

实验 5 测试覆盖率统计

四川大学软件学院 杨秋辉 yangqiuhui@scu.edu.cn

1 前言

本实验主要完成对测试覆盖率统计工具 EclEmma 的初步熟悉和使用。

1.1 测试工具简介

在本实验中使用的主要测试工具是 EclEmma。EclEmma 作为 Eclipse 的插件，通过汇总和突出显示覆盖率结果，可以有效地帮助开发人员考察测试覆盖率。EclEmma 在覆盖测试领域十分优秀，以致于它在 2006 年成为了 Eclipse Community Awards Winners 决赛选手。

1.2 被测系统

该实验的被测系统为 JFreeChart。使用实验指导资料中的 Range.java 类。

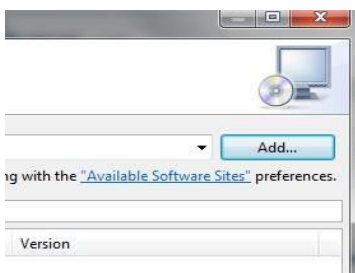
本实验指导中的练习阶段使用的被测系统是 ATM 模拟系统，以及一个简单的 helloworld 程序。

2 熟悉工具

本节目的是熟悉 EclEmma 的简单使用，下面将详细说明如何进行工具的安装和使用。

2.1 ECLEmma 的安装

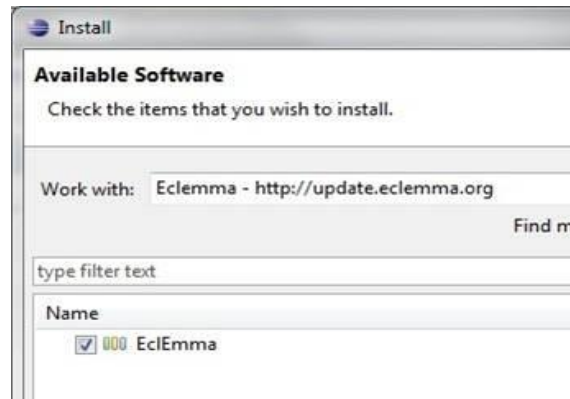
1. 启动 eclipse---点击 Help 菜单---Install New Software，在弹出的对话框中，点击 Add



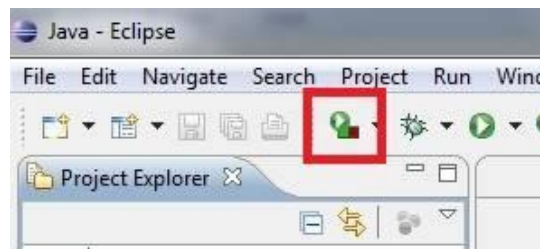
2. 输入 Name，例如 EclEmma。输入 Location: <http://update.eclemma.org/>



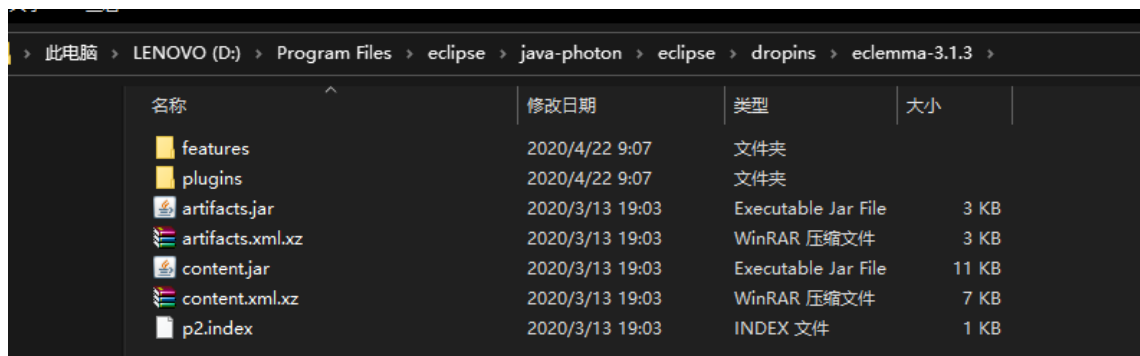
3. 在 Work With 处选择刚刚输入的 Location 地址，并选中 Name 中的 EclEmma，如下图。点击 Next，完成后续安装步骤。



