Team6

Deliverable 5

Introduction

As of now we have 25 tests. The first 10 (test1-test10) deal with the org.openmrs.logic.result component. The next 15 (test11-test25) deal with the org.openmrs.util component. However, only test11-test20 deal with the contains method from the DoubleRange class and only ***test11-test15*** deal with inclusivity and exclusivity.

What we are testing is simple, we create a DoubleRange object with the first two input arguments and then call the contains method with the third input argument. This can be seen in our testCaseExecutables/DoubleRangeContainsTest.java file.

The requirement is: Return true if double is in range and false otherwise, **low end is closed or inclusive**, **high end is open or exclusive.** A way to represent this mathematically is with a closed low end [and an open high end)

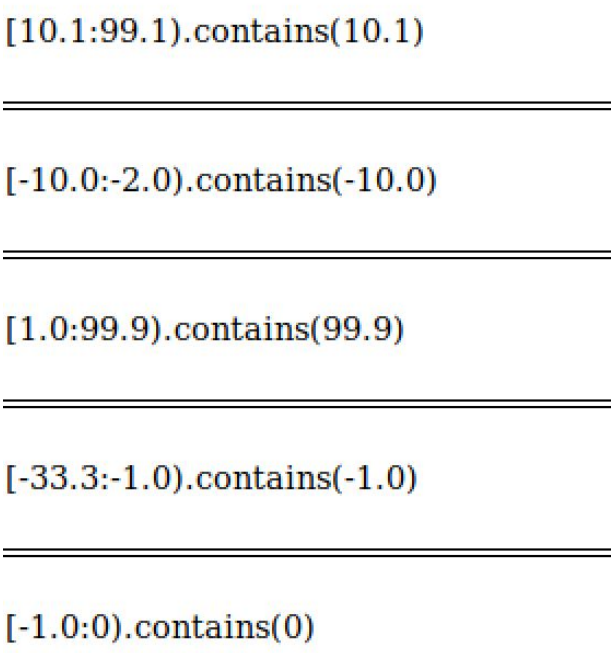

Fig 1.1 Cases test11-test15 all test closed low end and open high end

The code without changes will always say a DoubleRange of say [10.1:99.1) will contain the closed-inclusive low end of 10.1 but will not contain the open-exclusive high end of 99.1. Some

classes like the Apache common similarly-named classed treat both ends as inclusive and were thus not used.

[10.1:99.1).contains(10.1)	true	true	**Pass**
[-10.0:-2.0).contains(-10.0)	true	true	**Pass**
[1.0:99.9).contains(99.9)	false	false	**Pass**
[-33.3:-1.0).contains(-1.0)	false	false	**Pass**
[-1.0:0).contains(0)	false	false	**Pass**

Fig 1.2 Cases test11-test15 all PASS low and high end inclusivity and exclusivity, respectively

Changing the Code and Recompiling

The file we will need to change to *inject faults* can be found here from the root of the project:

TestAutomation/openmrs-core/api/src/main/java/org/openmrs/util/DoubleRange.java

There are two instance boolean variables named closedLow and closedHigh each set to true and false, respectively.

```java
/**
 * This Source Code Form is subject to the terms of the Mozilla Public License,
 * v. 2.0. If a copy of the MPL was not distributed with this file, You can
 * obtain one at http://mozilla.org/MPL/2.0/. OpenMRS is also distributed under
 * the terms of the Healthcare Disclaimer located at http://openmrs.org/license.
 *
 * Copyright (C) OpenMRS Inc. OpenMRS is a registered trademark and the OpenMRS
 * graphic logo is a trademark of OpenMRS Inc.
 */
package org.openmrs.util;

import org.apache.commons.lang3.builder.HashCodeBuilder;

/**
 * Represents a bounded or unbounded numeric range. By default the range is closed (ake inclusive)
 * on the low end and open (aka exclusive) on the high end: mathematically "[low, high)". (I'm not
 * using the similarly-named class from Apache commons because it doesn't implement comparable, and
 * because it only allows inclusive bounds.)
 */
public class DoubleRange implements Comparable<DoubleRange> {

    private Double low;

    private Double high;

    private boolean closedLow = true; //TODO: add setters and getters for these

    private boolean closedHigh = false;
```

Fig1.3 To inject faults to the code we simply need to flip the true and false booleans

Once we've flipped the true and false in the source code we need to recompile the package by going to Team6/TestAutomation/openmrs-core/api
And issue commands:
Team6/TestAutomation/openmrs-core/api $ mvn clean
Team6/TestAutomation/openmrs-core/api $ mvn compile

We run the tests again by calling our script:
Team6/TestAutomation $ ./scripts/runAllTests.sh

And the outputs are now:

[10.1:99.1).contains(10.1)	false	true	**Failure**
[-10.0:-2.0).contains(-10.0)	false	true	**Failure**
[1.0:99.9).contains(99.9)	true	false	**Failure**
[-33.3:-1.0).contains(-1.0)	true	false	**Failure**
[-1.0:0).contains(0)	true	false	**Failure**

Fig 1.4 Just switching where the ends are closed with the boolean variables is enough to inject faults