Chapter 5

Team 3: Meg, Mitch, Itzayana

CSCI 362 Project: OpenMrs

Experience

Our experience for this portion of the project was not as substantial as other portions due to the fact that it was such a short time between chapter 5 and chapter 4. With that being said, we did learn a few lessons about the importance of readability in our test cases. Dr. Bowring mentioned to us that passing in the date in the form of milliseconds made two of our test cases very hard to read. To remedy this issue, we decided to pass in the date via a string, the format goes: year, month, day, hour, minute, second, each separated by a "/". For examples, "1990/12/25/10/20/20" would be the date Dec 25th, 1990 at 10:20:20 am. This makes reading the result output much easier.

The Second thing we had to do this sprint is put in a few errors in our program and see whether our automated testing framework would catch these errors. The way we decided to implement these errors was through comments. In each method there is a comment that contains a snpit of code with an error in it. To run the code with the error in it simply uncomment that code segment and comment out the segment above it. Down below there are step by step instructions on how to insert these same errors into the code.

Error Insert Instructions:

1. Open the OpenmrsUtil.java file
    a. Comment out line 27, this will remove the correct code segment
    b. Uncomment line 28, this will activate the error in the code
    c. Comment out line 46, this will remove the correct code segment
    d. Uncomment line 47, this will activate the error in the code

e. Comment out line 68, this will remove the correct code segment
f. Uncomment line 69, this will activate the error in the code
g. Save the OpenmrsUtil.java file

2. Open the DateUtil.java file
    a. Comment out line 20, this will remove the correct code segment
    b. Uncomment line 21, this will activate the error in the code

Error Results

getLastMomentOfDay

For the getLastMomentOfDay method, we decided that the error we would implement would be to set the seconds to be 58 instead of 59. This seemed like an error that could easily occur if someone was typing fast or didn't read the requirements. When we ran our script with the error in place, it caught the error in all but one. This result makes sense, because the 58 was a hardcoded mistake and as a result it affected all the test cases.

```
calender.set(Calendar.SECOND, 59);
calender.set(Calendar.SECOND, 58); //THIS IS AN ERROR
```

Test ID	Class Name	Summary	Method Type	Inputs	Expected Outputs	Driver	Result	Pass/Fail
01	OpenmrsUtil	This method will take a date and time, then shift the time to be the last second of the day on that date.	getLastMomentOfDay	1969/12/31/1/1/1	Wed Dec 31 23:59:59 EST 1969	getLastMomentOfDayDriver	Wed Dec 31 23:59:58 EST 1969	Failed
02	OpenmrsUtil	This method will take a date and time, then shift the time to be the last second of the day on that date.	getLastMomentOfDay	1800/9/7/19/21/0	Sun Sep 07 23:59:59 EST 1800	getLastMomentOfDayDriver	Sun Sep 07 23:59:58 EST 1800	Failed
03	OpenmrsUtil	This method will take a date and time, then shift the time to be the last second of the day on that date.	getLastMomentOfDay	1969/12/31/4/4/4	Wed Dec 31 23:59:59 EST 1969	getLastMomentOfDayDriver	Wed Dec 31 23:59:58 EST 1969	Failed
04	OpenmrsUtil	This method will take a date and time, then shift the time to be the last second of the day on that date.	getLastMomentOfDay	2001/9/8/6/56/0	Sat Sep 08 23:59:59 EDT 2001	getLastMomentOfDayDriver	Sat Sep 08 23:59:58 EDT 2001	Failed
05	OpenmrsUtil	This method will take a date and time, then shift the time to be the last second of the day on that date.	getLastMomentOfDay	1998/12/20/20 /45/30	Sun Dec 20 23:59:59 EST 1998	getLastMomentOfDayDriver	Sun Dec 20 23:59:58 EST 1998	Failed

How to implement this error in getLastMomentOfDay:

1. Open the OpenmrsUtil.java file
2. Comment out line 27, this will remove the correct code segment
3. Uncomment line 28, this will activate the error in the code

DateUtil

For the DataUtil method, we decided that the error we would implement would be to truncate hours instead of seconds. By truncating by the hour, the seconds section of time will always be 00 rather than the actual number of seconds. When we ran our script with the error in place, it caught the error in all cases. In that case the seconds and hours time segments had been set to zero anyway so truncating the seconds didn't make a difference.

```
Instant instant = date.toInstant().truncatedTo(ChronoUnit.SECONDS);
Instant instant = date.toInstant().truncatedTo(ChronoUnit.HOURS); //THIS IS THE ERROR
```

06	DateUtil	This method will return a Date and time with the milliseconds truncated	truncateToSeconds	1969/12/31/19 /20/34	Wed Dec 31 19:20:34 EST 1969	DateUtilDriver	Wed Dec 31 19:00:00 EST 1969	Failed
07	DateUtil	This method will return a Date and time with the milliseconds truncated	truncateToSeconds	393038/10/28/13 /14/15	Sun Oct 28 13:14:15 EDT 393038	DateUtilDriver	Sun Oct 28 13:00:00 EDT 393038	Failed
08	DateUtil	This method will return a Date and time with the milliseconds truncated	truncateToSeconds	1969/12/31/19 /00/00	Wed Dec 31 19:00:00 EST 1969	DateUtilDriver	Wed Dec 31 19:00:00 EST 1969	Passed
09	DateUtil	This method will return a Date and time with the milliseconds truncated	truncateToSeconds	1969/12/21/23 /37/30	Sun Dec 21 23:37:30 EST 1969	DateUtilDriver	Mon Dec 22 00:00:00 EST 1969	Failed
10	DateUtil	This method will return a Date and time with the milliseconds truncated	truncateToSeconds	1800/11/30/12 /01/36	Sun Nov 30 12:01:36 EST 1800	DateUtilDriver	Sun Nov 30 13:00:00 EST 1800	Failed