4. 安装后重新启动 eclipse，如果成功安装，工具栏上会出现一个新的按钮



注：另外一种安装方法：使用资料中的 eclEmma-3.1.3.zip 文件，解压，然后将解压后的文件拷贝到 Eclipse 目录下的 dropins 中，文件目录结构如下：



2.2 EcLEMMa 的使用

2.2.1 黑盒测试后，查看.jar 的覆盖率

手动运行被测程序，查看被测系统的覆盖率情况。

1. 打开 eclipse（3.3 或以上版本）。

2. 打开 New Project 对话框，选择 File -> New -> Java Project


3. 项目名为 ATM，点击 Next。

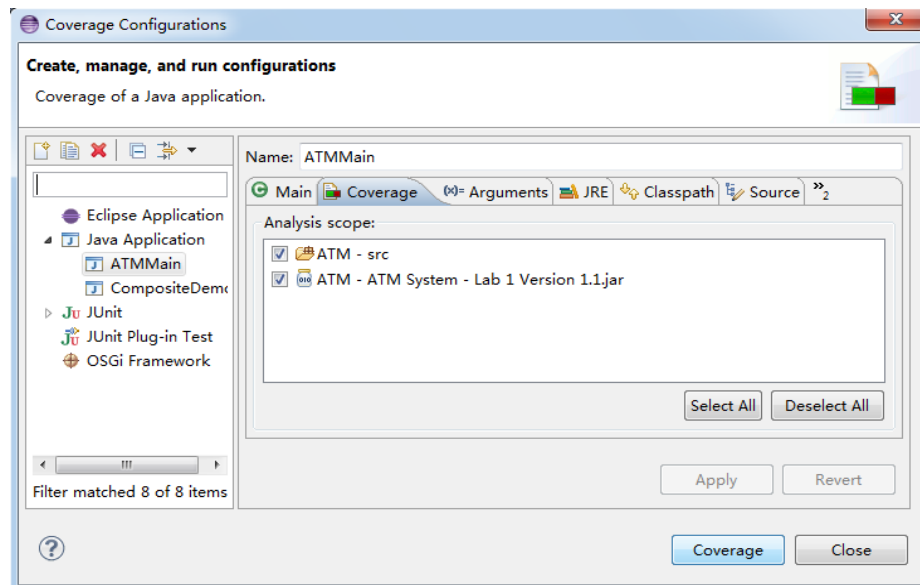
4. Java Settings 对话框将会显示。对话框顶部有 4 个选项卡：Source、Projects、Libraries 和 Order and Export。选择 Libraries 选项卡，然后点击 Add External JARs...按钮。

5. 选择添加实验指导资料中的 ATM System - Lab 1 Version 1.1.jar 文件。

6. 点击 finish。这时被测试系统的项目就建立了。

7. 展开 Referenced Libraries，右键 ATM System - Lab 1 Version 1.1.jar 文件，选择【Run As】---【Java Application】，此时会出现系统的界面，点击关闭。

8. 设置覆盖率统计范围。缺省情况下，EcLEmma 只是统计源代码的覆盖率情况，可以通过设置，使其统计在手动执行*.jar 时的覆盖情况。点击  中的下拉框，选择 Coverage Configurations...，在出现的窗口中，选择 Coverage 项，将 Analysis scope 中的 ATM - Atm System - Lab 1 Version 1.1.jar 选中，如下图：




9. 进行覆盖率统计。点击 Coverage 按钮，被测系统开始执行。随意执行你想要做的任何操作，EcLEmma 会记录哪些类或方法被执行了，当你退出系统时，在 Eclipse 窗口中会显示覆盖率情况，如下图：

