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： 计算机科学与技术三班

姓　　名： 郭怀

学　　号： 1705010303

指导教师： 何庭钦

时　　间： 2020-5-17

地　　点： _____

指导教师评语：   签名：_____ 年　月　日

实验名称	BMI 测试		
实验性质 （必修、选修）		实验类型（验证、设计、创新、综合）	
实验课时		实验日期	2020-5-17
实验仪器设备以及实验软硬件要求	笔记本电脑一台		
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）能数量综合使用等价类划分和边界值分析解决黑盒测试需求； （4）学习测试用例的书写。 （5）学习并应用 junit 框架		
实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法） 由于内容过多，所以实验内容在后面单独列出。			
实验结果与分析 通过这次实验，加深了我对等价类测试和边界值测试的理解，通过编码也让我对 junit 的使用更加熟练， 能够为以后的测试提供帮助。			

测试用例设计

等价类划分法：

1.**有效等价类**（格式：体重 身高 结果）

1.1 偏瘦

45 1.75 偏瘦
45 1.8 偏瘦
40 1.75 偏瘦
40 1.8 偏瘦

1.2 正常

60 1.8 正常
60 1.7 正常
65 1.8 正常
65 1.7 正常

1.3 偏胖

70 1.59 偏胖
70 1.63 偏胖
65 1.59 偏胖
65 1.63 偏胖

1.4 肥胖

95 1.75 肥胖
95 1.8 肥胖
100 1.75 肥胖
100 1.8 肥胖

2.无效等价类

-60 1.7 无效
60 -1.7 无效
-60 -1.7 无效

一共 5 个等价类

边界值法：

输入参数的范围

1.身高的范围(单位为 m)

人类身高范围大约为[0.5,3.0]。刚出生的婴儿身高一般在 0.5 米以上，最高没有超过 3 米的人（最高的人是美国人罗伯特·沃德洛，身高 2.72）。

2.体重的范围(单位为 kg)

