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 5, 2018

Contents

1 White Box Testing

1.1 Equivalence Class Testing

- **Technique**: *Equivalence Class Testing*
- **Class**: *net.sf.borg.common.DateUtil.java*
- **Method**: *minuteString(int mins))*
- **Method Description**: This method generate a human reable string for a particular numbe of minutes.It returns the string interms of hours or minutes or both hours and mintues.
  - **mins** - The argument is an integer
- **Justification**: Equivalence class testing is suitable for this method since argument of this method is an integer which is an independent variable and the entire range of input can be partitioned while assuring disjointness and non-redundancy between each partition set. We have chosen these partition integer range based on when we use minute, minutes, hour, and hours.In order to partition the integer argument into hours and minutes,we divide the Minutes by 60 to get the range of hours and the remainder (minutes % 60) to get the range of the minutes.The paritions for this method are:
  - **Mins / 60 = 1 and Mins % 60 = 0 :** To test 1 hour.
    - Range of hours: [1]
    - Range of minutes: [0]
  - **Mins / 60 = 1 and Mins % 60 = 1 :** To test the 1 hour with 1 minute.
    - Range of hours: (1,+∞)
    - Range of minutes: [1]
  - **Mins / 60 = 1 and Mins % 60 > 1 :** To test 1 hour with minutes more than 1 minute.
    - Range of hours: [1]
    - Range of Minutes : (1, 59]
  - **Mins / 60 > 1 and Mins % 60 = 0 :** To test the hours more than 1 hour.
    - Range of hours: (1,+∞)

  - Range of minutes: [0]

- **Mins / 60 > 1 and Mins % 60 = 1 :** To test hours more than 1 hour with 1 minute.
  - Range of hours: (1,+∞)
  - Range of Minutes :[1]
- **Mins / 60 > 1 and Mins % 60 > 1 :** To test the hours more than 1 hour with minutes more than 1 minute.
  - Range of hours: (1,+∞)
  - Range of Minutes : (1, 59]
- **Mins / 60 = 0 and Mins % 60 = 0 :** To test 0 minutes.
  - Range of hours: [0]
  - Range of minutes: [0]
- **Mins / 60 = 0 and Mins % 60 = 1 :** To test 1 minute.
  - Range of hours: [0]
  - Range of minutes: [1]
- **Mins / 60 = 0 and Mins % 60 > 1 :** To test minutes more than 1 minute and less than 60 minutes or 1 hour.
  - Range of hours: [0]
  - Range of minutes: (1,59]

The method did not specifity how negative minutes should be treated, so we omit the negative integers as an argument for this method.For example, -75 can be converted as -1 hour and 15 minutes or 45 minutes or any other way. Therefore, this case is tested in the whitebox testing thorugh additional test cases after analyzing structure of the method.

- **Statement Coverage Using Black-Box Methods :** 100% [refer to Figure 1 & 2]

- **Justification for getting 100% in blackbox tests :** Since the method converts the interger in terms of human readable minutes and hours, the range of valid outputs can be partition as described in the method description. Since we know how the conversion of integer values into hours and minutes are implemented, there is a need to check negative integers.

