实验 4　变异测试

四川大学软件学院 杨秋辉 yangqiuhui@scu.edu.cn

1 前言

本实验中，学生使用变异测试工具对被测系统进行变异测试，以评估自己所写的测试用例质量。本实验用 1 次课（2 学时）完成。

具体任务是：根据提供的被测系统代码及其 Javadoc 描述，编写 junit 测试用例，并使用变异工具对被测系统进行变异测试，同时完成实验报告编写。

1.1 实验目的

熟悉变异测试的概念，熟悉变异测试工具 PiTest（简称 PIT）的使用流程。

1.2 变异工具简介

本次实验使用的工具是 PiTest，该工具的 eclipse 插件易于安装使用，运行效率较高。还有其他一些支持 Java 语言的变异工具，如：mujava 、muclipse、jumble、jester 等，其功能和使用方式各有不同。

PiTest 提供 29 种不同类型的变异算子，可以对代码进行多种类型的变异，包括：CONDITIONALS_BOUNDARY、INCREMENTS、MATH 等。工具在使用时，不需要用户指定使用哪些变异算子、进行哪些变异，工具会根据被测代码的具体情况，自动选择可以使用的变异算子，完成对被测代码的变异。详细介绍参见其官网，网址为：https://github.com/hcoles/pitest，http://pitest.org/。

变异测试工具的基本使用流程包括：测试用例编写（自动或手工完成）、使用工具进行变异，在变异体上执行测试用例，查看变异测试结果。

1.3 变异测试工具安装

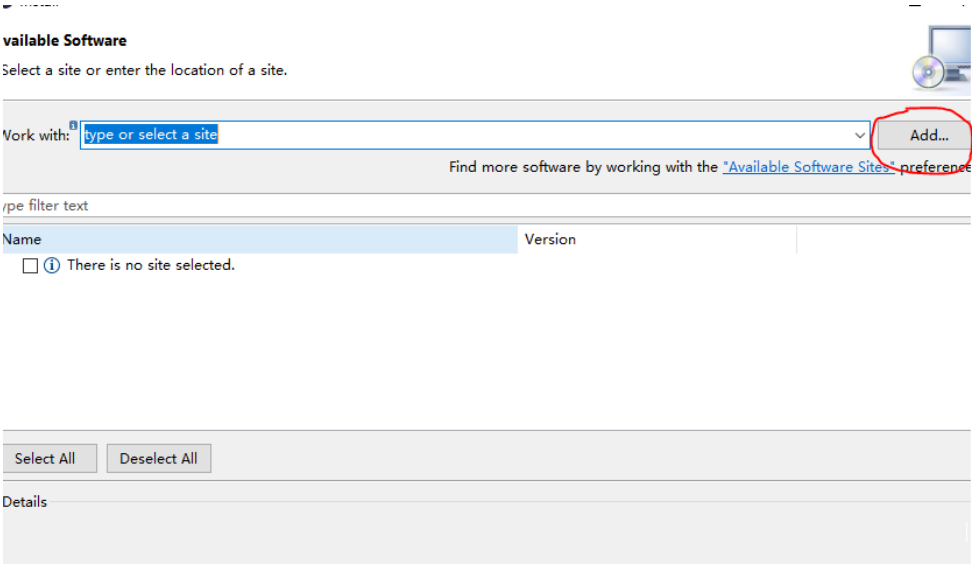
1.3.1 系统需求

PiTest 需要的 jdk 环境 Java5 或以上，而且类路径上有 JUnit 或 TestNG。支持 JUnit4.6 或以上的版本（可在 eclipse 的安装目录下的 plugins 文件下查看 JUnit 版本）。

