以通过 ID、Class、XPath 等方式进行定位。● 执行操作：对定位到的元素执行相应的操作，如点击、输入文本、提交表单等。● 验证结果：获取页面元素的属性或文本内容，并进行断言验证，以确认页面行为符合预期。● 清理环境：在测试结束后，释放资源，关闭浏览器驱动，清理测试环境。			
(3) 常用方法			
<ul style="list-style-type: none">● find_element(): 定位单个元素。● find_elements(): 定位多个元素。● click(): 点击元素。● send_keys(): 向输入框中输入文本。			

- `submit()`: 提交表单。
- `get_attribute()`: 获取元素的属性值。
- `text`: 获取元素的文本内容。

(4) 使用场景

- 功能测试: 验证 Web 应用程序的功能是否按预期工作。
- 回归测试: 自动运行以确保新功能不会破坏现有功能。
- 跨浏览器测试: 确保 Web 应用程序在不同浏览器上的兼容性。
- 性能测试: 通过自动化测试脚本模拟大量用户操作, 评估系统的性能指标。

二、实验步骤和方法

1. 功能自动化测试

要求用尽可能少的测试用例检测出尽可能多的软件缺陷:

- (1) 功能需求分析。选择目标网站 CSDN 官网 <https://www.csdn.net>, 根据软件需求, 选择主要功能进行测试;
- (2) 安装“selenium ID”插件。启动 edge 浏览器, 点击“扩展” -> 点击“获取 Microsoft Edge 扩展” -> 搜索“selenium ID”插件 -> 点击“获取”安装 selenium IDE 插件。
- (3) 视频录制。点击“selenium IDE”插件 -> 选择“Create a new project” -> 输入项目名称 -> 输入需要录制测试的网站 -> 点击“REC”按钮进行视频录制 -> 进行需求中的功能测试。



Welcome to Selenium IDE!
Version 3.17.0

What would you like to do?

[Record a new test in a new project](#)

[Open an existing project](#)

[Create a new project](#)

[Close Selenium IDE](#)

To learn more on Selenium IDE and how to use it visit the [the Selenium IDE project page](#).

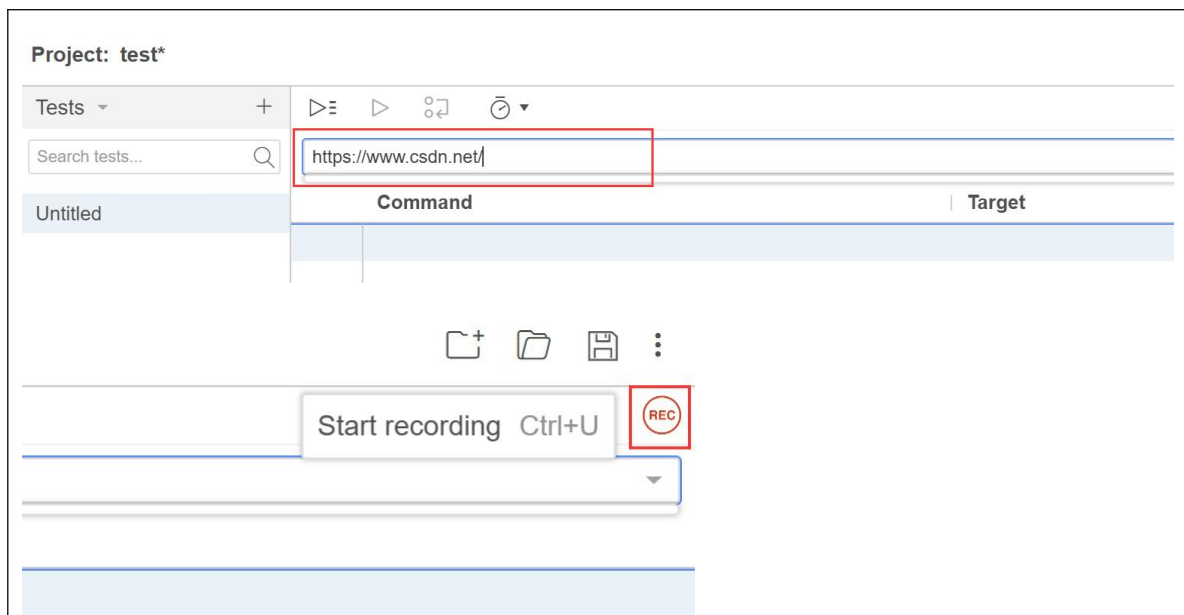
Name your new project

Please provide a name for your new project.