人类体重的范围大约为[2,500]。

测试用例

确定边界并取值

1.身高

身高的边界是 0.5 和 3.0。

在这两个边界处分别取一个长度为 2t 的邻域[0.5-t，0.5+t]、[3.0-t，3.0+t]，然后取邻域的两个端点值和 0.5、3.0 作为测试数据。

确定邻域的长度（即确定 t 的取值）：

(1).当 t=0.2 时

邻域[0.5-t，0.5+t]的端点为 0.3、0.7，[3.0-t，3.0+t]的端点为 2.8、3.2

(2).当 t=0.4 时

邻域[0.5-t，0.5+t]的端点为 0.1、0.9，[3.0-t，3.0+t]的端点为 2.6、3.4

所以身高的测试数据为{0.5、3.0、0.3、0.7、0.1、0.9、2.8、3.2、2.6、3.4}

2.体重

体重的边界是 2 和 500。

同样，在边界处取分别取一个长度为 2t 的邻域[2-t，2+t]、[500-t，500+t],然后取邻域的两个端点以及 2、500 作为测试数据。

(1).t=0.5 时

邻域[2-t，2+t]的端点为 1.5、2.5，邻域[500-t，500+t]的端点为 499.5、500.5。

(2).t=1 时

邻域[2-t，2+t]的端点为 1、3，邻域[500-t，500+t]的端点为 499、501。

所以体重的测试数据为{2、500、1.5、2.5、1、3、499.5、500.5、499、501}

设计测试用例

由上面得出身高和体重的测试数据为：

身高的测试数据为{0.5、3.0、0.3、0.7、0.1、0.9、2.8、3.2、2.6、3.4}

体重的测试数据为{2、500、1.5、2.5、1、3、499.5、500.5、499、501}

用这两个集合生成笛卡儿积（共 10*10=100 个）：

2.0 0.5 偏瘦	500.0 0.5 肥胖	1.5 0.5 偏瘦	2.5 0.5 偏瘦
2.0 3.0 偏瘦	500.0 3.0 肥胖	1.5 3.0 偏瘦	2.5 3.0 偏瘦
2.0 0.3 正常	500.0 0.3 肥胖	1.5 0.3 偏瘦	2.5 0.3 偏胖
2.0 0.7 偏瘦	500.0 0.7 肥胖	1.5 0.7 偏瘦	2.5 0.7 偏瘦
2.0 0.1 肥胖	500.0 0.1 肥胖	1.5 0.1 肥胖	2.5 0.1 肥胖
2.0 0.9 偏瘦	500.0 0.9 肥胖	1.5 0.9 偏瘦	2.5 0.9 偏瘦
2.0 2.8 偏瘦	500.0 2.8 肥胖	1.5 2.8 偏瘦	2.5 2.8 偏瘦
2.0 3.2 偏瘦	500.0 3.2 肥胖	1.5 3.2 偏瘦	2.5 3.2 偏瘦
2.0 2.6 偏瘦	500.0 2.6 肥胖	1.5 2.6 偏瘦	2.5 2.6 偏瘦
2.0 3.4 偏瘦	500.0 3.4 肥胖	1.5 3.4 偏瘦	2.5 3.4 偏瘦
1.0 0.5 偏瘦	3.0 0.5 偏瘦	499.5 0.5 肥胖	500.5 0.5 肥胖
1.0 3.0 偏瘦	3.0 3.0 偏瘦	499.5 3.0 肥胖	500.5 3.0 肥胖
1.0 0.3 偏瘦	3.0 0.3 肥胖	499.5 0.3 肥胖	500.5 0.3 肥胖
1.0 0.7 偏瘦	3.0 0.7 偏瘦	499.5 0.7 肥胖	500.5 0.7 肥胖
1.0 0.1 肥胖	3.0 0.1 肥胖	499.5 0.1 肥胖	500.5 0.1 肥胖
1.0 0.9 偏瘦	3.0 0.9 偏瘦	499.5 0.9 肥胖	500.5 0.9 肥胖
1.0 2.8 偏瘦	3.0 2.8 偏瘦	499.5 2.8 肥胖	500.5 2.8 肥胖
1.0 3.2 偏瘦	3.0 3.2 偏瘦	499.5 3.2 肥胖	500.5 3.2 肥胖
1.0 2.6 偏瘦	3.0 2.6 偏瘦	499.5 2.6 肥胖	500.5 2.6 肥胖
1.0 3.4 偏瘦	3.0 3.4 偏瘦	499.5 3.4 肥胖	500.5 3.4 肥胖
499.0 0.5 肥胖	501.0 0.5 肥胖		
499.0 3.0 肥胖	501.0 3.0 肥胖		
499.0 0.3 肥胖	501.0 0.3 肥胖		
499.0 0.7 肥胖	501.0 0.7 肥胖		
499.0 0.1 肥胖	501.0 0.1 肥胖		
499.0 0.9 肥胖	501.0 0.9 肥胖		
499.0 2.8 肥胖	501.0 2.8 肥胖		
499.0 3.2 肥胖	501.0 3.2 肥胖		
499.0 2.6 肥胖	501.0 2.6 肥胖		
499.0 3.4 肥胖	501.0 3.4 肥胖		

然后再从每部分中取第一个和最后一个测试用例来测试。

实际测试

1.等价类测试（代码键附件 EqClassTest.java）

针对上面等价类划分中的到的 5 个等价类的测试用例编写测试代码，并为每个等价类都创建一个接口用作分类的标签。代码如下：

测试前的一些操作：

```
static BMI bmi;//测试对象，static修饰，在整个测试过程中只创建一次测试对象

//测试开始前创建测试对象（整个测试过程中只执行一次）
@BeforeClass
public static void beforeClass() {
    bmi=new BMI();
}


//测试结束，销毁测试对象
@AfterClass
public static void afterClass() {
    bmi=null;
}

//测试一个用例之前，进行提示
@Before
public void beforeMethod() {
    System.out.println("测试一个用例.....");
}

//一个测试用例测试完成时进行提示
@After
public void afterMethod() {
    System.out.println("一个测试用例测试完成");
}
```