此外，PiTest 需要支持 XStream 的增强模式的 JVM。

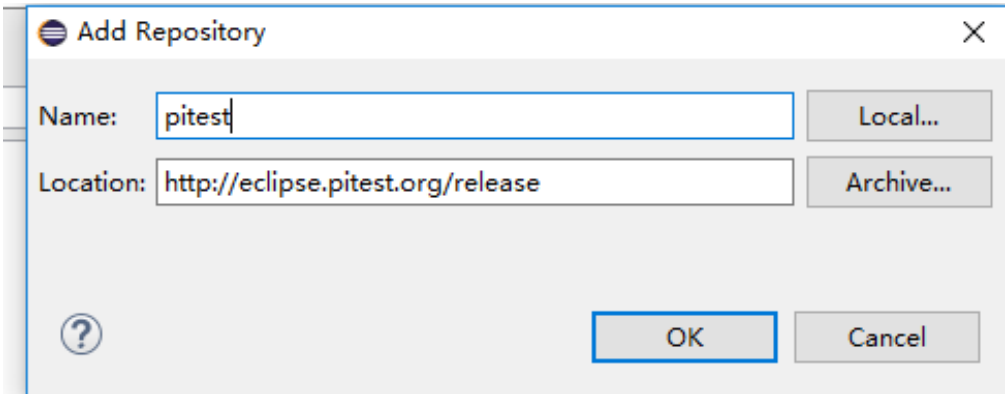
1.3.2 安装 PITest 插件到 eclipse 中

（1）打开 eclipse，点击 help-->Install new software...，出现如下界面，

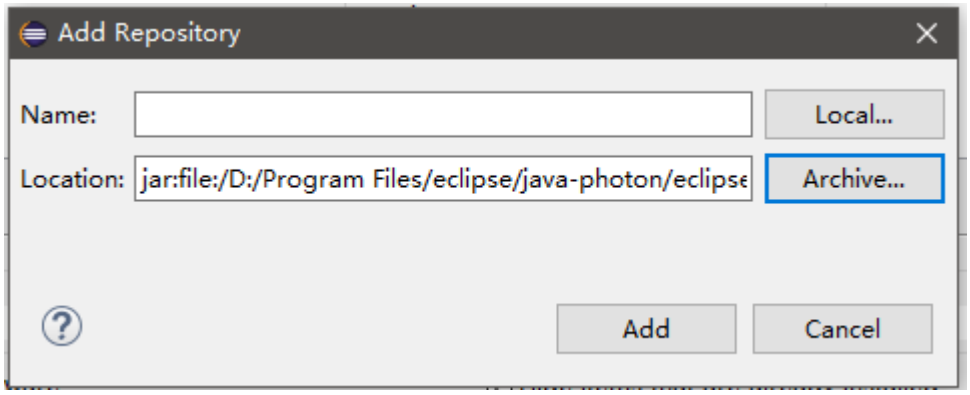


（2）以下方式任选其一：

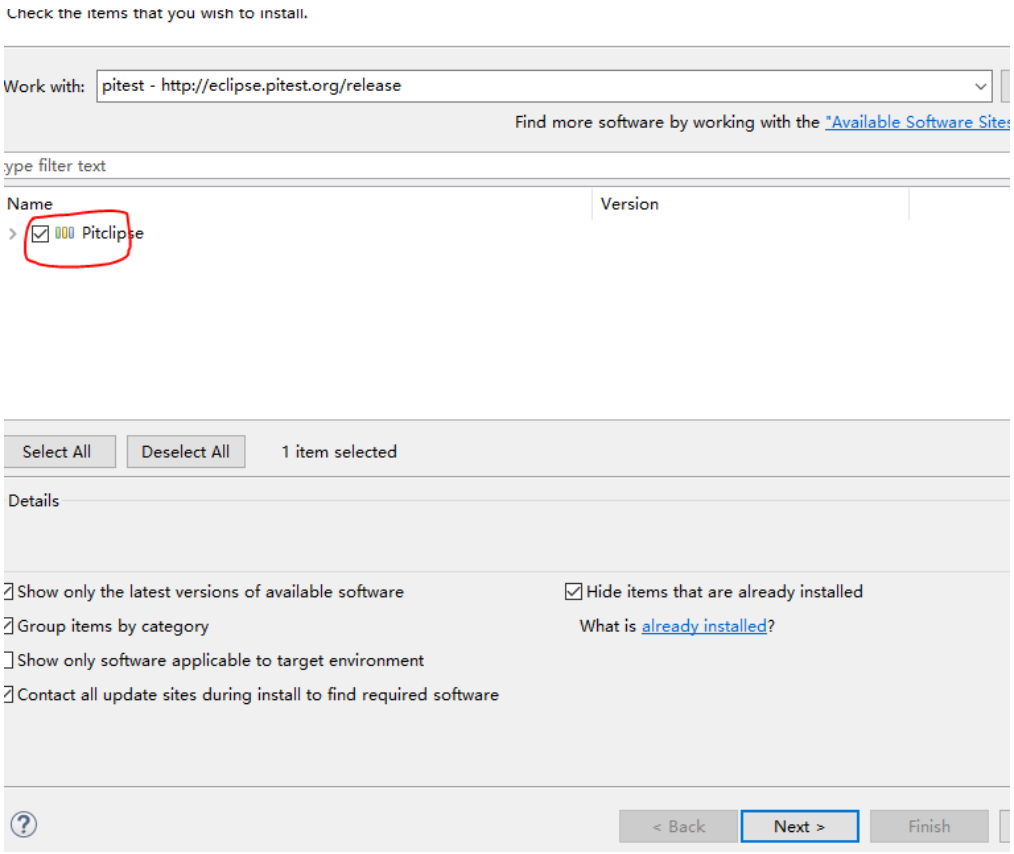
1. 从 URL 安装：点击 add,在出现的会话框中填写如下网址，即 PITest 在 eclipse 中插件的原址：http://eclipse.pitest.org/release/



