**实验4 – 变异测试**

学号: 2017141463145 姓名: 龙行超

分数：

# 引言（5分）

变异测试：在细节方面对源程序进行改动的软件测试方法。这些改动就被称之为变异操作，这些操作可以是：模拟典型的应用程序错误，例如大于等于少写了等于；还能是强制的产生有效测试，例如是每个表达式都等于0。变异操作的目的是：帮助测试者发现有效地测试或者定位测试数据的弱点。

变异算子：在符合语法规则前提下， 变异算子定义了从原有程序生成差别极小程序（即变异体） 的转换规则。例如一个典型的算子将“+”操作符号变异为“-”操作符，那么被测试的程序中表达式a+b>c执行该变异算子之后，将会变为a-b>c。

一阶变异体：原有程序p上执行单一变异算子并形成变异体p1，则称p1为p的一阶变异体。

高阶变异体：原有程序p上依次执行多次变异算子并形成变异体p1，则称p1为p的高阶变异体。

可杀除变异体：若存在测试用例t，在变异体p1和原有程序p上的执行结果不一致，则称该变异体p1相对于测试用例集T是可杀除变异体。

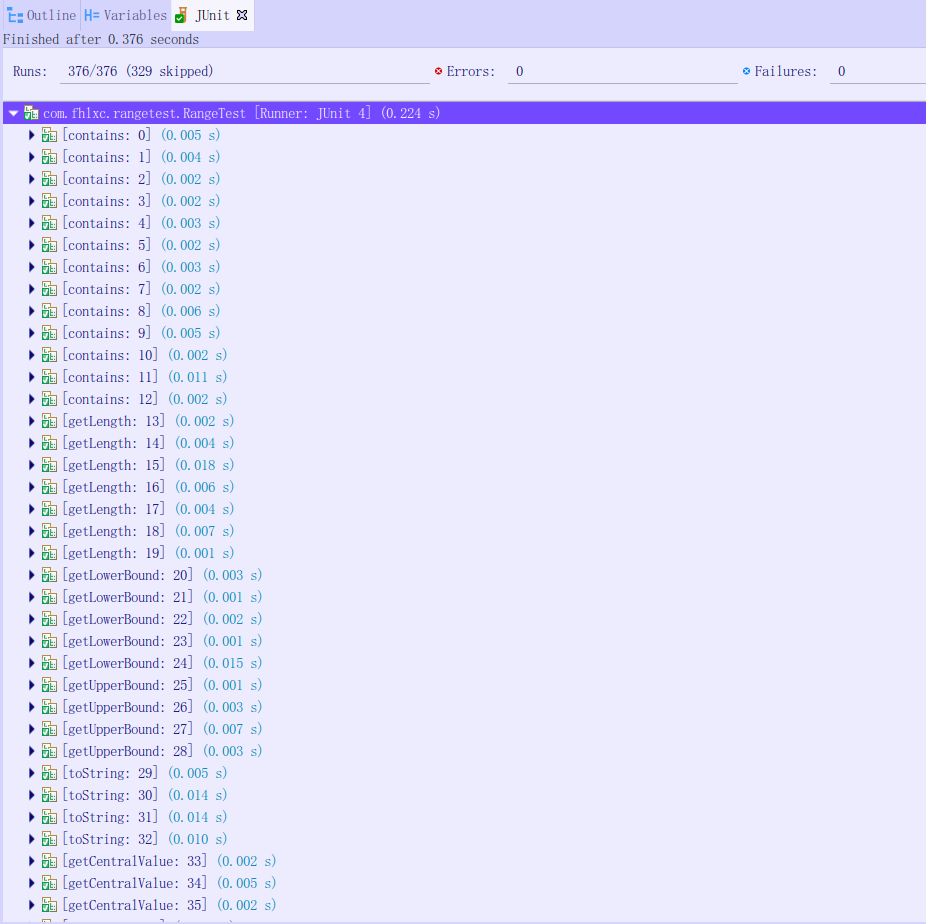
可存活变异体：若不存在任何测试用例t，使得变异体p1和原有程序p上的执行结果不一致，则称该变异体p1相当于测试用例集T是可存活变异体。一般通过设计新的测试用例可杀死一部分可存活测试用例，剩余的可存活变异体可能是等价变异体。

等价变异体：若变异体p1与原有程序p在语法上存在差异，但是在语义上与p一致，则称变异体p1是p的等价变异体。

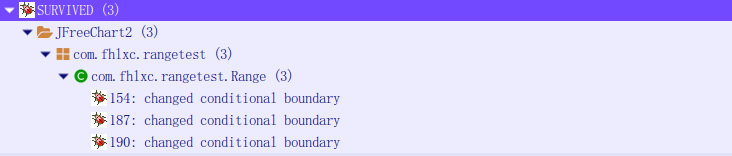
本实验通过eclipse的Pitest插件对上次的实验做变异测试，来完善测试用例。