PROJECT NAME

test

You can change the name of your project at any time by clicking it and entering a new name.



(4) 查看视频录制过程中功能的测试情况。



(5) 使用 selenium 常用命令的进行功能自动化测试。

- 安装 Edge 浏览器驱动。下载并安装 Edge 浏览器的 WebDriver。可以从微软官网下载适用于自己的 Edge 版本的 WebDriver。下载完成后，将 WebDriver 的路径添加到系统环境变量中。
- 编写测试脚本。使用 Selenium 编写测试脚本，打开 Edge 浏览器并访问 <https://www.csdn.net/> 网站。以下是一个示例 Python 脚本：

```
1. from selenium import webdriver
2.
3. # 启动 Edge 浏览器
4. driver = webdriver.Edge()
5.
6. # 打开网页
7. driver.get("https://www.csdn.net/")
8.
9. # 进行其他操作，如定位元素、点击按钮等
10.
11. # 关闭浏览器
12. driver.quit()
```

- 执行测试。保存上述测试脚本为一个.py 文件，然后在命令行中运行该文件，即可执行测试。测试过程中，Edge 浏览器会自动打开并访问 <https://www.csdn.net/> 网站，然后执行脚本中定义的操作。

2. 性能自动化测试
- (1) 创建测试计划
- (2) 在测试计划中创建线程组
- (3) 增添 HTTP 请求
- (4) 在线程组中添加“聚合报告”和“察看结果树”来查看测试结果

线程组

名称: 访问百度

注释:

在取样器错误后要执行的动作

☒ 继续 ☐ 启动下一进程循环 ☐ 停止线程 ☐ 停止测试 ☐ 立即停止测试

线程属性

线程数: 50

Ramp-Up时间 (秒): 1

循环次数 ☐ 永远 1

☒ Same user on each iteration

☐ 延迟创建线程直到需要

实验结果与分析

1. 功能自动化测试

Selenium IDE - test*

Project: test*

Executing ▾

test* | https://www.csdn.net/

	Command	Target	Value
1	open	https://www.csdn.net/	
2	set window size	1391x762	
3	click	linkText=人工智能	
4	click	css=active-blog-nth-child(1) blog-text	
5	store window handle	root	
6	select window	handle=\${win6407}	
7	click	linkText=AI大模型探索之路-训练篇1: 大语言模型微调基础认知	
8	select window	handle=\${win6248}	

Runs: 1 Failures: 1

Log Reference

3. click on linkText=人工智能 OK 16:55:12

4. click on css=active-blog-nth-child(1) blog-text OK 16:55:14

5. storeWindowHandle on root OK 16:55:15

6. selectWindow on handle=\${win6407} OK 16:55:15

7. click on linkText=AI大模型探索之路-训练篇1: 大语言模型微调基础认知 Failed: Exceeded waiting time for new window to appear 2000ms 16:55:15

test ended with 1 error(s) 16:55:23

2. 性能自动化测试

(1) 查看结果树

(2) 聚合报告

聚合报告

名称：

聚合报告

注释：

所有数据写入一个文件

文件名

浏览...

显示日志内容：

☐ 仅错误日志

☐ 仅成功日志

配置

Label	# 样本	平均值	中位数	90% 百分位	95% 百分位	99% 百分位	最小值	最大值	异常 %	吞吐量	接收 KB...	发送 KB...
HTTP 请求 总体	50	110	86	219	219	225	46	225	0.00%	47.2/sec	115.02	5.44