2. 从本地安装：点击 add，在出现的对话框中点击 Archive...，选择下载的 org.pitest.pitclipse.p2-2.1.0.zip



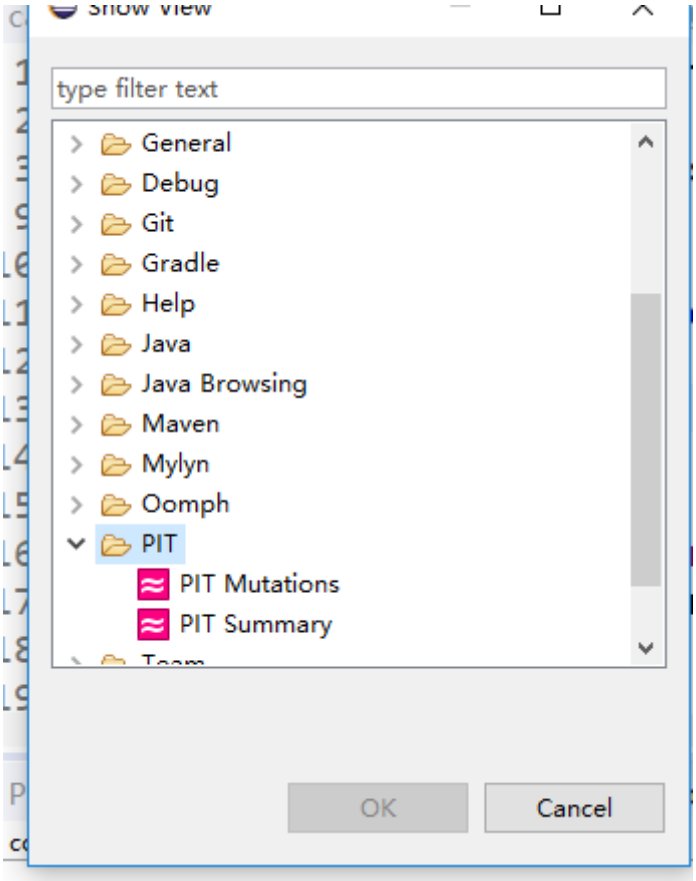
（3）加载出 Pitclipse，勾上，点击 next，勾选“同意许可协议”，最后 finish，完成插件的安装。



（4）选择重启，查看该插件是否安装成功。

a.随便选择一个项目或类右键，选择 run as，出现 PITest Mutation Test 即安装成功。

b.选择菜单栏中的 window--Show view--Others，查找是否有 PIT 的文件夹，并把 PIT Mutation 和 PIT Summary 双击调到控制台，PIT 有自己的 GUI 界面。



1.4 被测程序

JFreeChart 项目里的 Range 类（Range.java 源文件）

2 实验流程

1、新建一个项目，取名为：JFreeChart；新建一个包，包名为：rangetest；将 Range.java 文件拷贝到该包下，注意在 Range.java 文件中修改引入的包名。

2、右键 Range 类文件，选择:【new】---【Junit Test Case】,新建的测试类，类名为 RangeTest，superclass 框里的内容空着，需要创建的桩函数（Which method studs would you like to create?）栏中选中 setUp()、tearDown()、constructor()。

3、创建完成后，将上周生成的 RangeTest.java 里的内容拷贝到这周的 RangeTest.java 中，注意删除里面引入的“import org.jfree.data.Range;”这句话。