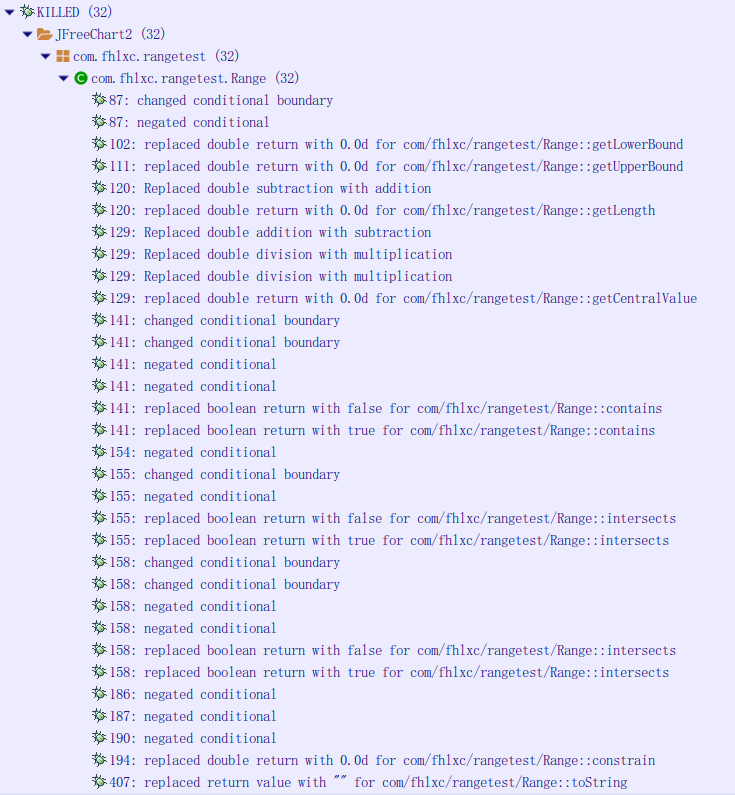
# 变异测试结果分析（90分）

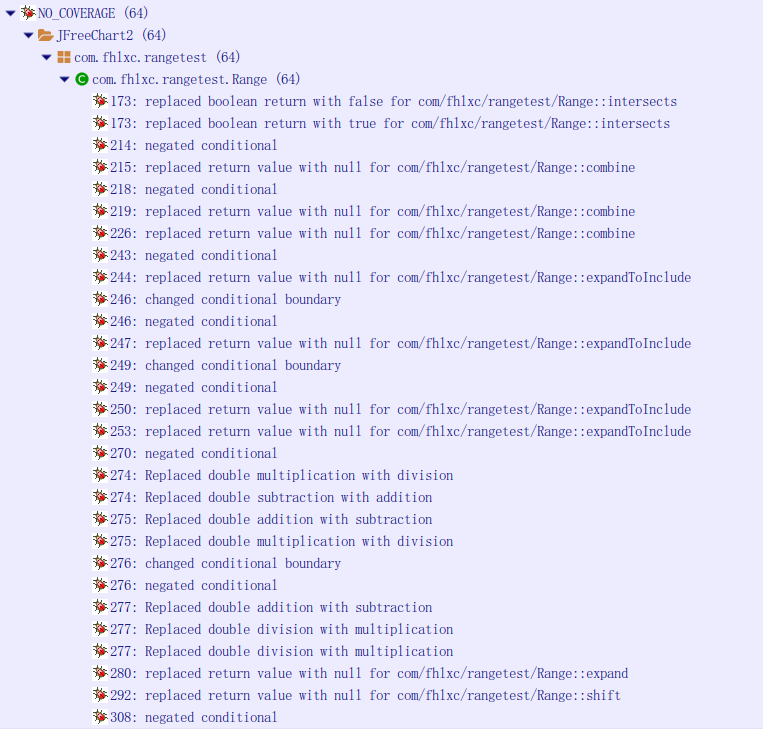
* 1. **测试用例运行结果**

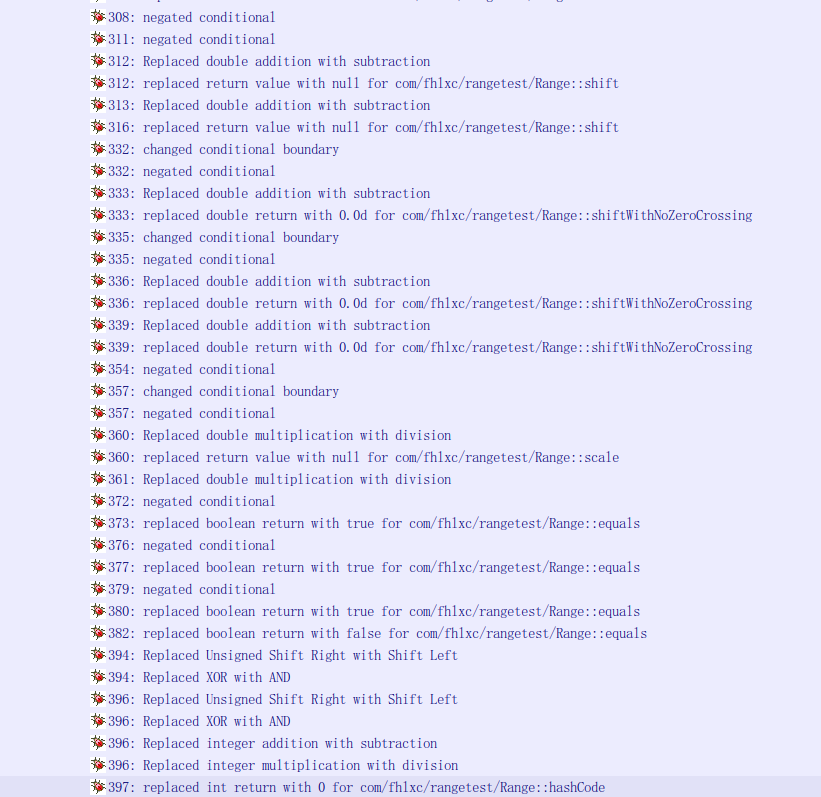


* 1. **变异测试运行结果**

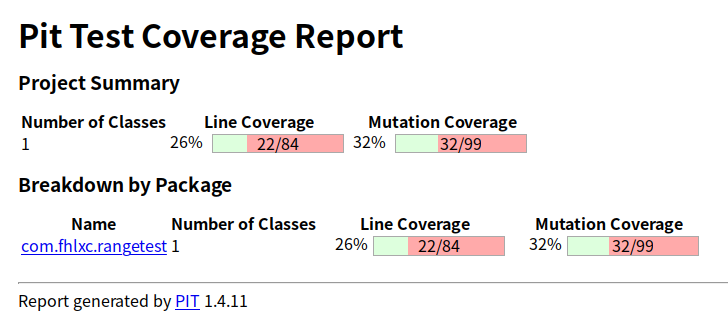




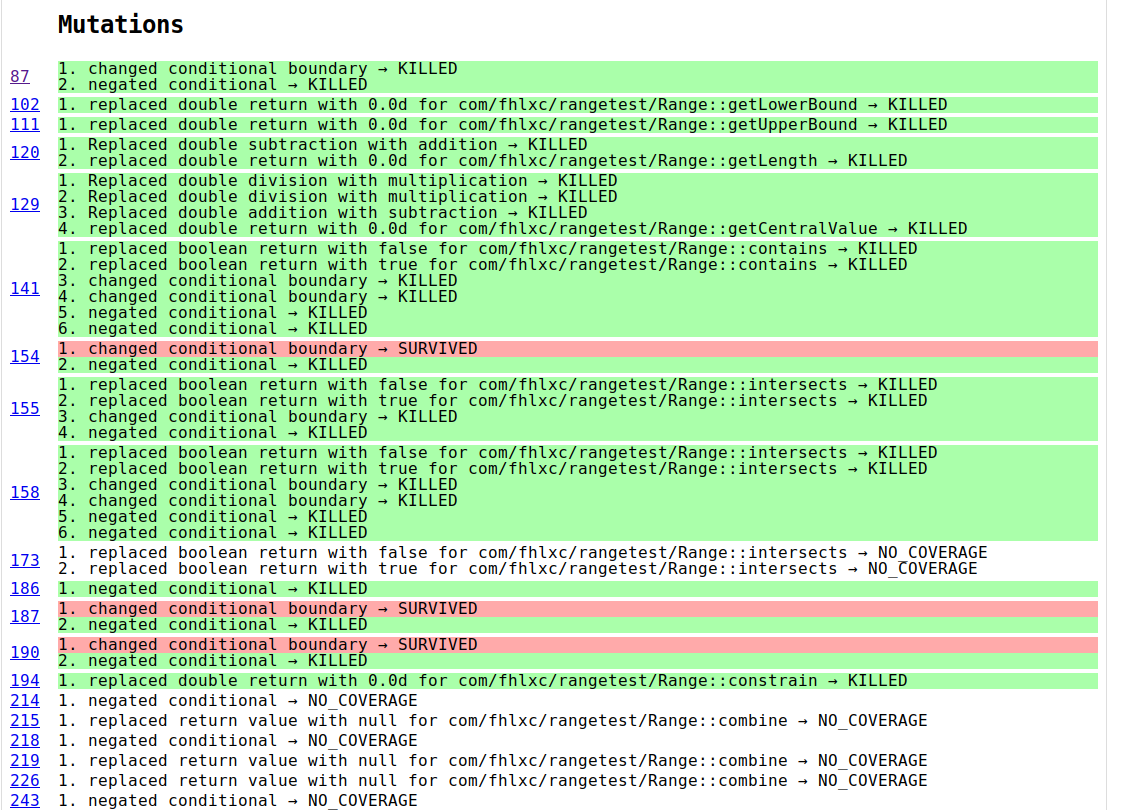


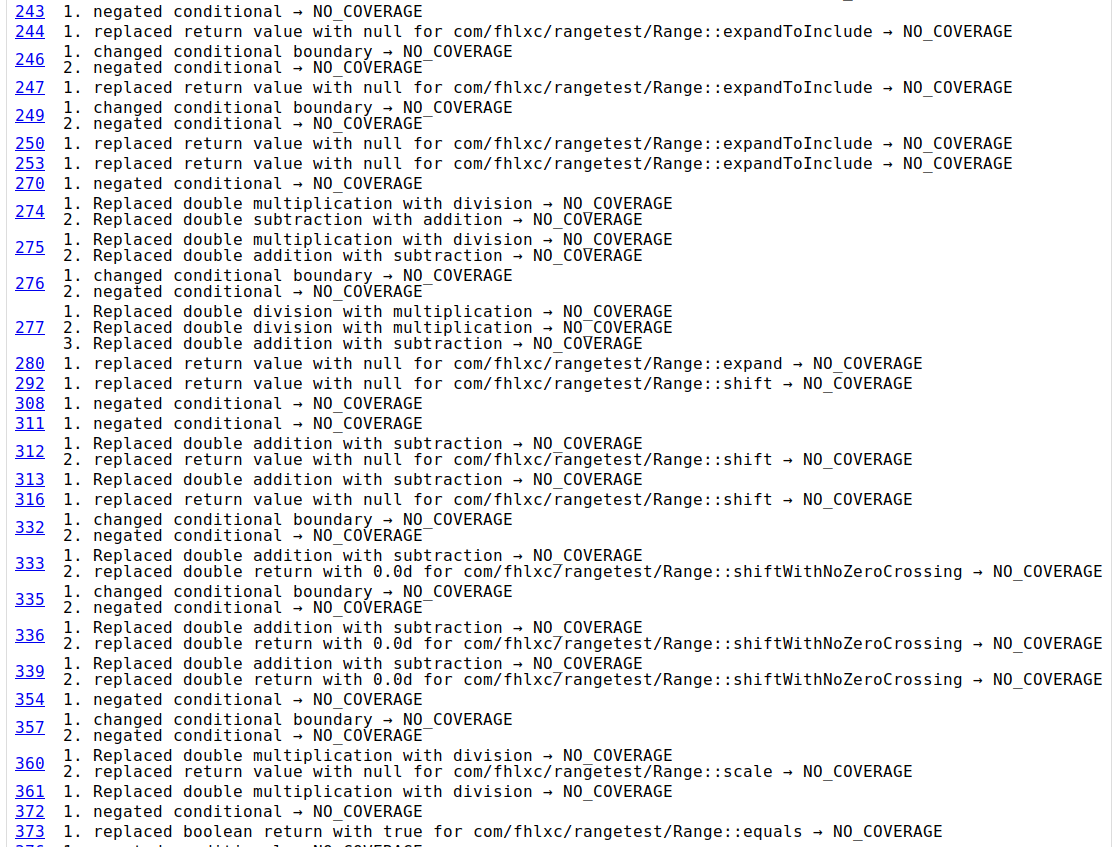


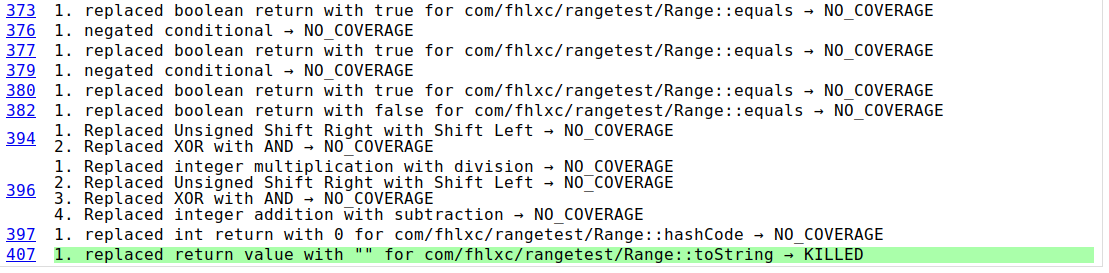
变异覆盖结果



变异覆盖分数截图（很重要）







变异操作截图

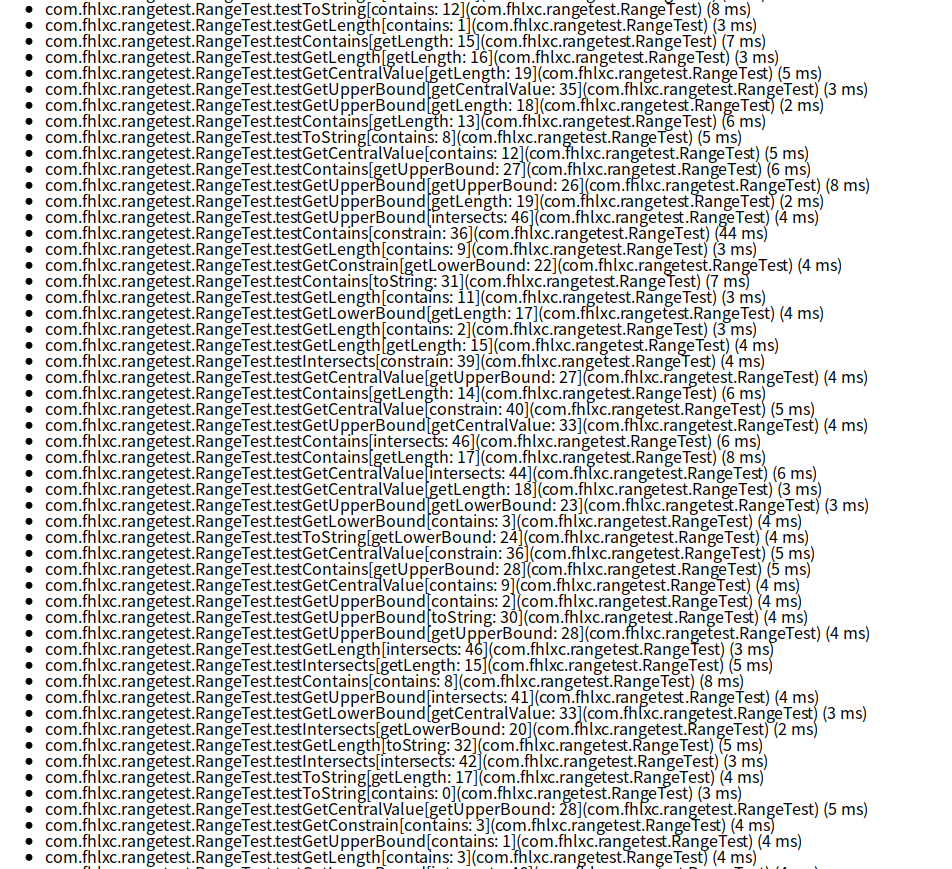
















激活算子和运行时间截图

* 1. **结果分析（蓝色部分为具体行的代码）**