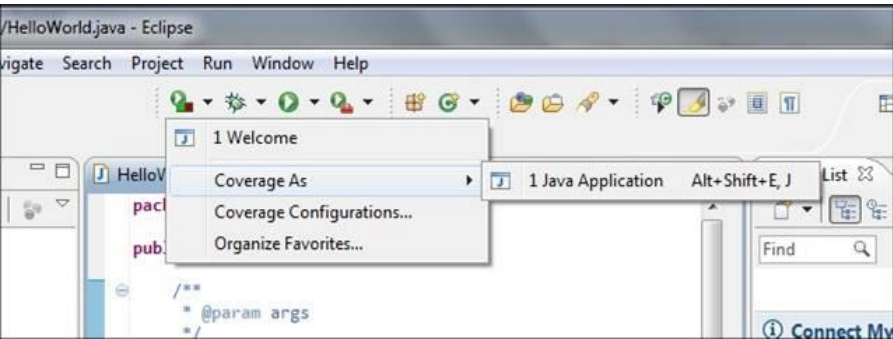
ATMMain (2019-5-21 19:48:05)				
Element	Coverage	Covered Instructi...	Missed Instructi...	Total Instructions
ATM	<div><div></div></div> 73.4 %	3,788	1,372	5,160
ATM System - Lab 1 Version 1.1.jar	<div><div></div></div> 73.4 %	3,788	1,372	5,160
atm.transaction	<div><div></div></div> 33.5 %	277	550	827
simulation	<div><div></div></div> 82.2 %	2,432	526	2,958
atm	<div><div></div></div> 72.9 %	288	107	395
banking	<div><div></div></div> 83.3 %	479	96	575
atm.physical	<div><div></div></div> 81.9 %	230	51	281
(default package)	<div><div></div></div> 66.1 %	82	42	124

2.2.2 黑盒测试后，查看源程序代码的覆盖率

1.在 Eclipse 中新建一个项目，再建立一个 HelloWorld 类，其代码如下：

```
public class HelloWorld {
    public static void main(String[] args) {
        int rand = (int) (Math.random() * 100);
        if (rand % 2 == 0) {
            System.out.println("Hello, world! 0");
        }else{
            System.out.println("Hello, world! 1");
        }
        int result = rand % 2 == 0 ? rand + rand : rand * rand;
        System.out.println(result);
    }
}
```

2. 保存代码后，通过 EclEmma 运行 HelloWorld。点击，选择下拉列表中的 Coverage As>1 Java Application，运行 HelloWorld 程序



3. 运行结果如下。可以看到，各代码行被标记为不同的颜色。绿色表示完整执行；红色表示没有执行；黄色表示部分执行。黄色的行通常出现在本行代码包含分支的情况。由于程序中的 rand 是随机数，因此你的运行结果可能与这里会有不同（两个 println 中有且只有一个红色的行）。

```
public class HelloWorld {  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        int rand = (int) (Math.random()*100);  
        if(rand%2==0){  
            System.out.println( "Hello, world! 0");  
        }  
        else  
            System.out.println("Hello, world! 1");  
  
        int result = rand%2==0? rand+rand:rand*rand;  
        System.out.println(result);  
    }  
}
```


4. 将鼠标移到黄色行前面的黄色菱形上，会显示此行代码为黄色的原因（下图中的意思是有一个分支没有执行）：

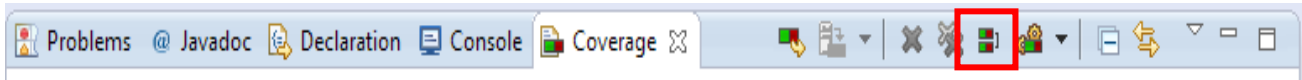
```
public class HelloWorld {  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        int rand = (int) (Math.random()*100);  
        1 of 2 branches missed.{  
        System.out.println( "Hello, world! 0");  
        }  
        else  
            System.out.println("Hello, world! 1");  
  
        int result = rand%2==0? rand+rand:rand*rand;  
        System.out.println(result);  
    }  
}
```