另外再选 Range 类里面的 3 个方法，编写相应的 Junit 测试用例。

Range.java | RangeTest.java

```
    @Before
    public void setUp() throws Exception {
        testRange=new Range(-1,1);
    }

    @After
    public void tearDown() throws Exception {
        testRange=null;
    }

    @Test
    public void testContains() {
        assertTrue("this value is not within -1 and 1",testRange.contains(0));
    }
    @Test
    public void testContains1() {
        assertFalse("this value is  within -1 and 1",testRange.contains(2));
    }
    @Test
    public void testCentralValue() {
        assertEquals(0,testRange.getCentralValue(),0.0000001d);
    }

}
```

8 个方法的测试用例都放在 RangeTest. java 中

4、运行你的 RangeTest. java（【Run As】---【Junit Test】），查看测试结果，如果某个测试用例执行结果为 error，说明测试用例本身有问题，则修改或删除该测试用例。

5、如果某个测试用例执行结果为 failure，说明被测代码有缺陷，需要根据 Javadoc 中关于此方法的说明，对被测代码进行修改，直到你的测试用例通过。

6、所有测试用例通过后，在 RangeTest. java 上运行变异测试：【Run As】---【PITest Mutation Test】，出现如下图结果：

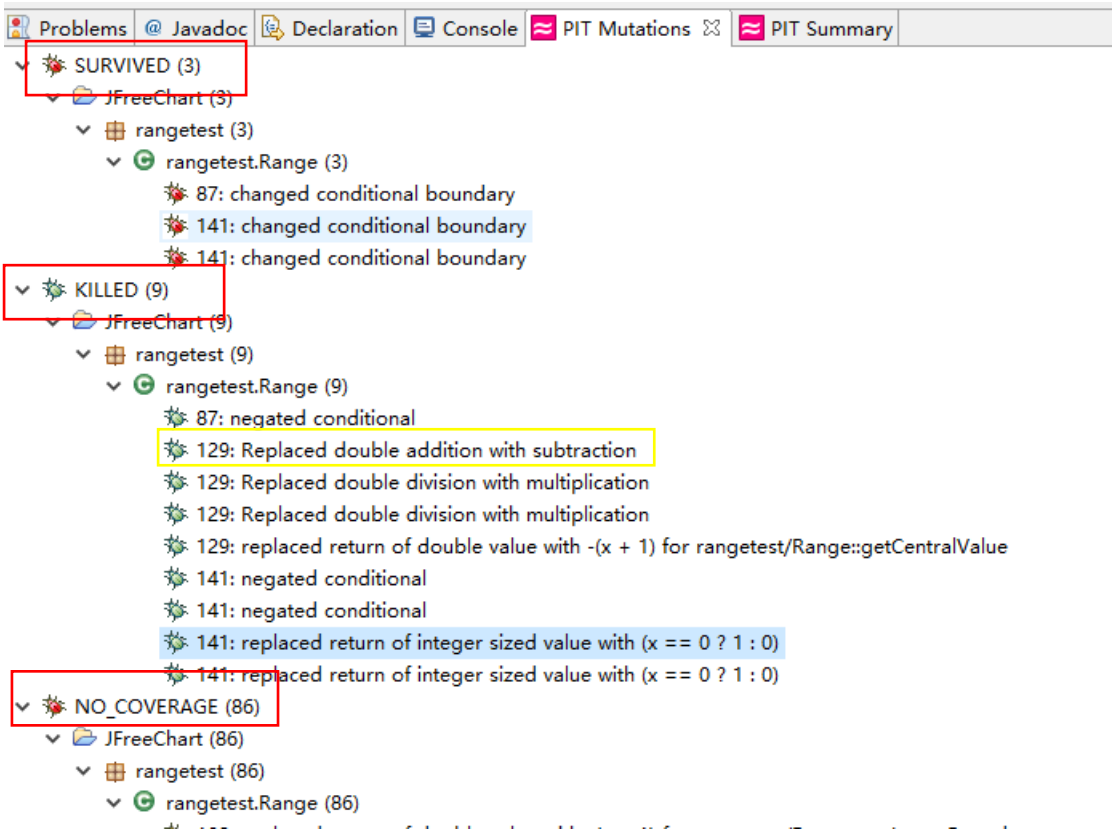
```
        testRange=new Range(-1,1);
    }

    @After
    public void tearDown() throws Exception {
```