|  |  |
| --- | --- |
| 154行的第1个变异体的变异操作 | changed conditional boundary |
| 状态 | SURVIVED |
| 源代码 | **public** **boolean** intersects(**double** b0, **double** b1) {  **if** (b0 <= **this**.lower) {  **return** (b1 >= **this**.lower);  }  **else** {  **return** (b0 <= **this**.upper && b1 >= b0);  }  } |
| 分析 | 变异操作虽然将b0 <= this.lower变异为b0 < this.lower，这只改变了程序的语法,在语义上没有改变，仍然能够正确输出是否相交，所以这个变异体与原有程序是等价的，因此状态是已存活，并且没有一个测试用例能够杀死这个变异体。 |

|  |  |
| --- | --- |
| 187行的第1个变异体的变异操作 | changed conditional boundary |
| 状态 | SURVIVED |
| 源代码 | **public** **double** constrain(**double** value) {  **double** result = value;  **if** (!contains(value)) {  **if** (value > **this**.upper) {  result = **this**.upper;  }  **else** **if** (value < **this**.lower) {  result = **this**.lower;  }  }  **return** result;  } |
| 分析 | 变异操作虽然将value > this.upper变异为value >= this.upper，这只改变了程序的语法,在语义上没有改变，当值和upper相等时，无论取等与否都是返回的upper，所以变异体与原有程序是等价的，因此状态是已存活，并且没有一个测试用例能够杀死这个变异体。 |