第一个等价类的测试用例：

```
//四个偏瘦的测试用例
@Category(ThinCase.class)
@Test
public void tinCase1() {
    bmi.setParams(45, 1.75);
    assertTrue("偏瘦".equals(bmi.getBMIType()));
}


@Category(ThinCase.class)
@Test
public void tinCase2() {
    bmi.setParams(45, 1.8);
    assertTrue("偏瘦".equals(bmi.getBMIType()));
}


@Category(ThinCase.class)
@Test
public void tinCase3() {
    bmi.setParams(40, 1.75);
    assertTrue("偏瘦".equals(bmi.getBMIType()));
}


@Category(ThinCase.class)
@Test
public void tinCase4() {
    bmi.setParams(40, 1.8);
    assertTrue("偏瘦".equals(bmi.getBMIType()));
}
```

第二个等价类的测试用例：

```
//四个体重“正常”的测试用例
@Category(NormCase.class)
@Test
public void normCase1() {
    bmi.setParams(60, 1.8);
    assertTrue("正常".equals(bmi.getBMIType()));
}


@Category(NormCase.class)
@Test
public void normCase2() {
    bmi.setParams(60, 1.7);
    assertTrue("正常".equals(bmi.getBMIType()));
}


@Category(NormCase.class)
@Test
public void normCase3() {
    bmi.setParams(65, 1.8);
    assertTrue("正常".equals(bmi.getBMIType()));
}

@Category(NormCase.class)
@Test
public void normCase4() {
    bmi.setParams(65, 1.7);
    assertTrue("正常".equals(bmi.getBMIType()));
}
```

第三个等价类的测试用例：

```
//四个偏胖的测试用例
@Category(FattyCase.class)
@Test
public void fattyCase1() {
    bmi.setParams(70, 1.59);
    assertTrue("偏胖".equals(bmi.getBMIType()));
}


@Category(FattyCase.class)
@Test
public void fattyCase2() {
    bmi.setParams(70, 1.63);
    assertTrue("偏胖".equals(bmi.getBMIType()));
}


@Category(FattyCase.class)
@Test
public void fattyCase3() {
    bmi.setParams(65, 1.59);
    assertTrue("偏胖".equals(bmi.getBMIType()));
}

@Category(FattyCase.class)
@Test
public void fattyCase4() {
    bmi.setParams(65, 1.63);
    assertTrue("偏胖".equals(bmi.getBMIType()));
}
```

第四个等价类的测试用例：

```
//四个肥胖测试用例
@Category(FatCase.class)
@Test
public void fatCase1() {
    bmi.setParams(95, 1.75);
    assertTrue("肥胖".equals(bmi.getBMIType()));
}


@Category(FatCase.class)
@Test
public void fatCase2() {
    bmi.setParams(95, 1.8);
    assertTrue("肥胖".equals(bmi.getBMIType()));
}


@Category(FatCase.class)
@Test
public void fatCase3() {
    bmi.setParams(100, 1.75);
    assertTrue("肥胖".equals(bmi.getBMIType()));
}

@Category(FatCase.class)
@Test
public void fatCase4() {
    bmi.setParams(100, 1.8);
    assertTrue("肥胖".equals(bmi.getBMIType()));
}
```

