**软件质量保证与测试**

**2020春季**

教师：杨秋辉

**实验报告5 –测试覆盖率统计**

姓名：龙行超 学号：2017141463145

1. **引言（2分）**

测试覆盖率：用来度量测试完整性的一种手段，分为三种代码覆盖率、需求覆盖率和缺陷覆盖率。掌握测试覆盖率数据，有利于客观认识软件质量，正确了解测试状态，有效改进测试工作。

代码覆盖率：面向软件开发和实现的定义，它关注执行测试用例时，有哪些代码被执行到了，有哪些没被执行到，也即，被执行代码总数与代码总数量之间的比值。

需求覆盖率：用于黑盒测试，基于软件需求而并非软件开发，是测试所覆盖需求数目与总需求个数的比值。

缺陷覆盖率：面向测试结果的覆盖率，测试实际发现缺陷的个数与测试应该发现的缺陷的个数的比值。

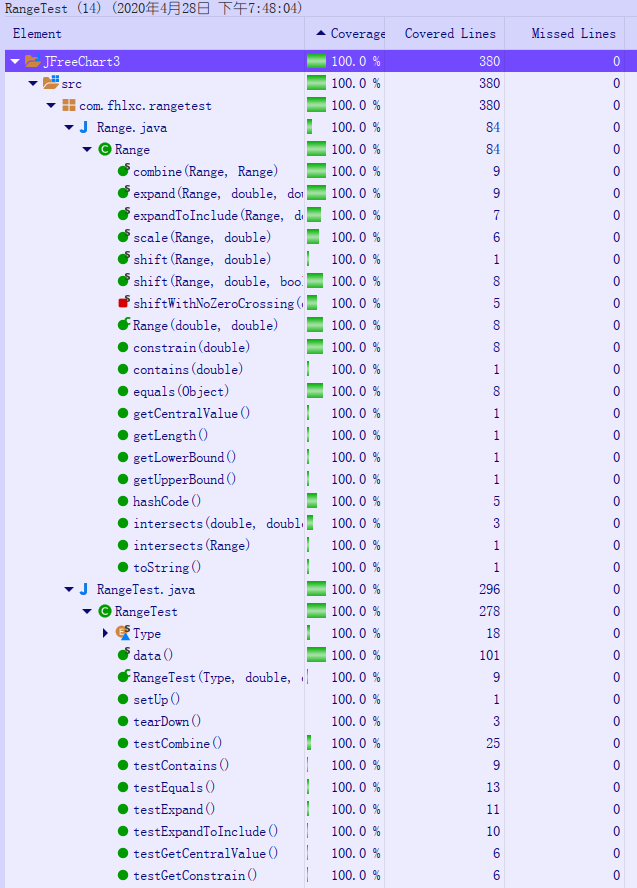
以上三种覆盖率并不是相互排斥的，他们适用于各种不同的场景，是相互补充的。

EclEmma开源的软件测试工具，对java代码进行覆盖率的统计与分析，可以将多次运行的结果组合在一起，查看覆盖情况。本次实验就是利用这个工具，在eclipse上，在变异测试基础上，进行测试用例的再设计，完善测试。