|  |  |
| --- | --- |
| 190行的第1个变异体的变异操作 | changed conditional boundary |
| 状态 | SURVIVED |
| 源代码 | **public** **double** constrain(**double** value) {  **double** result = value;  **if** (!contains(value)) {  **if** (value > **this**.upper) {  result = **this**.upper;  }  **else** **if** (value < **this**.lower) {  result = **this**.lower;  }  }  **return** result;  } |
| 分析 | 变异操作虽然将value < this.lower变异为value <= this.lower，这只改变了程序的语法,在语义上没有改变，当值和lower相等时，无论取等与否都是返回的lower，所以变异体与原有程序是等价的，因此状态是已存活，并且没有一个测试用例能够杀死这个变异体。 |

|  |  |
| --- | --- |
| 173行的第1、2个变异体的变异操作 | 1. replaced boolean return with false for com/fhlxc/rangetest/Range::intersects  2. replaced boolean return with true for com/fhlxc/rangetest/Range::intersects |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **boolean** intersects(Range range) {  **return** intersects(range.getLowerBound(), range.getUpperBound());  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 214行的第1个变异体的变异操作 | negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range combine(Range range1, Range range2) {  **if** (range1 == **null**) {  **return** range2;  }  **else** {  **if** (range2 == **null**) {  **return** range1;  }  **else** {  **double** l = Math.min(range1.getLowerBound(),  range2.getLowerBound());  **double** u = Math.max(range1.getUpperBound(),  range2.getUpperBound());  **return** **new** Range(l, u);  }  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 215行的第1个变异体的变异操作 | replaced return value with null for com/fhlxc/rangetest/Range::combine |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range combine(Range range1, Range range2) {  **if** (range1 == **null**) {  **return** range2;  }  **else** {  **if** (range2 == **null**) {  **return** range1;  }  **else** {  **double** l = Math.min(range1.getLowerBound(),  range2.getLowerBound());  **double** u = Math.max(range1.getUpperBound(),  range2.getUpperBound());  **return** **new** Range(l, u);  }  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 218行的第1个变异体的变异操作 | negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range combine(Range range1, Range range2) {  **if** (range1 == **null**) {  **return** range2;  }  **else** {  **if** (range2 == **null**) {  **return** range1;  }  **else** {  **double** l = Math.min(range1.getLowerBound(),  range2.getLowerBound());  **double** u = Math.max(range1.getUpperBound(),  range2.getUpperBound());  **return** **new** Range(l, u);  }  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 219行的第1个变异体的变异操作 | replaced return value with null for com/fhlxc/rangetest/Range::combine |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range combine(Range range1, Range range2) {  **if** (range1 == **null**) {  **return** range2;  }  **else** {  **if** (range2 == **null**) {  **return** range1;  }  **else** {  **double** l = Math.min(range1.getLowerBound(),  range2.getLowerBound());  **double** u = Math.max(range1.getUpperBound(),  range2.getUpperBound());  **return** **new** Range(l, u);  }  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 226行的第1个变异体的变异操作 | replaced return value with null for com/fhlxc/rangetest/Range::combine |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range combine(Range range1, Range range2) {  **if** (range1 == **null**) {  **return** range2;  }  **else** {  **if** (range2 == **null**) {  **return** range1;  }  **else** {  **double** l = Math.min(range1.getLowerBound(),  range2.getLowerBound());  **double** u = Math.max(range1.getUpperBound(),  range2.getUpperBound());  **return** **new** Range(l, u);  }  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 243行的第1个变异体的变异操作 | negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range expandToInclude(Range range, **double** value) {  **if** (range == **null**) {  **return** **new** Range(value, value);  }  **if** (value < range.getLowerBound()) {  **return** **new** Range(value, range.getUpperBound());  }  **else** **if** (value > range.getUpperBound()) {  **return** **new** Range(range.getLowerBound(), value);  }  **else** {  **return** range;  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 244行的第1个变异体的变异操作 | replaced return value with null for com/fhlxc/rangetest/Range::expandToInclude |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range expandToInclude(Range range, **double** value) {  **if** (range == **null**) {  **return** **new** Range(value, value);  }  **if** (value < range.getLowerBound()) {  **return** **new** Range(value, range.getUpperBound());  }  **else** **if** (value > range.getUpperBound()) {  **return** **new** Range(range.getLowerBound(), value);  }  **else** {  **return** range;  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 246行的第1、2个变异体的变异操作 | 1. changed conditional boundary  2. negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range expandToInclude(Range range, **double** value) {  **if** (range == **null**) {  **return** **new** Range(value, value);  }  **if** (value < range.getLowerBound()) {  **return** **new** Range(value, range.getUpperBound());  }  **else** **if** (value > range.getUpperBound()) {  **return** **new** Range(range.getLowerBound(), value);  }  **else** {  **return** range;  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 247行的第1个变异体的变异操作 | replaced return value with null for com/fhlxc/rangetest/Range::expandToInclude |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range expandToInclude(Range range, **double** value) {  **if** (range == **null**) {  **return** **new** Range(value, value);  }  **if** (value < range.getLowerBound()) {  **return** **new** Range(value, range.getUpperBound());  }  **else** **if** (value > range.getUpperBound()) {  **return** **new** Range(range.getLowerBound(), value);  }  **else** {  **return** range;  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 249行的第1、2个变异体的变异操作 | 1. changed conditional boundary  2. negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range expandToInclude(Range range, **double** value) {  **if** (range == **null**) {  **return** **new** Range(value, value);  }  **if** (value < range.getLowerBound()) {  **return** **new** Range(value, range.getUpperBound());  }  **else** **if** (value > range.getUpperBound()) {  **return** **new** Range(range.getLowerBound(), value);  }  **else** {  **return** range;  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 250行的第1个变异体的变异操作 | replaced return value with null for com/fhlxc/rangetest/Range::expandToInclude |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range expandToInclude(Range range, **double** value) {  **if** (range == **null**) {  **return** **new** Range(value, value);  }  **if** (value < range.getLowerBound()) {  **return** **new** Range(value, range.getUpperBound());  }  **else** **if** (value > range.getUpperBound()) {  **return** **new** Range(range.getLowerBound(), value);  }  **else** {  **return** range;  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 253行的第1个变异体的变异操作 | replaced return value with null for com/fhlxc/rangetest/Range::expandToInclude |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range expandToInclude(Range range, **double** value) {  **if** (range == **null**) {  **return** **new** Range(value, value);  }  **if** (value < range.getLowerBound()) {  **return** **new** Range(value, range.getUpperBound());  }  **else** **if** (value > range.getUpperBound()) {  **return** **new** Range(range.getLowerBound(), value);  }  **else** {  **return** range;  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 270行的第1个变异体的变异操作 | negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range expand(Range range,  **double** lowerMargin, **double** upperMargin) {  **if** (range == **null**) {  **throw** **new** IllegalArgumentException("Null 'range' argument.");  }  **double** length = range.getLength();  **double** lower = range.getLowerBound() - length \* lowerMargin;  **double** upper = range.getUpperBound() + length \* upperMargin;  **if** (lower > upper) {  lower = lower / 2.0 + upper / 2.0;  upper = lower;  }  **return** **new** Range(lower, upper);  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 274行的第1、2个变异体的变异操作 | 1. Replaced double multiplication with division  2. Replaced double subtraction with addition |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range expand(Range range,  **double** lowerMargin, **double** upperMargin) {  **if** (range == **null**) {  **throw** **new** IllegalArgumentException("Null 'range' argument.");  }  **double** length = range.getLength();  **double** lower = range.getLowerBound() - length \* lowerMargin;  **double** upper = range.getUpperBound() + length \* upperMargin;  **if** (lower > upper) {  lower = lower / 2.0 + upper / 2.0;  upper = lower;  }  **return** **new** Range(lower, upper);  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 275行的第1、2个变异体的变异操作 | 1. Replaced double multiplication with division  2. Replaced double addition with subtraction |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range expand(Range range,  **double** lowerMargin, **double** upperMargin) {  **if** (range == **null**) {  **throw** **new** IllegalArgumentException("Null 'range' argument.");  }  **double** length = range.getLength();  **double** lower = range.getLowerBound() - length \* lowerMargin;  **double** upper = range.getUpperBound() + length \* upperMargin;  **if** (lower > upper) {  lower = lower / 2.0 + upper / 2.0;  upper = lower;  }  **return** **new** Range(lower, upper);  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 276行的第1、2个变异体的变异操作 | 1. changed conditional boundary  2. negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range expand(Range range,  **double** lowerMargin, **double** upperMargin) {  **if** (range == **null**) {  **throw** **new** IllegalArgumentException("Null 'range' argument.");  }  **double** length = range.getLength();  **double** lower = range.getLowerBound() - length \* lowerMargin;  **double** upper = range.getUpperBound() + length \* upperMargin;  **if** (lower > upper) {  lower = lower / 2.0 + upper / 2.0;  upper = lower;  }  **return** **new** Range(lower, upper);  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 277行的第1、2、3个变异体的变异操作 | 1. Replaced double division with multiplication  2. Replaced double division with multiplication  3. Replaced double addition with subtraction |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range expand(Range range,  **double** lowerMargin, **double** upperMargin) {  **if** (range == **null**) {  **throw** **new** IllegalArgumentException("Null 'range' argument.");  }  **double** length = range.getLength();  **double** lower = range.getLowerBound() - length \* lowerMargin;  **double** upper = range.getUpperBound() + length \* upperMargin;  **if** (lower > upper) {  lower = lower / 2.0 + upper / 2.0;  upper = lower;  }  **return** **new** Range(lower, upper);  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 280行的第1个变异体的变异操作 | replaced return value with null for com/fhlxc/rangetest/Range::expand |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range expand(Range range,  **double** lowerMargin, **double** upperMargin) {  **if** (range == **null**) {  **throw** **new** IllegalArgumentException("Null 'range' argument.");  }  **double** length = range.getLength();  **double** lower = range.getLowerBound() - length \* lowerMargin;  **double** upper = range.getUpperBound() + length \* upperMargin;  **if** (lower > upper) {  lower = lower / 2.0 + upper / 2.0;  upper = lower;  }  **return** **new** Range(lower, upper);  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 292行的第1个变异体的变异操作 | replaced return value with null for com/fhlxc/rangetest/Range::shift |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range shift(Range base, **double** delta) {  **return** shift(base, delta, **false**);  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 308行的第1个变异体的变异操作 | negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range shift(Range base, **double** delta,  **boolean** allowZeroCrossing) {  **if** (base == **null**) {  **throw** **new** IllegalArgumentException("Null 'base' argument.");  }  **if** (allowZeroCrossing) {  **return** **new** Range(base.getLowerBound() + delta,  base.getUpperBound() + delta);  }  **else** {  **return** **new** Range(shiftWithNoZeroCrossing(base.getLowerBound(),  delta), shiftWithNoZeroCrossing(base.getUpperBound(),  delta));  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 311行的第1个变异体的变异操作 | negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range shift(Range base, **double** delta,  **boolean** allowZeroCrossing) {  **if** (base == **null**) {  **throw** **new** IllegalArgumentException("Null 'base' argument.");  }  **if** (allowZeroCrossing) {  **return** **new** Range(base.getLowerBound() + delta,  base.getUpperBound() + delta);  }  **else** {  **return** **new** Range(shiftWithNoZeroCrossing(base.getLowerBound(),  delta), shiftWithNoZeroCrossing(base.getUpperBound(),  delta));  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 312行的第1、2个变异体的变异操作 | 1. Replaced double addition with subtraction  2. replaced return value with null for com/fhlxc/rangetest/Range::shift |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range shift(Range base, **double** delta,  **boolean** allowZeroCrossing) {  **if** (base == **null**) {  **throw** **new** IllegalArgumentException("Null 'base' argument.");  }  **if** (allowZeroCrossing) {  **return** **new** Range(base.getLowerBound() + delta,  base.getUpperBound() + delta);  }  **else** {  **return** **new** Range(shiftWithNoZeroCrossing(base.getLowerBound(),  delta), shiftWithNoZeroCrossing(base.getUpperBound(),  delta));  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 313行的第1个变异体的变异操作 | Replaced double addition with subtraction |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range shift(Range base, **double** delta,  **boolean** allowZeroCrossing) {  **if** (base == **null**) {  **throw** **new** IllegalArgumentException("Null 'base' argument.");  }  **if** (allowZeroCrossing) {  **return** **new** Range(base.getLowerBound() + delta,  base.getUpperBound() + delta);  }  **else** {  **return** **new** Range(shiftWithNoZeroCrossing(base.getLowerBound(),  delta), shiftWithNoZeroCrossing(base.getUpperBound(),  delta));  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 316行的第1个变异体的变异操作 | replaced return value with null for com/fhlxc/rangetest/Range::shift |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range shift(Range base, **double** delta,  **boolean** allowZeroCrossing) {  **if** (base == **null**) {  **throw** **new** IllegalArgumentException("Null 'base' argument.");  }  **if** (allowZeroCrossing) {  **return** **new** Range(base.getLowerBound() + delta,  base.getUpperBound() + delta);  }  **else** {  **return** **new** Range(shiftWithNoZeroCrossing(base.getLowerBound(),  delta), shiftWithNoZeroCrossing(base.getUpperBound(),  delta));  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 332行的第1、2个变异体的变异操作 | 1. changed conditional boundary  2. negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **private** **static** **double** shiftWithNoZeroCrossing(**double** value, **double** delta) {  **if** (value > 0.0) {  **return** Math.max(value + delta, 0.0);  }  **else** **if** (value < 0.0) {  **return** Math.min(value + delta, 0.0);  }  **else** {  **return** value + delta;  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 333行的第1、2个变异体的变异操作 | 1. Replaced double addition with subtraction  2. replaced double return with 0.0d for com/fhlxc/rangetest/Range::shiftWithNoZeroCrossing |
| 状态 | NO\_COVERAGE |