5. Eclipse 窗口下方的 Coverage 窗口中可以显示程序的测试覆盖率情况

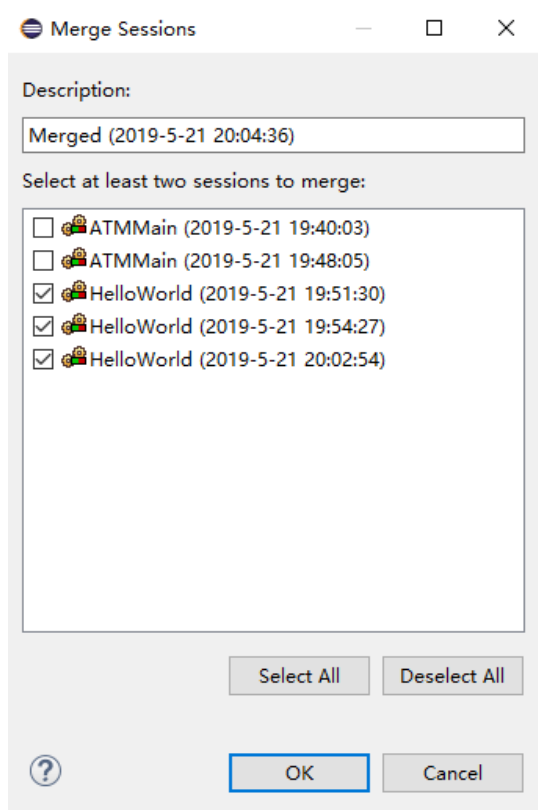
HelloWorld (2019-5-21 20:02:54)				
Element	Coverage	Covered Instructi...	Missed Instructi...	Total Instructions
▼ HelloWorld	<div><div></div></div> 74.3 %	26	9	35
▼ src	<div><div></div></div> 74.3 %	26	9	35
▼ helloworld	<div><div></div></div> 74.3 %	26	9	35
▼ HelloWorld.java	<div><div></div></div> 74.3 %	26	9	35
▼ HelloWorld	<div><div></div></div> 74.3 %	26	9	35
main(String[])	<div><div></div></div> 81.2 %	26	6	32

6. 覆盖率结果合并。

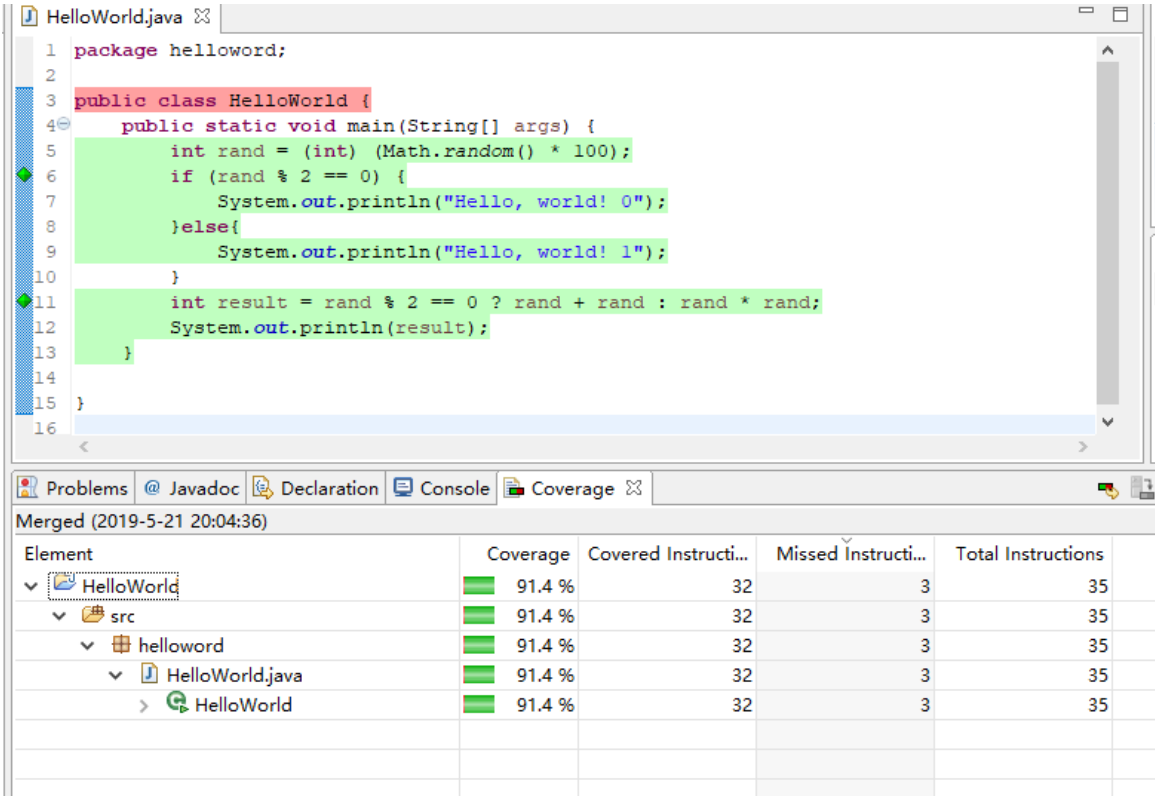
想在一次运行中覆盖所有的代码通常比较困难，如果能把多次测试的覆盖数据综合起来进行察看，那么我们就更方便的掌握多次测试的测试效果。EclEmma 提供了这样的功能。现在，让我们重复数次对 HelloWorld 的覆盖测试。我们注意到 Coverage 视图总是显示最新完成的一次覆盖测试。事实上，EclEmma 为我们保存了所有的测试结果。接下来，我们将通过 Coverage 视图的工具按钮来结合多次覆盖测试的结果。单击下图所示工具栏按钮中的。



