CS 362 Software Engineering 2
Instructor: Dr. Ali Aburas
Student: **Khuong Luu**

Assignment 5

1. Bug report

1.1. Bug report 1:

Symptom: Program displays error message
Step to reproduce: Run the program
Expected behavior: Display a list of appointments between 2 particular dates
Actual behavior: After displaying all upcoming appointments, the program displays the following error message:

```
Exception in thread "main" java.lang.NullPointerException
	at calendar.DataHandler.getApptOccurences(DataHandler.java:315)
	at calendar.DataHandler.getApptRange(DataHandler.java:235)
	at calendar.CalendarMain.main(CalendarMain.java:164)
```

Delta debugging: I narrowed this bug down to the cause that comes from the constructor of the CalDay doesn't set the date on calendar as valid, which in turn cause the getApptOccurence throw NullPointerException

Caught by: method **testConstructor1** of test class **CalDayUnitTest**

```
CalDayUnitTest.testConstructor1:20 expected:<false> but was:<true>
```

More details from my test output trace:

```
testConstructor1(calendar.CalDayUnitTest)  Time elapsed: 0.001 sec  <<< FAILURE!
java.lang.AssertionError: expected:<false> but was:<true>
	at org.junit.Assert.fail(Assert.java:88)
	at org.junit.Assert.failNotEquals(Assert.java:834)
	at org.junit.Assert.assertEquals(Assert.java:118)
	at org.junit.Assert.assertEquals(Assert.java:144)
	at calendar.CalDayUnitTest.testConstructor1(CalDayUnitTest.java:20)
```

1.2. Bug report 2:

Symptom: None from the user-facing perspective, but caught by some test cases
Step to reproduce: Run the below test cases
Expected behavior: The attribute “valid” of class Appt has a value of “true” after the class is constructed
Actual behavior: The attribute “valid” of class Appt has a value of “false” after the class is constructed

Caught by method **testSetValid** of test class **ApptUnitTest**

```
ApptUnitTest.testSetValid10:197 expected:<true> but was:<false>
ApptUnitTest.testSetValid6:105 expected:<true> but was:<false>
ApptUnitTest.testSetValid7:128 expected:<true> but was:<false>
ApptUnitTest.testSetValid8:143 expected:<true> but was:<false>
ApptUnitTest.testSetValid9:170 expected:<true> but was:<false>
```