Problems | Javadoc | Declaration | Console | PIT Mutations | PIT Summary

```
<terminated> RangeTest (1) [PIT Mutation Test] C:\Program Files\Java\jdk1.8.0_151\bin\javaw.exe (2019年5月15日 下午3:18:4
> KILLED 3 SURVIVED 0 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 21
--------------------------------------------------------------------------------
> org.pitest.mutationtest.engine.gregor.mutators.NegateConditionalsMutator
>> Generated 26 Killed 3 (12%)
> KILLED 3 SURVIVED 0 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 23
--------------------------------------------------------------------------------
Sending results: PitResults [htmlResultFile=D:\variation\.metadata\.plugins\org.pitest.pit
Closing server
Closed
```

运行结果中出现“Closing server” “Closed”，且没有其他异常信息，即成功运行了 PITest。

6、查看变异体情况。进入 PIT Mutation 界面查看，这个报告中显示的是关于变异体的信息，分为三部分，第一部分是存活的（Survived）变异体，第二部分是被杀死的（Killed）变异体，，第三部分是未被覆盖的（NO_COVERAGE）变异体。

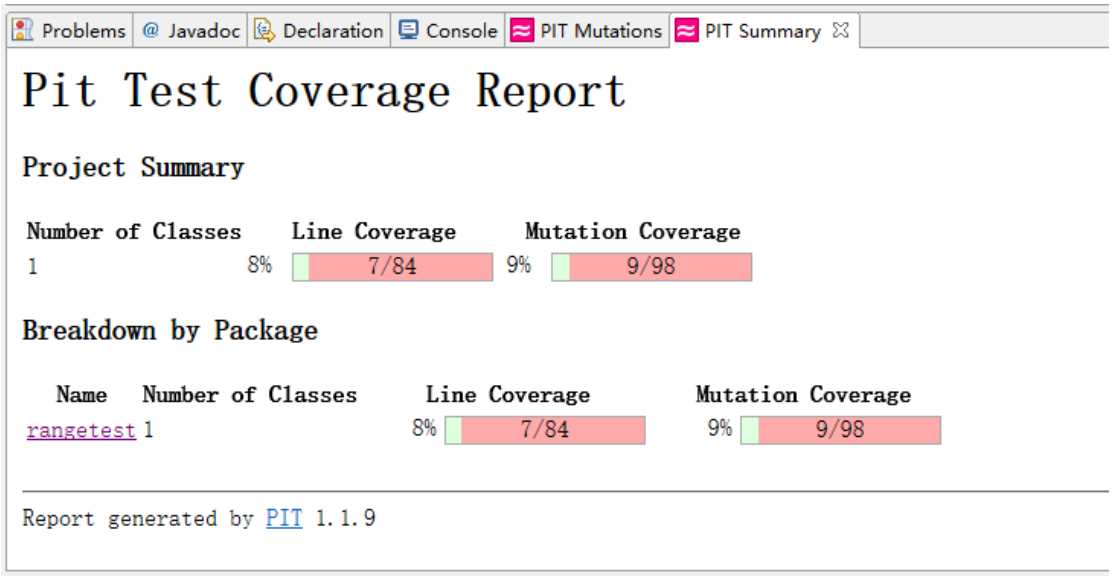


这个界面中还可以查看针对具体代码行的变异操作，如上图中被杀死的变异体中，在源程序中的 129 行中，使用减法代替了加法（图中黄色边框标注的内容）。

具体的变异操作可参考链接：

http://pitest.org/quickstart/mutators/#NEGATE_CONDITIONALS

7、查看变异测试结果报告。进入 PIT Summary 界面查看。这个报告中给出了代码覆盖情况和变异覆盖（变异分数）的信息，如图：

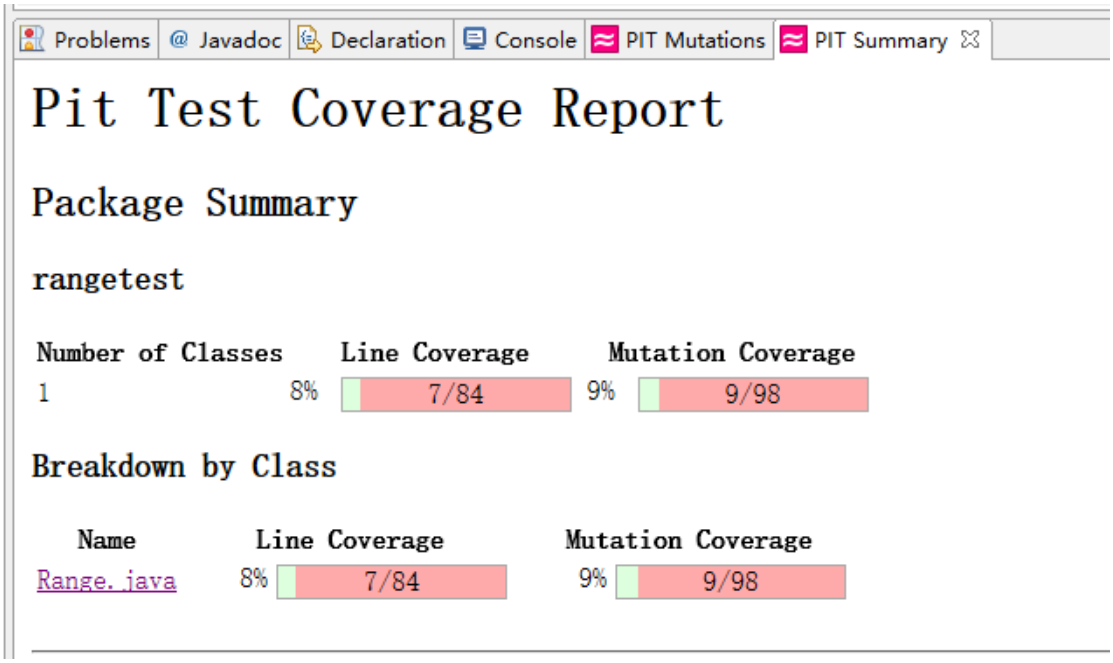


注：PiTest 中，变异覆盖（Mutation Coverage）的概念与变异分数概念一致，其值为：杀死的变异体数量/总变异体数量。

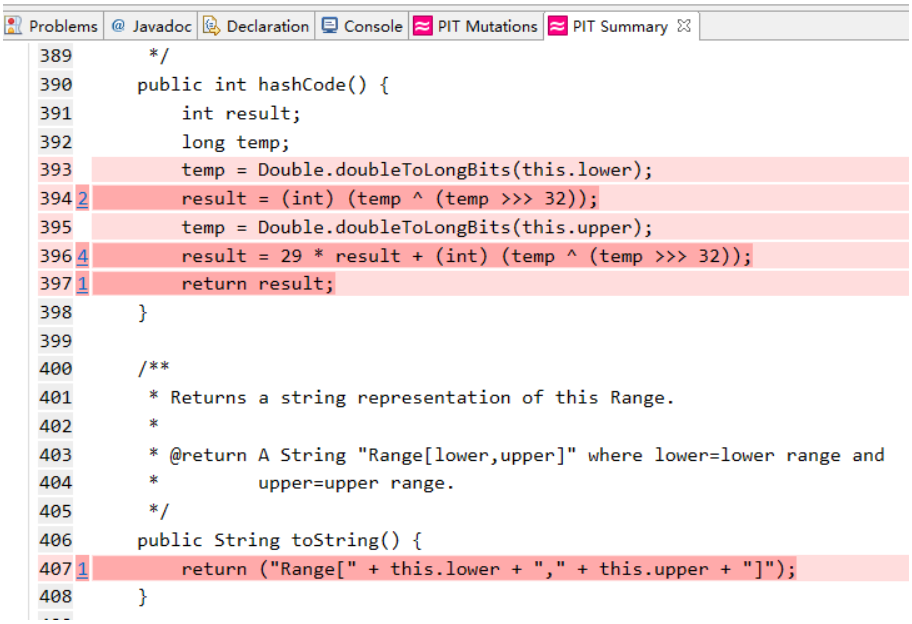
该图中的信息是项目的覆盖信息以及变异分数和项目中包的覆盖信息以及变异分数。项目的行覆盖是 8%，变异覆盖是 9%，rangetest 这个包的行覆盖是 8%，变异覆盖是 9%，由于该项目

