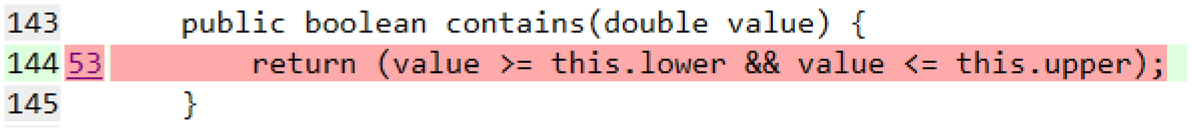
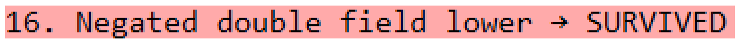
RANGE NOT KILLED

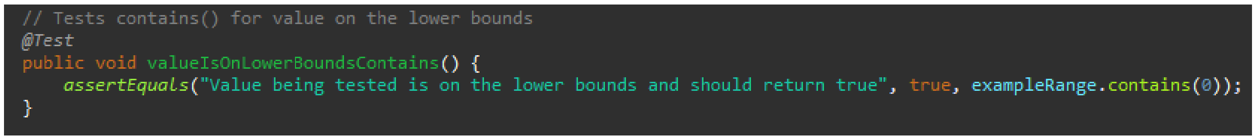
contains()



```
143         public boolean contains(double value) {
144 53          return (value >= this.lower && value <= this.upper);
145         }
```



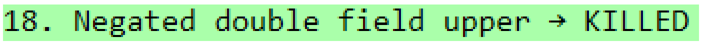
```
16. Negated double field lower → SURVIVED
```



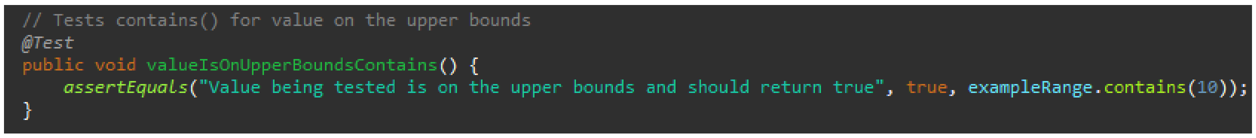
```
// Tests contains() for value on the lower bounds
@Test
public void valueIsOnLowerBoundsContains() {
    assertEquals("Value being tested is on the lower bounds and should return true", true, exampleRange.contains(0));
}
```

This mutation survives since the test case that tests a value on the lower bound uses a range of 0-10. With this mutation, since zero negated is still zero, the test case still passes.

RANGE KILLED



```
18. Negated double field upper → KILLED
```



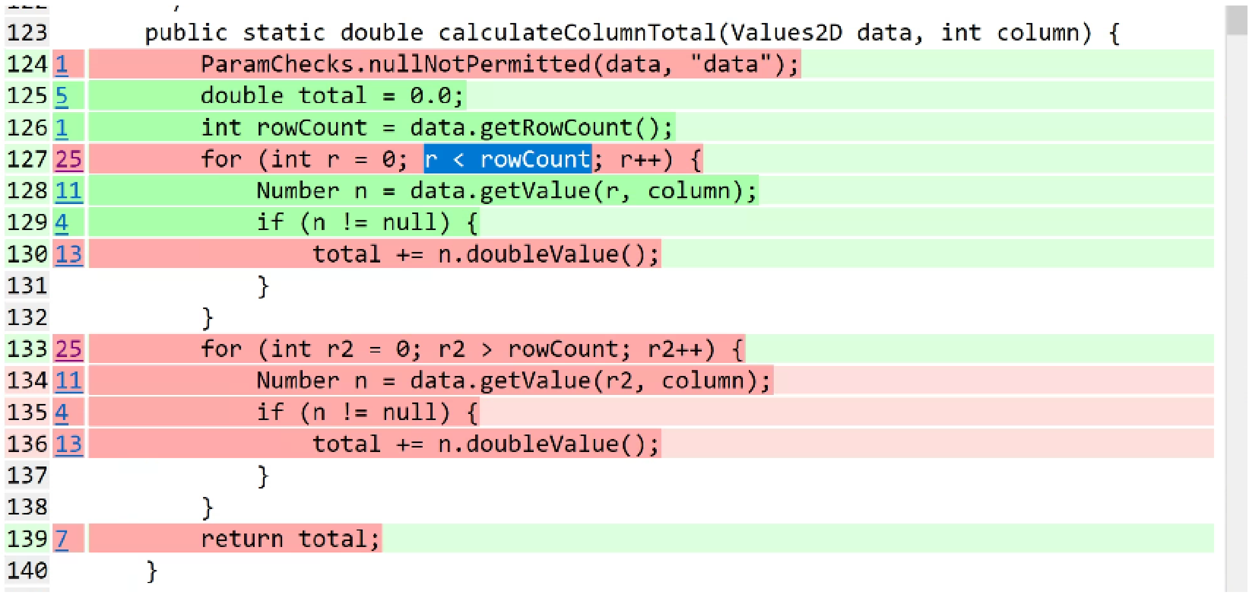
```
// Tests contains() for value on the upper bounds
@Test
public void valueIsOnUpperBoundsContains() {
    assertEquals("Value being tested is on the upper bounds and should return true", true, exampleRange.contains(10));
}
```

This test case uses the range 0-10. Since this mutation negates the upper bound value, 10 becomes -10. The resulting range zero to negative ten does not contain ten so the test case fails. Therefore, the mutation is killed.

DATAUTILITIES NOT KILLED

calculateColumnTotal()

Line 127



```
123         public static double calculateColumnTotal(Values2D data, int column) {
124 1           ParamChecks.nullNotPermitted(data, "data");
125 5           double total = 0.0;
126 1           int rowCount = data.getRowCount();
127 25          for (int r = 0; r < rowCount; r++) {
128 11              Number n = data.getValue(r, column);
129 4               if (n != null) {
130 13                  total += n.doubleValue();
131                 }
132             }
133 25          for (int r2 = 0; r2 > rowCount; r2++) {
134 11              Number n = data.getValue(r2, column);
135 4               if (n != null) {
136 13                  total += n.doubleValue();
137                 }
138             }
139 7           return total;
140         }
```

```
17. Less than to not equal → SURVIVED
```

In line 127, r < rowCount is changed to r != rowCount by the mutation. This is an example of an equivalent class since the condition for r < rowCount and r!= rowCount will have the same result and the for loop will iterate through all of the rows. Since this does not result in a failure, this mutation cannot be killed.

DATAUTILITIES KILLED

```
18. Incremented (a++) integer local variable number 5 → KILLED
```

In line 127, the local integer variable r is incremented. This would result in the for loop getValue method to be calling the row that is 1 more than the one that is intended. For example, instead of data.getValue(r = 0, column = 0), it would be data.getValue(r = 1, column = 0). This would result in a value that is different than the expected, causing a failure in the program. Therefore the mutation is killed.