More details from my test output:

```
Running calendar.ApptUnitTest
Tests run: 21, Failures: 8, Errors: 1, Skipped: 0, Time elapsed: 0.021 sec <<< FAILURE! - in
calendar.ApptUnitTest
testSetValid10(calendar.ApptUnitTest)  Time elapsed: 0 sec  <<< FAILURE!
java.lang.AssertionError: expected:<true> but was:<false>
	at org.junit.Assert.fail(Assert.java:88)
	at org.junit.Assert.failNotEquals(Assert.java:834)
	at org.junit.Assert.assertEquals(Assert.java:118)
	at org.junit.Assert.assertEquals(Assert.java:144)
	at calendar.ApptUnitTest.testSetValid10(ApptUnitTest.java:197)

testConstructor1(calendar.ApptUnitTest)  Time elapsed: 0.002 sec  <<< FAILURE!
org.junit.ComparisonFailure: expected:<	12/9/2018 at [3:30p]m ,Birthday Party, T...> but was:<
	12/9/2018 at [1:30a]m ,Birthday Party, T...>
	at org.junit.Assert.assertEquals(Assert.java:115)
	at org.junit.Assert.assertEquals(Assert.java:144)
	at calendar.ApptUnitTest.testConstructor1(ApptUnitTest.java:16)

testHasTimeSet1(calendar.ApptUnitTest)  Time elapsed: 0 sec  <<< FAILURE!
java.lang.AssertionError: null
	at org.junit.Assert.fail(Assert.java:86)
	at org.junit.Assert.assertTrue(Assert.java:41)
	at org.junit.Assert.assertFalse(Assert.java:64)
	at org.junit.Assert.assertFalse(Assert.java:74)
	at calendar.ApptUnitTest.testHasTimeSet1(ApptUnitTest.java:267)

testSetValid1(calendar.ApptUnitTest)  Time elapsed: 0.005 sec  <<< ERROR!
java.lang.ArrayIndexOutOfBoundsException: 13
	at calendar.CalendarUtil.NumDaysInMonth(CalendarUtil.java:30)
	at calendar.Appt.setValid(Appt.java:179)
	at calendar.ApptUnitTest.testSetValid1(ApptUnitTest.java:68)

testSetValid2(calendar.ApptUnitTest)  Time elapsed: 0.001 sec  <<< FAILURE!
java.lang.AssertionError: null
	at org.junit.Assert.fail(Assert.java:86)
	at org.junit.Assert.assertTrue(Assert.java:41)
	at org.junit.Assert.assertFalse(Assert.java:64)
	at org.junit.Assert.assertFalse(Assert.java:74)
	at calendar.ApptUnitTest.testSetValid2(ApptUnitTest.java:76)

testSetValid6(calendar.ApptUnitTest)  Time elapsed: 0.001 sec  <<< FAILURE!
java.lang.AssertionError: expected:<true> but was:<false>
	at org.junit.Assert.fail(Assert.java:88)
	at org.junit.Assert.failNotEquals(Assert.java:834)
	at org.junit.Assert.assertEquals(Assert.java:118)
	at org.junit.Assert.assertEquals(Assert.java:144)
	at calendar.ApptUnitTest.testSetValid6(ApptUnitTest.java:105)

testSetValid7(calendar.ApptUnitTest)  Time elapsed: 0 sec  <<< FAILURE!
java.lang.AssertionError: expected:<true> but was:<false>
	at org.junit.Assert.fail(Assert.java:88)
	at org.junit.Assert.failNotEquals(Assert.java:834)
	at org.junit.Assert.assertEquals(Assert.java:118)
	at org.junit.Assert.assertEquals(Assert.java:144)
	at calendar.ApptUnitTest.testSetValid7(ApptUnitTest.java:128)

testSetValid8(calendar.ApptUnitTest)  Time elapsed: 0.001 sec  <<< FAILURE!
java.lang.AssertionError: expected:<true> but was:<false>
	at org.junit.Assert.fail(Assert.java:88)
	at org.junit.Assert.failNotEquals(Assert.java:834)
	at org.junit.Assert.assertEquals(Assert.java:118)
	at org.junit.Assert.assertEquals(Assert.java:144)
	at calendar.ApptUnitTest.testSetValid8(ApptUnitTest.java:143)

testSetValid9(calendar.ApptUnitTest)  Time elapsed: 0 sec  <<< FAILURE!
java.lang.AssertionError: expected:<true> but was:<false>
	at org.junit.Assert.fail(Assert.java:88)
	at org.junit.Assert.failNotEquals(Assert.java:834)
	at org.junit.Assert.assertEquals(Assert.java:118)
	at org.junit.Assert.assertEquals(Assert.java:144)
	at calendar.ApptUnitTest.testSetValid9(ApptUnitTest.java:170)
```

How I found the bug: It was very quick obvious since my test cases' details and traces have pointed it out very clearly.

1.3. Did you find any interesting bugs using your tests, explain why in some details?

- There are a lot of other NullPointerException error. He must have changed something that make my test throw NullPointerException exception (error)

CalDayUnitTest.testIterator1:61 » NullPointer

```
//@Test(timeout = 4000)
  @Test(expected = NullPointerException.class)
  public void testIterator1() throws Throwable  {
    CalDay cal = new CalDay();
    assertEquals(null, cal.iterator());
```

```
 }
```

CalDayUnitTest.testToString2:55 » NullPointer

```
//@Test(timeout = 4000)
  @Test(expected = NullPointerException.class)
  public void testToString2() throws Throwable  {
    CalDay cal = new CalDay();
    assertEquals("", cal.toString());
  }
```

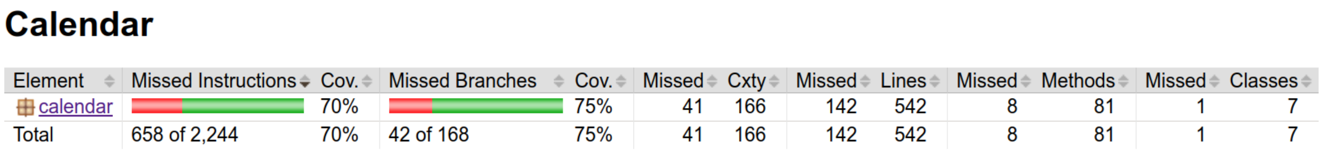
- There are also ArrayOutOfBound exceptions caught by my 2 random tests and 1 (manual) unit test, but I figured the cause was the same as bug #1 and bug #2

```
ApptRandomTest.randomTest:89 » ArrayIndexOutOfBounds 12
ApptUnitTest.testSetValid1:68 » ArrayIndexOutOfBounds 13
CalDayRandomTest.randomTest:69 » ArrayIndexOutOfBounds 12
```