中只有一个包，他们的行覆盖和变异覆盖是相同的。

点击 rangetest 这个包，会出现该包下所有类的行覆盖分数和变异覆盖分数。

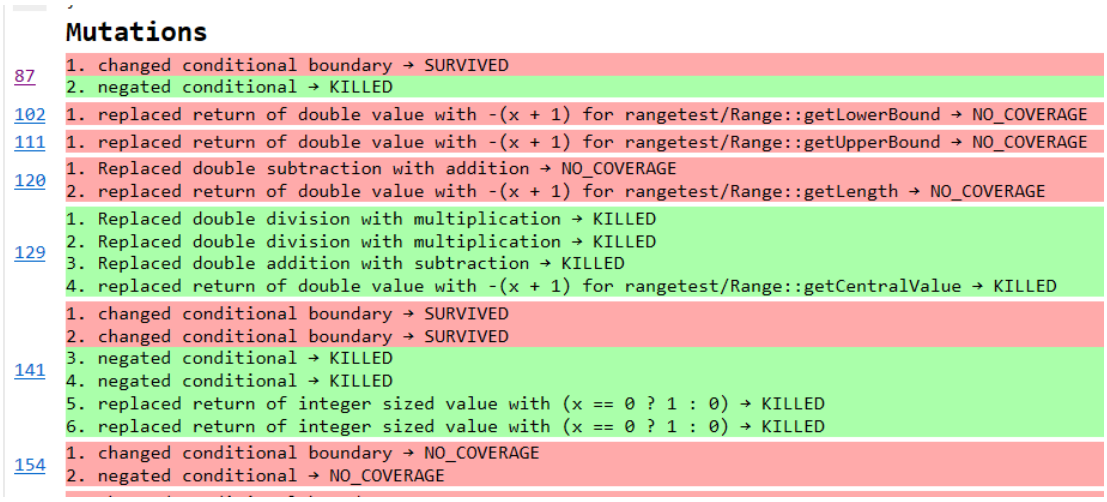


点击 Range.java 这个类，出现详细的报告信息，包括四个部分：源代码的测试覆盖情况，变异操作，激活的变异算子（即：在本次测试中使用的变异算子），以及测试的执行时间。

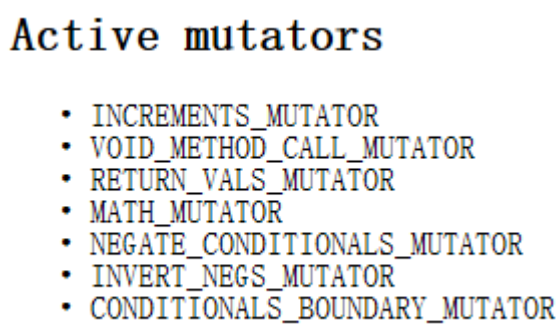


源代码的测试覆盖情况

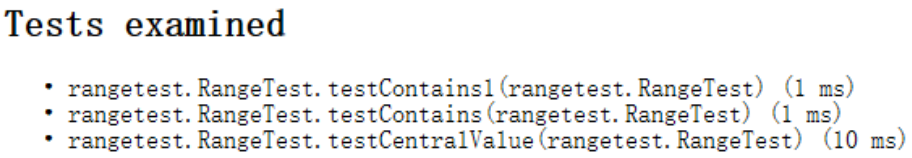
注：本次实验课暂不关注代码覆盖情况。



变异情况



激活的变异算子



测试用例的执行时间

8、**学生任务：**针对自己所选择的 Range 类中的 8 个方法，使用已经编写的测试用例，执行变异测试，查看测试结果。根据测试结果补充必要的测试用例，尽量达到较高的变异覆盖，并将变异测试结果记录到报告中。

要求：尽量达到较高的变异覆盖分数，变异覆盖分数会直接影响本次实验成绩。对于写了完整测试用例的方法，依然存在存活的变体和没有覆盖到的变体，需要分析原因。

3 总结

完成本次实验后，学生对变异测试应该有一定了解，对于学生自行检验 junit 测试用例的质量也会有很大的帮助。

在本次实验中使用的工具是 Pitclipse（PIT 在 eclipse 中的插件），实际上 PIT 是一个强

大的变异测试系统，其中还包括很多扩展应用，使其应用的也更广。

虽然 Pitclipse 是一种很基础的变异测试工具，只用于了当前的实验环境。但是通过本实验，我们可以基本了解变异测试的基本思路，并且还可以在它的基础上进行更加深入的学习。

4 提交的内容

请根据实验资料中给出的“实验 4-变异测试-实验报告模板.doc”撰写实验报告。要求提交的内容包括：实验报告 doc 或 pdf 文件+此次实验的整个项目。将所有内容打包提交，打包文件命名规定：“学号-姓名-变异测试实验报告.zip/rar”。