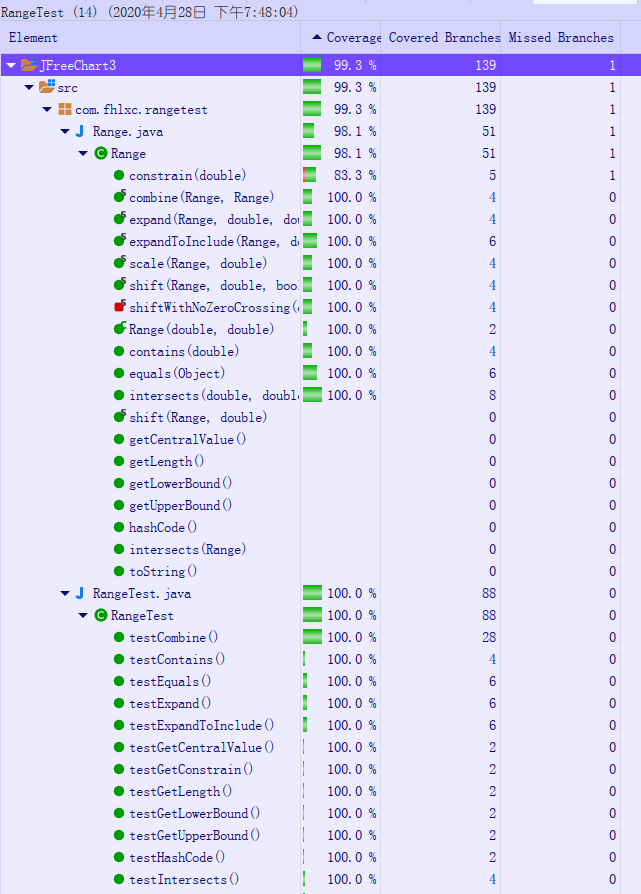
1. **测试策略（2分）**

采用动态白盒测试的方法，利用eclipse的插件Eclmma对JFreechart中的Range类进行单元测试，测试目标是覆盖率达到100%，对未达到100%的测试用例进行分析与说明，其中主要涉及的测试用例设计方法包括：活动流图、等价类、边界值。测试完成之后，需要编写测试文档，并导出覆盖测试的html文件，供后续使用。

1. **覆盖率与变异分数的关系分析（10分）**
   1. 覆盖率情况
2. 行覆盖情况



1. 分支覆盖情况



* 1. 变异分数为100%，但覆盖率未达到100%的方法及测试用例分析

没有这样的方法.

* 1. 覆盖率为100%，但变异分数未达到100%的方法及测试用例分析

没有这样的方法.

* 1. 测试用例完整，但该方法的变异分数和覆盖率未达到100%原因分析

方法内容如下(蓝色是变异操作地方,黄色是未覆盖的分支的地方):

**public** **double** constrain(**double** value) {

**double** result = value;

**if** (!contains(value)) {

//变为>=,不改变语义

**if** (value > **this**.upper) {

result = **this**.upper;

}

//变为<=,不改变语义

//分支无法覆盖，因为大于等于lower，小于等于upper的数在范围内，就无法进入这个分支

**else** **if** (value < **this**.lower) {

result = **this**.lower;

}

}

**return** result;

}

覆盖率未达到百分百的原因如下:

首先该行语句分为两个分支,分别是value < this.lower和value >= this.lower两个.第一个分支是覆盖了的,我接下来分析一下第二个分支. 要想进入第二个分支,需要满足三个条件:

1. range不包括该value
2. value要不小于lower
3. value不能大于upper

这三个条件中,第一个条件和后面两个条件冲突,所以覆盖率没办法达到百分之百.

变异分数未达到100%原因如下:

当改变为小于等于或者大于等于时,它就在range范围内了,那么得到的结果和没变异之前是一样的,因为根本没有执行变异的代码,因此变异前后是等价的,所以变异分数不能达到100%.

1. **两种测试用例质量评估方法分析（4分）**

覆盖率从分支、行、函数等角度诠释了测试用例对程序中的代码的检测情况。覆盖率高是认真、仔细做单元测试的必然结果，但是有些问题，即使覆盖率达到了100%，也是发现不了的。

变异测试做为对测试用例设计的补充，通过改变代码的行为，使程序得到不同的结果，帮助测试人员设计测试用例，发现问题。

总结，覆盖率是测试是否充分的一个指标，变异测试是另一种角度设计测试用例去发现程序的缺陷。

1. **针对本次实验，你遇到的问题和挑战，你的收获以及你有什么反馈（2分）**

通过这次的实验，我学会了使用覆盖测试的工具——EclEmma，对覆盖率有了一个更深入的理解，此外，还充分利用了数据流图、边界值和等价类的方法去设计有针对性的测试用例使得多项覆盖率指标尽可能的达到100%。