1.4. Explain in some details about the bug(s), such as the cause, how you found it, etc..?

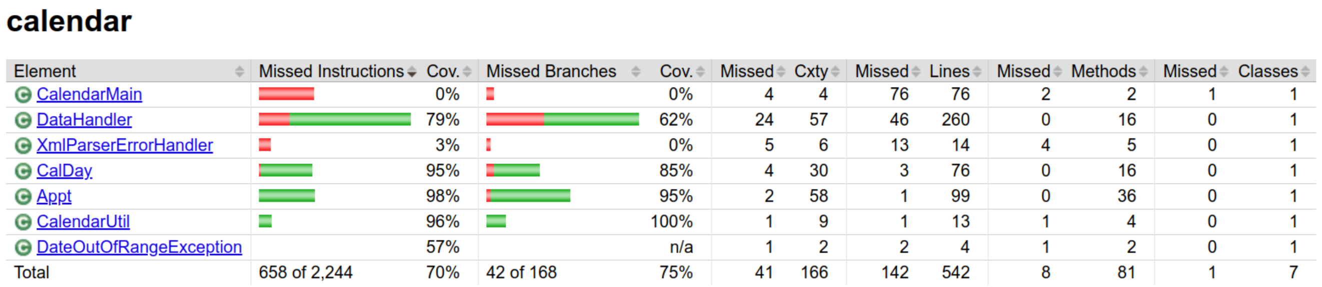
2. Test Report

2.1. Code Coverage Information



Calendar

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
calendar		70%		75%	41	166	142	542	8	81	1	7
Total	658 of 2,244	70%	42 of 168	75%	41	166	142	542	8	81	1	7



calendar

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
CalendarMain		0%		0%	4	4	76	76	2	2	1	1
DataHandler		79%		62%	24	57	46	260	0	16	0	1
XmlParserErrorHandler		3%		0%	5	6	13	14	4	5	0	1
CalDay		95%		85%	4	30	3	76	0	16	0	1
Appt		98%		95%	2	58	1	99	0	36	0	1
CalendarUtil		96%		100%	1	9	1	13	1	4	0	1
DateOutOfRangeException		57%		n/a	1	2	2	4	1	2	0	1
Total	658 of 2,244	70%	42 of 168	75%	41	166	142	542	8	81	1	7

2.2. My view of the reliability of the Calendar code of my teammate.

Because of the critical bugs such as “valid” is set to a wrong value, which is a very important part of the code, I think that the Calendar code is also totally not reliable. Beside of those critical bugs, the code has failed some more of my edgy unit test cases such as null (empty) value, big number values, and out of bound situation. This failing result support my claim even stronger that this code is nowhere to be reliable, and much of work need to be done.

3. Debugging

3.1. Process of identifying and fixing a bug

Narrow down the bug by unit test
Applying Agan’s rule #4 – Divide and Conquer, both in the first and the second bug in section (1), most of the tests are about the validity of the Appt and the CalDay, so it helped me localize (narrow down) the bug to how the valid attribute has been set.

Use debugger to confirm the bug
I used IntelliJ to place a breakpoint at the line where the valid attribute is set in the the constructor in both case of the bugs #1 and #2 since thank to my test cases I have quickly narrowed into these line. Then I ran the debugger to observe the related attribute of the object being constructed and realized the “valid” attribute has been faultily set to a wrong value. Had my test case have been less useful, I would have need to run my debugging many times, step through many lines to narrow down the faulty code location before stopping there to observe the suspected values.

Fix the bug and confirm passing the test – officially confirm the bug has been fixed
Applying Agan’s rule #5 – Change one thing at a time, I change one of the related attribute in the failing method/class one by one. them to the opposite boolean value and the corresponding tests passed, which mean this is the bug. And indeed, they were faultily (and intentionally) set to the opposite boolean value (true to false and false to true)

3.2. Show how you used a debugger (e.g., Eclipse,or IntelliJ) to understand and debug your code.

As I mentioned in section 3.1, step 2 above, I used IntelliJ to place a breakpoint at the line where the valid attribute is set in the the constructor in both case of the bugs #1 and #2 since thank to my test cases I have quickly narrowed into these line.

Then I ran the debugger to observe the related attribute of the object being constructed and realized the “valid” attribute has been faultily set to a wrong value.

Had my test case have been less useful, I would have need to run my debugging many times, step through many lines to narrow down the faulty code location before stopping there to observe the suspected values.

3.3. Did you use any of Agan’s principle in debugging Calendar?

Yes, as mentioned in section 3.1 above, I applied Rule #4 – Divide and Conquer and Rule #5 – Change one thing at a time