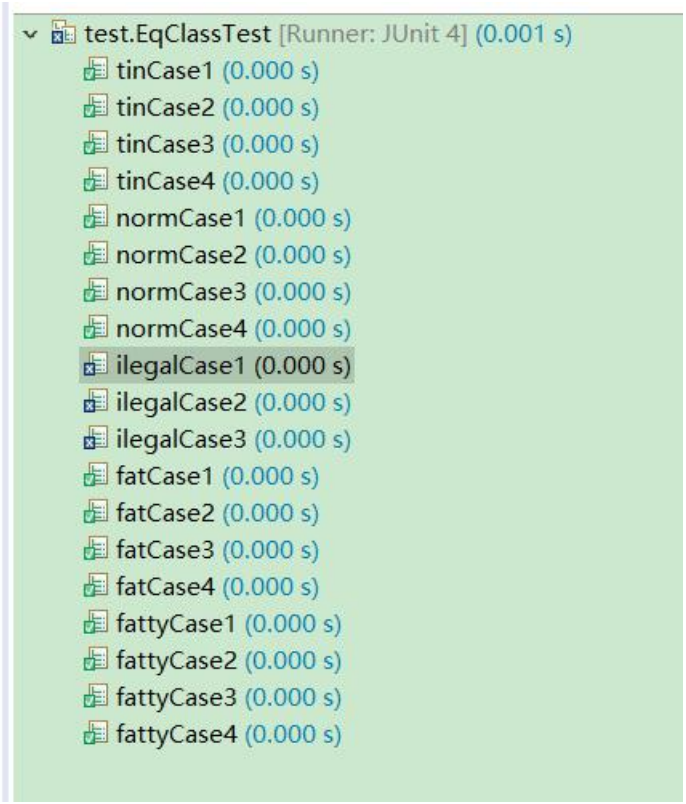
第五个等价类的测试用例：

```
//三个非法的测试用例
@Category(IlegalCase.class)
@Test
public void ilegalCase1() {
    bmi.setParams(-60,1.7);
    assertTrue("无效".equals(bmi.getBMIType()));
}

@Category(IlegalCase.class)
@Test
public void ilegalCase2() {
    bmi.setParams(60,-1.7);
    assertTrue("无效".equals(bmi.getBMIType()));
}

@Category(IlegalCase.class)
@Test
public void ilegalCase3() {
    bmi.setParams(-60,-1.7);
    assertTrue("无效".equals(bmi.getBMIType()));
}
```

测试结果：



如图，有三个测试用例没有通过测试，这三个测试用例都是属于无效等价类的。

```
//三个非法的测试用例
@Category(IlegalCase.class)
@Test
public void ilegalCase1() {
    bmi.setParams(-60,1.7);
    assertTrue("无效".equals(bmi.getBMIType()));
}

@Category(IlegalCase.class)
@Test
public void ilegalCase2() {
    bmi.setParams(60,-1.7);
    assertTrue("无效".equals(bmi.getBMIType()));
}

@Category(IlegalCase.class)
@Test
public void ilegalCase3() {
    bmi.setParams(-60,-1.7);
    assertTrue("无效".equals(bmi.getBMIType()));
}
```

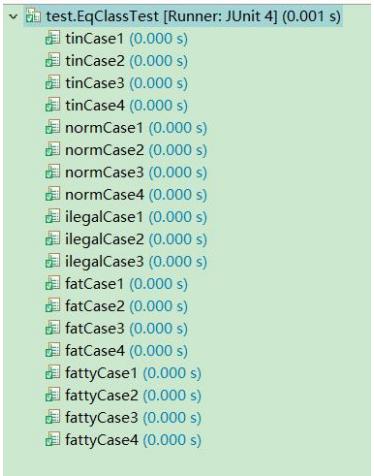
分析：应该时测试对象中没有处理非法数据。
查看测试对象的代码发现测试对象中有对非法数据的测试：

```
// 根据BMI值判断健康状况
public String getBMIType(){
    double bmi = 0.0;
    String result = "";
    if( weight>0 && height>0){
        //1.计算bmi
        bmi = weight/(height*height);
        //2.根据bmi判断所属健康分类
        if(bmi<18.5){
            result = "偏瘦";
        }else if(bmi<24){
            result = "正常";
        }else if(bmi<28){
            result = "偏胖";
        }else{
            result = "肥胖";
        }
    }else{
        return "Weight or height error!";
    }
    return result;
}
```

只是它的返回值和设计的测试用例总的值不一样，改一下即可：

```
// 根据BMI值判断健康状况
public String getBMIType(){
    double bmi = 0.0;
    String result = "";
    if( weight>0 && height>0){
        //1.计算bmi
        bmi = weight/(height*height);
        //2.根据bmi判断所属健康分类
        if(bmi<18.5){
            result = "偏瘦";
        }else if(bmi<24){
            result = "正常";
        }else if(bmi<28){
            result = "偏胖";
        }else{
            result = "肥胖";
        }
    }else{
        return "无效";
    }
    return result;
}
```