| 源代码 | **private** **static** **double** shiftWithNoZeroCrossing(**double** value, **double** delta) {  **if** (value > 0.0) {  **return** Math.max(value + delta, 0.0);  }  **else** **if** (value < 0.0) {  **return** Math.min(value + delta, 0.0);  }  **else** {  **return** value + delta;  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 335行的第1、2个变异体的变异操作 | 1. changed conditional boundary  2. negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **private** **static** **double** shiftWithNoZeroCrossing(**double** value, **double** delta) {  **if** (value > 0.0) {  **return** Math.max(value + delta, 0.0);  }  **else** **if** (value < 0.0) {  **return** Math.min(value + delta, 0.0);  }  **else** {  **return** value + delta;  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 336行的第1、2个变异体的变异操作 | 1. Replaced double addition with subtraction  2. replaced double return with 0.0d for com/fhlxc/rangetest/Range::shiftWithNoZeroCrossing |
| 状态 | NO\_COVERAGE |
| 源代码 | **private** **static** **double** shiftWithNoZeroCrossing(**double** value, **double** delta) {  **if** (value > 0.0) {  **return** Math.max(value + delta, 0.0);  }  **else** **if** (value < 0.0) {  **return** Math.min(value + delta, 0.0);  }  **else** {  **return** value + delta;  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 339行的第1、2个变异体的变异操作 | 1. Replaced double addition with subtraction  2. replaced double return with 0.0d for com/fhlxc/rangetest/Range::shiftWithNoZeroCrossing |
| 状态 | NO\_COVERAGE |
| 源代码 | **private** **static** **double** shiftWithNoZeroCrossing(**double** value, **double** delta) {  **if** (value > 0.0) {  **return** Math.max(value + delta, 0.0);  }  **else** **if** (value < 0.0) {  **return** Math.min(value + delta, 0.0);  }  **else** {  **return** value + delta;  }  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 354行的第1个变异体的变异操作 | negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range scale(Range base, **double** factor) {  **if** (base == **null**) {  **throw** **new** IllegalArgumentException("Null 'base' argument.");  }  **if** (factor < 0) {  **throw** **new** IllegalArgumentException("Negative 'factor' argument.");  }  **return** **new** Range(base.getLowerBound() \* factor,  base.getUpperBound() \* factor);  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 357行的第1、2个变异体的变异操作 | 1. changed conditional boundary  2. negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range scale(Range base, **double** factor) {  **if** (base == **null**) {  **throw** **new** IllegalArgumentException("Null 'base' argument.");  }  **if** (factor < 0) {  **throw** **new** IllegalArgumentException("Negative 'factor' argument.");  }  **return** **new** Range(base.getLowerBound() \* factor,  base.getUpperBound() \* factor);  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 360行的第1、2个变异体的变异操作 | 1. Replaced double multiplication with division  2. replaced return value with null for com/fhlxc/rangetest/Range::scale |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range scale(Range base, **double** factor) {  **if** (base == **null**) {  **throw** **new** IllegalArgumentException("Null 'base' argument.");  }  **if** (factor < 0) {  **throw** **new** IllegalArgumentException("Negative 'factor' argument.");  }  **return** **new** Range(base.getLowerBound() \* factor,  base.getUpperBound() \* factor);  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 361行的第1个变异体的变异操作 | Replaced double multiplication with division |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **static** Range scale(Range base, **double** factor) {  **if** (base == **null**) {  **throw** **new** IllegalArgumentException("Null 'base' argument.");  }  **if** (factor < 0) {  **throw** **new** IllegalArgumentException("Negative 'factor' argument.");  }  **return** **new** Range(base.getLowerBound() \* factor,  base.getUpperBound() \* factor);  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 372行的第1个变异体的变异操作 | negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **boolean** equals(Object obj) {  **if** (!(obj **instanceof** Range)) {  **return** **false**;  }  Range range = (Range) obj;  **if** (!(**this**.lower == range.lower)) {  **return** **false**;  }  **if** (!(**this**.upper == range.upper)) {  **return** **false**;  }  **return** **true**;  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 373行的第1个变异体的变异操作 | replaced boolean return with true for com/fhlxc/rangetest/Range::equals |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **boolean** equals(Object obj) {  **if** (!(obj **instanceof** Range)) {  **return** **false**;  }  Range range = (Range) obj;  **if** (!(**this**.lower == range.lower)) {  **return** **false**;  }  **if** (!(**this**.upper == range.upper)) {  **return** **false**;  }  **return** **true**;  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 376行的第1个变异体的变异操作 | negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **boolean** equals(Object obj) {  **if** (!(obj **instanceof** Range)) {  **return** **false**;  }  Range range = (Range) obj;  **if** (!(**this**.lower == range.lower)) {  **return** **false**;  }  **if** (!(**this**.upper == range.upper)) {  **return** **false**;  }  **return** **true**;  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 377行的第1个变异体的变异操作 | replaced boolean return with true for com/fhlxc/rangetest/Range::equals |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **boolean** equals(Object obj) {  **if** (!(obj **instanceof** Range)) {  **return** **false**;  }  Range range = (Range) obj;  **if** (!(**this**.lower == range.lower)) {  **return** **false**;  }  **if** (!(**this**.upper == range.upper)) {  **return** **false**;  }  **return** **true**;  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 379行的第1个变异体的变异操作 | negated conditional |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **boolean** equals(Object obj) {  **if** (!(obj **instanceof** Range)) {  **return** **false**;  }  Range range = (Range) obj;  **if** (!(**this**.lower == range.lower)) {  **return** **false**;  }  **if** (!(**this**.upper == range.upper)) {  **return** **false**;  }  **return** **true**;  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 380行的第1个变异体的变异操作 | replaced boolean return with true for com/fhlxc/rangetest/Range::equals |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **boolean** equals(Object obj) {  **if** (!(obj **instanceof** Range)) {  **return** **false**;  }  Range range = (Range) obj;  **if** (!(**this**.lower == range.lower)) {  **return** **false**;  }  **if** (!(**this**.upper == range.upper)) {  **return** **false**;  }  **return** **true**;  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 382行的第1个变异体的变异操作 | replaced boolean return with false for com/fhlxc/rangetest/Range::equals |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **boolean** equals(Object obj) {  **if** (!(obj **instanceof** Range)) {  **return** **false**;  }  Range range = (Range) obj;  **if** (!(**this**.lower == range.lower)) {  **return** **false**;  }  **if** (!(**this**.upper == range.upper)) {  **return** **false**;  }  **return** **true**;  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 394行的第1、2个变异体的变异操作 | 1. Replaced Unsigned Shift Right with Shift Left  2. Replaced XOR with AND |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **int** hashCode() {  **int** result;  **long** temp;  temp = Double.doubleToLongBits(**this**.lower);  result = (**int**) (temp ^ (temp >>> 32));  temp = Double.doubleToLongBits(**this**.upper);  result = 29 \* result + (**int**) (temp ^ (temp >>> 32));  **return** result;  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 396行的第1、2、3、4个变异体的变异操作 | 1. Replaced integer multiplication with division  2. Replaced Unsigned Shift Right with Shift Left  3. Replaced XOR with AND  4. Replaced integer addition with subtraction |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **int** hashCode() {  **int** result;  **long** temp;  temp = Double.doubleToLongBits(**this**.lower);  result = (**int**) (temp ^ (temp >>> 32));  temp = Double.doubleToLongBits(**this**.upper);  result = 29 \* result + (**int**) (temp ^ (temp >>> 32));  **return** result;  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