How to implement this error in DateUtil:

1. Open the DateUtil.java file
2. Comment out line 20, this will remove the correct code segment
3. Uncomment line 21, this will activate the error in the code

containsOnlyDigits

For the containsOnlyDigits method, we decided that the error we would implement would be to return if a string is empty instead of if it's not. If a string makes it to the end of the program it will return the opposite of what we want. When we ran our script with the error in place, the error was not caught in the test cases that were expected to be false. This was confusing at first, but after going back and looking at the code we realized that this method had a separate return statement that is triggered when a string contains something other than digits. Which left those cases untouched by our error.

```
return !test.isEmpty();
return test.isEmpty(); //THIS IS THE ERROR
```

11	OpenmrsUtil	This method returns true if a string only contains digits	containsOnlyDigits	808760	true	containsOnlyDigitsDriver	false	Failed
12	OpenmrsUtil	This method returns true if a string only contains digits	containsOnlyDigits	-842	false	containsOnlyDigitsDriver	false	Passed
13	OpenmrsUtil	This method returns true if a string only contains digits	containsOnlyDigits	8940	true	containsOnlyDigitsDriver	false	Failed
14	OpenmrsUtil	This method returns true if a string only contains digits	containsOnlyDigits	34hi80	false	containsOnlyDigitsDriver	false	Passed
15	OpenmrsUtil	This method returns true if a string only contains digits	containsOnlyDigits	60 78	false	containsOnlyDigitsDriver	false	Passed

How to implement this error in containsOnlyDigits:

1. Open the OpenmrsUtil.java file
2. Comment out line 68, this will remove the correct code segment
3. Uncomment line 69, this will activate the error in the code

convertToInteger

For the convertToInteger method, we decided that the error we would implement would be to flip the greater than and less than symbols in an if-statement. This error will make the program think that convertible numbers are invalid. When we ran our script with the error in place, the error was not caught in the test cases that were expected to return a null. This makes sense because now the range of numbers that will trigger that if-statement is all possible numbers. So any number I put in as input will return a null.

```
if (longValue < Integer.MIN_VALUE || longValue > Integer.MAX_VALUE) {
if (longValue > Integer.MIN_VALUE || longValue < Integer.MAX_VALUE) { //THIS IS THE ERROR
```

16	OpenmrsUtil	This method turns a long into an integer, as long as it does not exceed the int memory limit	convertToInteger	600	600	convertToIntegerDriver	null	Failed
17	OpenmrsUtil	This method turns a long into an integer, as long as it does not exceed the int memory limit	convertToInteger	6000000000	null	convertToIntegerDriver	null	Passed
18	OpenmrsUtil	This method turns a long into an integer, as long as it does not exceed the int memory limit	convertToInteger	-83878527402	null	convertToIntegerDriver	null	Passed
19	OpenmrsUtil	This method turns a long into an integer, as long as it does not exceed the int memory limit	convertToInteger	2147483647	2147483647	convertToIntegerDriver	null	Failed
20	OpenmrsUtil	This method turns a long into an integer, as long as it does not exceed the int memory limit	convertToInteger	-2147483648	-2147483648	convertToIntegerDriver	null	Failed

How to implement this error in convertToInteger:

1. Open the OpenmrsUtil.java file
2. Comment out line 82, this will remove the correct code segment
3. Uncomment line 83, this will activate the error in the code

containsUpperAndLower

For the containsUpperAndLower method, we decided that the error we would implement would take out the part of the pattern that recognized Uppercase letters. This error will make the program return true as long as the input contains lower case letters. When we ran our script with the error in place, the error was not caught in the test cases that contained only lowercase letters. This makes sense because now the inputs with both upper and lowercase letters would still pass the test because this error only checks for lowercase and returns true if an input contains lowercase letters.

```
Pattern pattern = Pattern.compile("^(?=.*?[A-Z])(?=.*?[a-z])[\\w|\\W]*$");
Pattern pattern = Pattern.compile("(?=.*?[a-z])[\\w|\\W]*$"); //THIS IS THE ERROR
```

21	OpenmrsUtil	This method returns true if a string has upper and lower case letters	containsUpperAndLowerCase	HELLO THERE	false	containsUpperAndLowerCaseDriver	false	Passed
22	OpenmrsUtil	This method returns true if a string has upper and lower case letters	containsUpperAndLowerCase	how are you today	false	containsUpperAndLowerCaseDriver	true	Failed
22	OpenmrsUtil	This method returns true if a string has upper and lower case letters	containsUpperAndLowerCase	I'm well how are YOU?	true	containsUpperAndLowerCaseDriver	true	Passed
24	OpenmrsUtil	This method returns true if a string has upper and lower case letters	containsUpperAndLowerCase	Hi12	true	containsUpperAndLowerCaseDriver	true	Passed
25	OpenmrsUtil	This method returns true if a string has upper and lower case letters	containsUpperAndLowerCase	hejfbjsd83eh	false	containsUpperAndLowerCaseDriver	true	Failed

How to implement this error in containsUpperAndLower:

1. Open the OpenmrsUtil.java file
2. Comment out line 46, this will remove the correct code segment
3. Uncomment line 47, this will activate the error in the code