一个如下图所示对话框将弹出，以供用户选择需要合并的覆盖测试。



在合并完成之后，我们可以观察到 Java 编辑器和 Coverage 视图都显示了合并之后的结果。



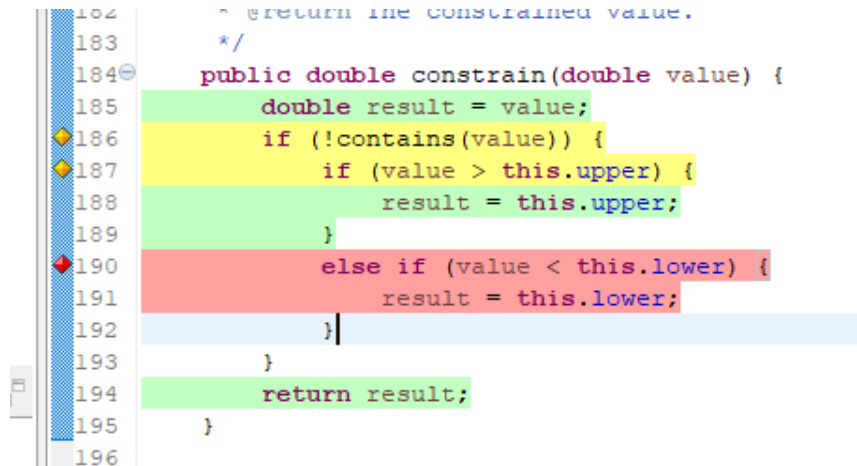
可以看到，通过多次运行覆盖测试，最终我们的代码达到了 91.4% 的测试覆盖率。有趣的是，图中的“public class HelloWorld {” 这一行代码被标记为红色，而此行代码实际上是不可执行的。原因在于，我们没有生成任何 HelloWorld 类的实例，因此缺省构造函数没有被调用，而 EclEmma 将这个特殊代码的覆盖状态标记在类声明的第一行。

2.2.3 白盒测试后，查看代码覆盖率

现在我们以上节课写的 Range 类的单元测试用例为例，来说明如何在单元测试后，查看被测代码覆盖率情况。

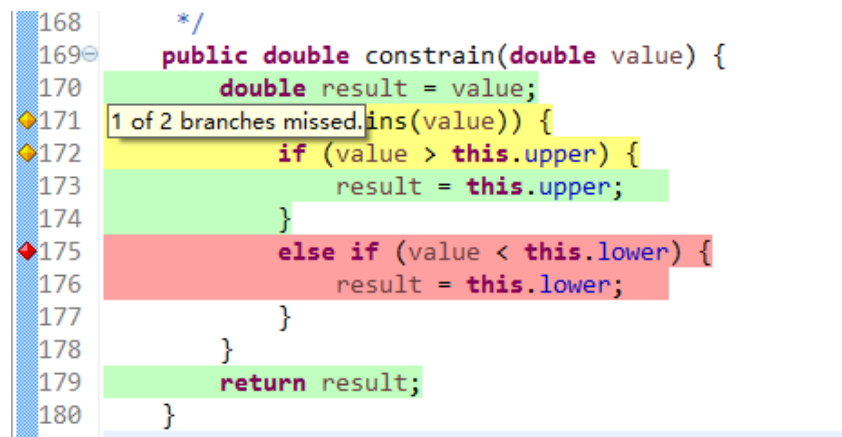
- 1. 打开上周的实验项目。
- 2. 在 RangeTest.class 上点击右键，Coverage As，然后 JUnit Test，查看测试用例和被测函数源代码，如下

```
@Test
public void testContains() {
    assertTrue("this value is not within -1 and 1",testRange.contains(0));
}
```