|  |  |
| --- | --- |
| 397行的第1个变异体的变异操作 | replaced int return with 0 for com/fhlxc/rangetest/Range::hashCode |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **int** hashCode() {  **int** result;  **long** temp;  temp = Double.doubleToLongBits(**this**.lower);  result = (**int**) (temp ^ (temp >>> 32));  temp = Double.doubleToLongBits(**this**.upper);  result = 29 \* result + (**int**) (temp ^ (temp >>> 32));  **return** result;  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

# 你的感受（5分）

这种测试方式很不一样，颠覆了一直以来我对测试的想法，竟然能反过来，用测试用例用在错误的程序上，得出与正确的程序不一样的结果，表示杀死了这个错误，这个想法真的很别样。

经过这次的实验，开阔了自己的思路，原来还能这么做测试。此外，亲身的使用变异测试工具，让我对这个方法更加的熟悉了，它具体是什么样的东西，在脑中也更加清晰了。

从中，也发现变异测试很依赖变异算子，一些算子在有些情况下，能发现一些潜在的问题，例如返回空指针、取反条件等等。然后，其对等价变异体处理力度不大，仍然存活的变异，分两种，一是测试用例没有涉及到，二是不存在用例能够区分，显然是不可能所有的测试用例都去跑一下，简单的程序还好，一眼能看出语义一致，那么针对复杂的呢？有没有什么分析的方法或者思路，能够验证它是等价变异体呢？