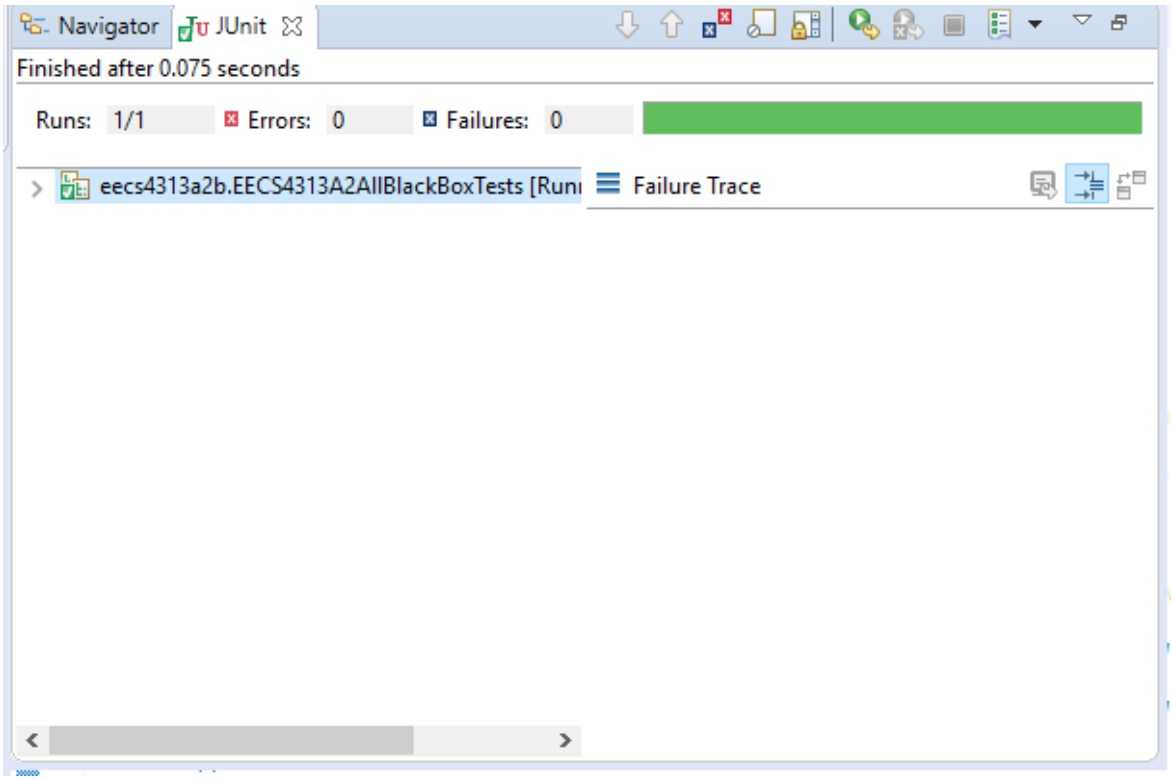

Figure 1: Test run result for the minuteString functionn

DateUtil.java	56.8 %	133	101	234
DateUtil	56.8 %	133	101	234
isAfter(Date, Date)	0.0 %	0	48	48
setToMidnight(Date)	0.0 %	0	26	26
dayOfEpoch(Date)	0.0 %	0	24	24
minuteString(int)	100.0 %	133	0	133

Figure 2: Statement Coverage View for the minuteString function

- **Additional cases:** Since hours are implemented by dividing integer by 60, the negative integers can produce negative hours . Also, negative hours produced emptystring according to the implementation.However, minutes are computed by taking the reminder of the division (mins%60) this will produce a positive number since reminder cannot be negative. Therefore, only two cases we can check in this whitebox testing procedure.They are :

- Mins/60 < 0 and Mins%60 > 1 - To test negative hours with more than 1 minut [Class 10]
  - Range of hours: $(0,-\infty)$
  - Range of minutes: [1]
- Mins/60 < 0 and Mins%60 = 1 - To test negative with 1 minute [Class 11]
  - Range of hours: $(0,-\infty)$
  - Range of minutes: (0,59]

  This covers the two invalid cases possible for the test based on the implementation of the method. since we can only convert negative integers in terms of minutes according to the implementation of the method.This is shown in Figure 3.However these two cases produced a bug because the method is producing negative results instead of positive results since we cannot have negative reminder. this causes the bug. The test cases fail for the two cases are shown in figure 4.

Positive Minutes (Mins % 60 > 1)	Class 10	Class 10	Class 9	Class 3	Class 6
Positive Minute (Mins % 60 = 1)	Class 11	Class 11	Class 8	Class 2	Class 5
0 Minutes (Mins % 60 = 0)			Class 7	Class 1	Class 4
Negative Minute (Mins % 60 = -1)					
Negative Minutes (Mins % 60 < -1)					
	Negative Hours (Mins / 60 < -1)	Negative Hour (Mins / 60 = -1)	0 Hours (Mins / 60 = 0)	Positive Hour (Mins / 60 = 1)	Positive Hours (Mins / 60 > 1)

Figure 3: Possible invalid cases for the minuteString function

Red coloured boxes in Figure 3 are representing impossible cases since negative integers can be only converted to minutes according to the implementation.Yellow

colored boxes show the addtional tests cases that we test in the whitebox and the green boxes are valid outputs that we checked in blackbox test cases.