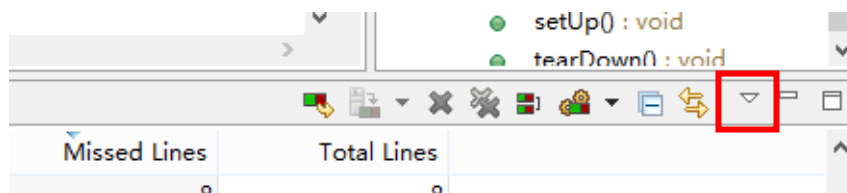
可以看到，源代码中各代码行被标记为不同的颜色。**绿色**表示完整执行；**红色**表示没有执行；**黄色**表示部分执行。黄色的行通常出现在本行代码包含分支的情况。

3. 将鼠标移到黄色行前面的黄色菱形上，会显示此行代码为黄色的原因（下图中的意思是有一个分支没有执行）：

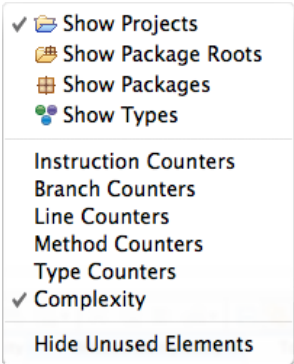


4. 在 Coverage 窗口可查看 Range 类下每个方法的分支覆盖情况、语句覆盖情况等(如果 Coverage 窗口没有打开，请从 windows, show view 里面打开)。

在 Coverage 窗口的右上角（如下图）



点击图中红框里的下三角符号，可以切换不同标准覆盖率情况，有指令（instruction）覆盖、行（lines）覆盖、分支（branch）覆盖、方法覆盖等，



切换到不同的视图，就可以查看相应的覆盖情况。

分支覆盖：

RangeTest (2019-5-21 20:42:30)					
Element	Coverage	Covered Branches	Missed Branches	Total Branches	
▼ Range	11.5 %	6	46	52	
intersects(double, double)	0.0 %	0	8	8	
expandToInclude(Range, double)	0.0 %	0	6	6	
equals(Object)	0.0 %	0	6	6	
combine(Range, Range)	0.0 %	0	4	4	
expand(Range, double, double)	0.0 %	0	4	4	
scale(Range, double)	0.0 %	0	4	4	
shift(Range, double, boolean)	0.0 %	0	4	4	
shiftWithNoZeroCrossing(double)	0.0 %	0	4	4	
constrain(double)	33.3 %	2	4	6	
Range(double, double)	50.0 %	1	1	2	
contains(double)	75.0 %	3	1	4	
shift(Range, double)		0	0	0	
getCentralValue()		0	0	0	
getLength()		0	0	0	

可以看到，constrain 方法的分支覆盖率为 33.3%。

切换到行覆盖视图：

RangeTest (2019-5-21 20:42:30)				
Element	Coverage	Covered Lines	Missed Lines	Total Lines
▼ Range	15.5 %	13	71	84
combine(Range, Range)	0.0 %	0	9	9
expand(Range, double, double)	0.0 %	0	9	9
shift(Range, double, boolean)	0.0 %	0	8	8
equals(Object)	0.0 %	0	8	8
expandToInclude(Range, double)	0.0 %	0	7	7
scale(Range, double)	0.0 %	0	6	6
shiftWithNoZeroCrossing(double)	0.0 %	0	5	5
hashCode()	0.0 %	0	5	5
Range(double, double)	62.5 %	5	3	8
intersects(double, double)	0.0 %	0	3	3
constrain(double)	75.0 %	6	2	8
shift(Range, double)	0.0 %	0	1	1
getLength()	0.0 %	0	1	1
getLowerBound()	0.0 %	0	1	1

可以看到，constrain 方法的行覆盖率为 75%。

5. 此时覆盖率没有达到 100%，需要补充更多的测试用例来提高覆盖率。比如针对上面的 constrain 方法：

首先补充完整测试用例：

```
@Test
public void testConstrain() {
    assertEquals(1, testRange.constrain(2), 0.0000001d);
}

@Test
public void testConstrain1() {
    assertEquals(-1, testRange.constrain(-2), 0.0000001d);
}

@Test
public void testConstrain2() {
    assertEquals(0, testRange.constrain(0), 0.0000001d);
}
```

