EECS 4313 Assignment 2
Black-box and White-box Testing with JUnit

Student Name — Student Number — EECS Account
Edward Vaisman — 212849857 — eddyv
Robin Bandzar — 212200531 — cse23028
Kirusanth Thiruchelvam — 212918298 — kirusant
Sadman Sakib Hasan — 212497509 — cse23152

March 1, 2018

Contents

1 Black Box Testing

1.1 Boundary Value Testing

- Specification of the selected Java methods.
- Justification of the testing technique chosen, i.e., why is it appropriate for this method.
- Description of your application of the three testing strategies. Be clear about which test cases you implemented.
- Evaluation of the test cases derived by the testing technique. Include the screen-shots of the test running results. If you had to complement the derived test cases with special value testing, describe that as well. The marker will not read your code in order to see what you tested. You have to describe it.
- Attaching bug reports if bugs are discovered using your testing methods. You should use the same bug report format as in Assignment 1. Do not file these bug reports to the projects bug report system.

1.2 Equivalence Class Testing

- Specification of the selected Java methods.
- Justification of the testing technique chosen, i.e., why is it appropriate for this method.
- Description of your application of the three testing strategies. Be clear about which test cases you implemented.
- Evaluation of the test cases derived by the testing technique. Include the screen-shots of the test running results. If you had to complement the derived test cases with special value testing, describe that as well. The marker will not read your code in order to see what you tested. You have to describe it.
- Attaching bug reports if bugs are discovered using your testing methods. You should use the same bug report format as in Assignment 1. Do not file these bug reports to the projects bug report system.

1.3 Decision Table Testing

- **Technique**: *Decision Table Testing*
- **Class**: *net.sf.borg.common.DateUtil.java*
- **Method**: *isAfter(Date d1, Date d2)*
- **Method description**: The method checks if a given date *d1* falls on a later calendar day than date *d2*. It returns **true** if *d1* does fall on a later calendar day than *d2* and **false** otherwise.
  - **d1** - The first argument is of type Java Date Object.
  - **d2** - The second argument is of type Java Date Object.
- **Justification**: Decision table testing technique is an appropriate testing technique for this method because there are decision making to be done among the input variables. It consists of logical relationships among the input variables, i.e date *d1* appearing before, after or at the same time as date *d2*, which directly affects the output.

	Rule 1-2	**Rule 3**	**Rule 4**	**Rule 5**	**Rule 6**	**Rule 7**	**Rule 8**
C1: d1 < d2	T	T	T	F	F	F	F
C2: d1 = d2	T	F	F	T	T	F	F
C3: d1 > d2	-	T	F	T	F	T	F
A1: Date is after						X	
A2: Date is not after			X		X		
A3: Impossible	X	X		X			X

Table 1: Decision Table for the isAfter method

Rationale: The decision table above outlines 8 rules. The rules are derived from three equivalence classes: d1 is less than d2, d1 is equal to d2 and d1 is greater than d2. Each equivalence class can have 2 different values (i.e T or F), giving us $2^3 = 8$ rules. Out of the 8 rules, 5 of the rules points to *Impossible* cases, hence cannot be converted to test cases. The other 3 rules are converted into test cases. Below is the code snippet and the test run screenshot for the test cases derived using the decision table technique.

```
@Test
public void testIsAfter() {
        /** Method used: Decision Table Testing **/

        Date d1 = new Date(117, 11, 3);
        Date d2 = new Date(117, 11, 3);
        boolean result;

        // date d1 is equal to d2
        result = DateUtil.isAfter(d1, d2);
        assertFalse("Date d1 is equal to d2", result);

        // date d1 is before d2
        d1.setDate(2);
        result = DateUtil.isAfter(d1, d2);
        assertFalse("Date d1 is before d2", result);

        // date d1 is after d2
        d1.setDate(4);
        result = DateUtil.isAfter(d1, d2);
        assertTrue("Date d1 is after d2", result);
}
```

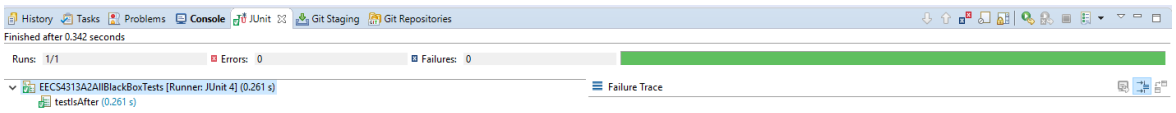

Figure 1: Test results using Decision Table Testing Technique

2 White Box Testing

- The statement coverage measurements for your Assignment 2 test suite.
- A description of the test cases that you added in this assignment to improve statement coverage. The marker will not read your code in order to see what you tested. You have to describe it.
- The statement coverage measurements for your final submission. Include the screenshots of the test running results and the screenshots of the coverage measurement. If your coverage is not 100
- The Control Flow Graph you created. Indicate the segments clearly (you will probably need to include the code for this).
- The path coverage discussion described in section 2 above.
- Attaching bug reports if bugs are discovered using your testing methods. You should use the same bug report format as in Assignment 1. Do not file these bug reports to the projects bug report system.
- An appendix with the specification of the methods you are testing (if there are new ones).