改完之后再次测试，测试用例全都通过：



2.边界值测试（源码在 BdText.java）

根据前面便边界值分析中的到的测试用例编写测试代码，如下：

```
@RunWith(Parameterized.class)
public class BdText {
    BMI bmi;//测试对象

    //测试数据
    double w;
    double h;
    String expect;

    //构造方法
    public BdText(double w,double h,String expect)

    //准备测试数据集合
    public static  List datas() {


    //测试开始之前，创建测试对象
    @Before
    public void before() {
        bmi=new BMI(w,h);
    }

    //测试结束，销毁测试对象
    @After
    public void after() {
        bmi=null;
    }

    //进行测试
    @Test
    public void test() {
        assertTrue(expect.equals(bmi.getBMIType()));
    }


}
```

其中 datas()方法用于提供测试用例，代码如下：

```
//准备测试数据集合
@Parameters(name = "身高：{0} 体重：{1} 期待的结果：{2}")
public static  List datas() {
    return (List) Arrays.asList(
            new Object[][]{

                    {2.0, 0.5 ,"偏瘦"},
                    {2.0 ,3.4, "偏瘦"},
                    {500.0, 0.5, "肥胖"},
                    {500.0 ,3.4, "肥胖"},
                    {1.5, 0.5, "偏瘦"},
                    {1.5, 3.4 ,"偏瘦"},
                    {2.5, 0.5, "偏瘦"},
                    {2.5, 3.4, "偏瘦"},
                    {1.0, 0.5, "偏瘦"},
                    {1.0, 3.4, "偏瘦"},
                    {3.0, 0.5, "偏瘦"},
                    {3.0 ,3.4, "偏瘦"},
                    {499.5, 0.5,"肥胖"},
                    {499.5, 3.4, "肥胖"},
                    {500.5, 0.5, "肥胖"},
                    {500.5, 3.4, "肥胖"},
                    {499.0, 0.5, "肥胖"},
                    {499.0, 3.4, "肥胖"},
                    {501.0, 0.5, "肥胖"},
                    {501.0 ,3.4, "肥胖"}
            }
    );
}
```

测试运行结果如下：


Finished after 0.023 seconds
Runs: 20/20
Errors: 0
Failures: 0
test.BdText [Runner: JUnit 4] (0.001 s)
[身高：2 体重：0.5 期待的结果：偏瘦] (0.001 s)
[身高：2 体重：3.4 期待的结果：偏瘦] (0.000 s)
[身高：500 体重：0.5 期待的结果：肥胖] (0.000 s)
[身高：500 体重：3.4 期待的结果：肥胖] (0.000 s)
[身高：1.5 体重：0.5 期待的结果：偏瘦] (0.000 s)
[身高：1.5 体重：3.4 期待的结果：偏瘦] (0.000 s)
[身高：2.5 体重：0.5 期待的结果：偏瘦] (0.000 s)
[身高：2.5 体重：3.4 期待的结果：偏瘦] (0.000 s)
[身高：1 体重：0.5 期待的结果：偏瘦] (0.000 s)
[身高：1 体重：3.4 期待的结果：偏瘦] (0.000 s)
[身高：3 体重：0.5 期待的结果：偏瘦] (0.000 s)
[身高：3 体重：3.4 期待的结果：偏瘦] (0.000 s)
[身高：499.5 体重：0.5 期待的结果：肥胖] (0.000 s)
[身高：499.5 体重：3.4 期待的结果：肥胖] (0.000 s)
[身高：500.5 体重：0.5 期待的结果：肥胖] (0.000 s)
[身高：500.5 体重：3.4 期待的结果：肥胖] (0.000 s)
[身高：499 体重：0.5 期待的结果：肥胖] (0.000 s)
[身高：499 体重：3.4 期待的结果：肥胖] (0.000 s)
[身高：501 体重：0.5 期待的结果：肥胖] (0.000 s)
[身高：501 体重：3.4 期待的结果：肥胖] (0.000 s)