然后【overage As】-【JUnit Test】，查看他的覆盖率情况：分支覆盖分数提升到了 83.3%：

RangeTest (2019-5-23 15:09:55)				
Element	Coverage	Covered Branches	Missed Branches	Total Branches
▼ JFreeChart	19.2 %	10	42	52
▼ src	19.2 %	10	42	52
▼ rangetest	19.2 %	10	42	52
▼ Range.java	19.2 %	10	42	52
▼ Range	19.2 %	10	42	52
intersects(double, double)	0.0 %	0	8	8
expandToInclude(Range, double)	0.0 %	0	6	6
equals(Object)	0.0 %	0	6	6
combine(Range, Range)	0.0 %	0	4	4
expand(Range, double, double)	0.0 %	0	4	4
scale(Range, double)	0.0 %	0	4	4
shift(Range, double, boolean)	0.0 %	0	4	4
shiftWithNoZeroCrossing(double)	0.0 %	0	4	4
Range(double, double)	50.0 %	1	1	2
constrain(double)	83.3 %	5	1	6
shift(Range, double)		0	0	0
contains(double)	100.0 %	4	0	4
getCentralValue()		0	0	0
getLength()		0	0	0
getLowerBound()		0	0	0
getUpperBound()		0	0	0
hashCode()		0	0	0

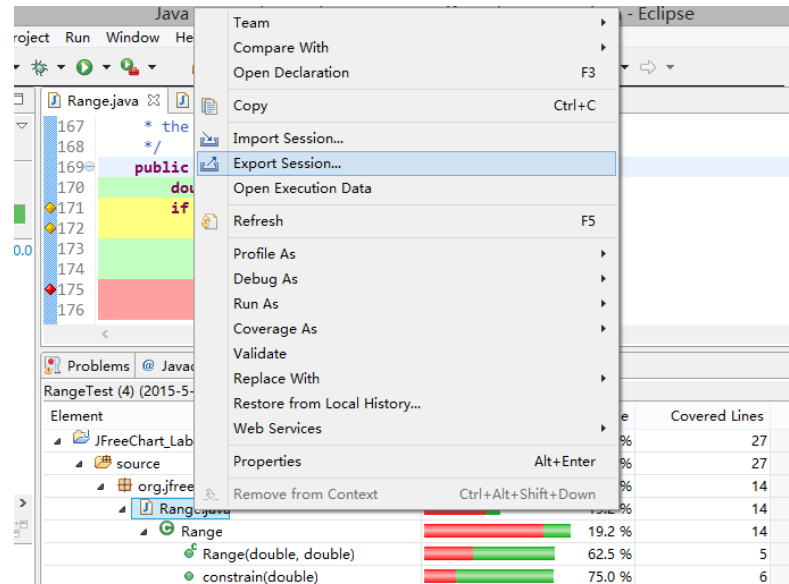
行覆盖分数提升为 100%：

RangeTest (2019-5-23 15:09:55)				
Element	Coverage	Covered Lines	Missed Lines	Total Lines
▼ JFreeChart	<div><div></div></div> 31.0 %	31	69	100
▼ src	<div><div></div></div> 31.0 %	31	69	100
▼ rangetest	<div><div></div></div> 31.0 %	31	69	100
▼ Range.java	<div><div></div></div> 17.9 %	15	69	84
▼ Range	<div><div></div></div> 17.9 %	15	69	84
combine(Range, Range)	<div><div></div></div> 0.0 %	0	9	9
expand(Range, double, dou	<div><div></div></div> 0.0 %	0	9	9
shift(Range, double, boole	<div><div></div></div> 0.0 %	0	8	8
equals(Object)	<div><div></div></div> 0.0 %	0	8	8
expandToInclude(Range, d	<div><div></div></div> 0.0 %	0	7	7
scale(Range, double)	<div><div></div></div> 0.0 %	0	6	6
shiftWithNoZeroCrossing(d	<div><div></div></div> 0.0 %	0	5	5
hashCode()	<div><div></div></div> 0.0 %	0	5	5
Range(double, double)	<div><div></div></div> 62.5 %	5	3	8
intersects(double, double)	<div><div></div></div> 0.0 %	0	3	3
shift(Range, double)	<div><div></div></div> 0.0 %	0	1	1
getLength()	<div><div></div></div> 0.0 %	0	1	1
getLowerBound()	<div><div></div></div> 0.0 %	0	1	1
getUpperBound()	<div><div></div></div> 0.0 %	0	1	1
intersects(Range)	<div><div></div></div> 0.0 %	0	1	1
toString()	<div><div></div></div> 0.0 %	0	1	1
constrain(double)	<div><div></div></div> 100.0 %	8	0	8
contains(double)	<div><div></div></div> 100.0 %	1	0	1
getCentralValue()	<div><div></div></div> 100.0 %	1	0	1