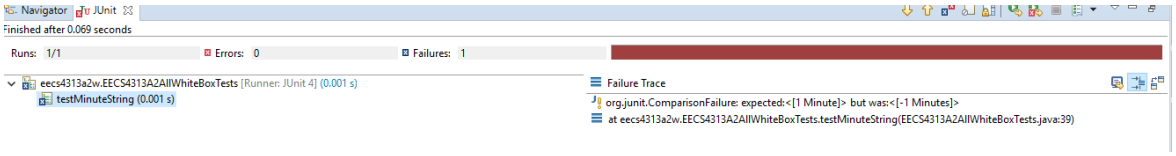

Figure 4: Additional test cases fail due to a bug in the minuteString function

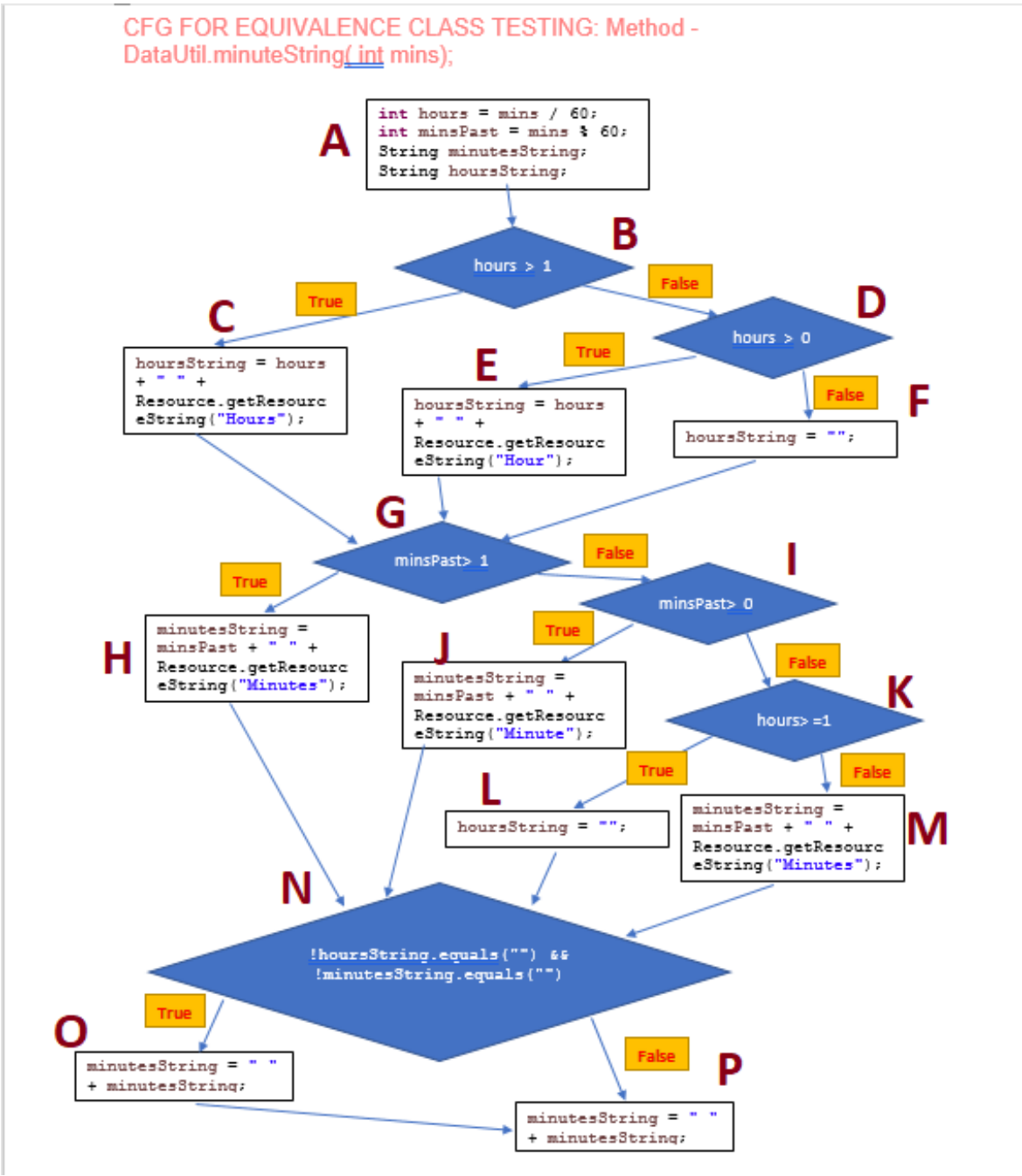

Figure 5: Control Flow Graph for the minuteString function

1.2 Bug Report

- **Bug Report Title**: DateUtil.minuteString Method does not output the correct result for negative integers
- **Reported by**: Kirusanth Thiruchelvam
- **Date reported**: March 3, 2018
- **Program (or component) name**: net.sf.borg.common.DateUtil.minuteString()
- **Configuration(s)**:
  - Operating System: Windows 10 Pro
  - Version: 10.0.1.16299 Build 16299
  - System Manufacturer: SAMSUNG ELECTRONICS CO., LTD.
  - System Model: QX310/QX410/QX510/SF310/SF410/SF510
  - BIOS: AMIBIOS Version 03MX.M005.20101011.SCY
  - Processor: Intel(R) Core(TM) i5 CPU M 460 @ 2.53 GHZ (4CPUs), 2.5GHz
  - Memory: 8192 MB RAM
  - Display Device: Intel(R) HD Graphics (Core i5)
  - BORG Calendar Version: 1.8.3
  - Java Version: 1.8.0_161
- **Report type**: Coding Error
- **Reproducibility**: 100%
- **Severity**: Low
- **Problem summary**: When inputing a negative integer as an argument for the minuteString method in DataUtil class. It produces the number in negative which is not correct since the reminder of the negative integer cannot be ne negaitive.
- **Problem description**:
  Steps to Reproduce
  1. Load the source code of Borg Calender in Java
  2. Create a new junit test case
  3. call the net.sf.borg.common.DateUtil.minuteString() in the class with -70 as an argument

Results

- Expected Results: After inputing the -70, it should gives 10 as the output since hour produce empty string and only minutes produce the results.The expected result is to see positive 10 but it gives -10 since (-70) mod 60 is 10 minutes.
- Real Results: The actual result produce the bug since it shows -10 as the outputs.
- **New or old bug**: New