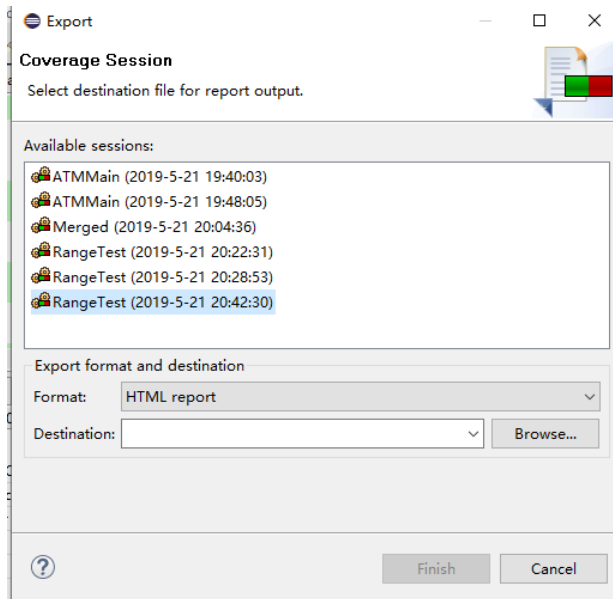
可以看出，对该方法写了完整的测试用例后，虽然行覆盖分数提高到了 100%，但是分支覆盖却依旧没有达到 100%，你需要在实验报告中分析原因。

2.2.4 导出测试覆盖率报告

1. 在 Coverage 视图主区域中点击右键，出现的快捷菜单中选择 “Export Session”



2. 出现下面的 Export 界面：



Available sessions: 要导出的 session

Format: 生成报告的类型 (HTML/XML/Text/EMMA session)

Destination: 导出的 session 存放的位置

2.3 检查你的单元测试覆盖率

1. 在之前的“变异测试”实验中，你已经为 Range.class 中的方法创建了测试用例，用 EclEmma 工具查看你的测试用例的覆盖率情况；
2. 对于未达到 100% 覆盖率的方法，增加测试用例，尽量提高覆盖率。
3. 记录每一个方法的覆盖细节，填写在实验报告中。

3 总结

通过完成这个实验，学生应该对使用 EclEmma 进行测试覆盖率统计有一定的了解。覆盖率统计工具还有很多，其功能大同小异。希望通过本实验，学生能够对覆盖率统计工具的使用、功能等方面有一个初步认识。

4 提交的内容及评分标准

4.1 实验报告

请各位同学下载课程网站上或课程群的实验报告模板“实验 5_测试覆盖率统计_实验报告模板.doc”，然后根据提示进行填写。实验报告命名为“学号_姓名_测试覆盖率统计_实验报告.doc”。

4.2 测试用例及覆盖率报告

导出测试覆盖率报告，Format 选择 HTML report，将导出的所有文件放到一个文件夹中，文件夹命名为：覆盖率报告

新建一个文件夹，命名格式为：姓名-学号-覆盖率统计实验。将你的实验报告 doc 文件、此次的实验项目、以及“覆盖率报告”这个文件夹都放入里面。最后提交“姓名-学号-覆盖率统计.zip”压缩包。

4.3 评分标准

测试用例 80%+实验报告 20%

测试用例得分=Range 类的覆盖率（指令覆盖率 50%+分支覆盖率 50%）

实验报告评分标准	分数比例
1. 引言（针对测试覆盖率、EclEmma 做简单描述）	2%
2. 详细描述你的单元测试策略	2%
3. 覆盖率与变异分数的关系分析	10%
4. 两种测试用例质量评估方法分析	4%
5. 针对本次实验，你遇到的问题和挑战，你的收获以及有什么反馈	2%