OpenMRS Automated Testing Framework

Team 2: Wright Ledbetter & Sam Lemon

CSCI 362: Software Engineering

Dr. Jim Bowring   College of Charleston

Contents

Introduction

OpenMRS is and open-source java-based patient medical record system that focuses on providing healthcare providers with a free customizable electronic record system (EMR). The goal of OpenMRS is to improve healthcare delivery in resource-constrained environments by providing a robust, user-driven medical record system.

We chose OpenMRS because there are millions of people infected with diseases such as HIV/AIDS, and drug resistant diseases like malaria every day in developing countries. In these countries a lack of education and resources ups the number of preventable deaths. Initially OpenMRS was built to provide care for HIV cases in Kenya and Rwanda, however since is completion it has expanded is depths to be more generic in what records are stored such as patient-care, observations made, encounters, notes, and other data views that strive to improve the effectiveness of the healthcare providers using the system.

Chapter 1

Our first task for the term project was deciding on a HFOSS program to begin building a framework for. We decided to go with ActivityInfo.

ActivityInfo is a Monitoring and Evaluation app for storing and reporting data. It was created to provide aid for organizations working in dispersed and remote areas. The software is all web based for quick and easy set up. ActivityInfo makes it easy for even non-technical program managers to set up a database for their project. Activity info also has a built in geographic database. This allows for all data collected by your organization to be mapped effortlessly.

We have begun the process to building a clone of the ActivityInfo repository into our repository. We have installed all the prereq software and programs listed in the ActivityInfo contributor's guide such as Java JDK, SQL, and Gradle.

We were able to complete the initial build and configuration of the project but it failed at 73% execution due to ActivityInfo failing to create and update a test SQL database. We previously were stuck at 43% execution because we had SQL 5.7 installed instead of 5.5 (because ActivityInfo was abandoned in 2017). We are unsure where to go from here with our SQL errors.

Chapter 2

Since Deliverable 1 we have switched Projects from ActivityInfo to Sakai which was a resource to help educators build websites for schools and courses. This original change was due to poor documentation of how to even build ActivityInfo and it also being an abandoned project as of 2017. Our work on Sakai developed much faster than ActivityInfo, but once we opened up its source code we discovered that it was again very unorganized with poor documentation (about 80+ directories in no discernible order). We attempted contacting the Sakai development team, but received no response, however we were able to contact another contributor, but he was unable to provide us with any information. Thus, bringing us to openMRS – a medical records storage app, we have had a much greater level of success with this project, only starting it 2 days ago, but we have already surpassed anything we have done with the other two projects. we were able to get the openMRS webapp up and running in the span of a few hours whereas the past projects took a couple of days worth of banging our heads against a wall to even build successfully.

Test Plan

Testing Process

The testing process will begin by testing openMRS methods via the terminal.

Requirements Trace-ability

- CANNOT be a getter or setter method (can be boolean) and does not take an object. preferably takes a string and outputs something
- isValid(String identifier) from api/src/main/java/org/openmrs/patient/impl/BaseHyphenatedIdentifierValidator this one is good but you have to go thru the LuhnIdentifierValidator.java because BHIV is abstract.
- getCheckDigit(String undecoratedIdentifier) this one is also good and I know it is a "get" but it does math using ASCII and Bowring gave me the go ahead to use this method. It is in LuhnIdentifierValidator.java
- hasPrivilege(String privilege) from /api/src/main/java/org/openmrs/User.java there may be more in this java file
- hasPrivilege(String privilege) from /api/src/main/java/org/openmrs/Role.java
- removeProxyPrivilege(String privilege) from /apt/src/main/java/org/openmrs/api/context/UserContext.java

Tested Items

- test the getPatientID(): this method should return the correct integer identification number for the specific given patient.
- test the getAllergyStatus(): this method should returns the allergy status of a patient, and is maintained by the supporting infrastructure.
- test the setPatientID(): this method should set the internal identifier for a specific patient in the medical database.
- test the getFullName(Locale locale): this method should return the entire full name of the drug followed by the concept name of the drug.
- test the setQuantity(Double quantity): this method should set the quantity for the drug order.

Test Schedule

Sept.28th - Oct.11th

- have the 5 original test cases completed, with at least another 10 being developed.

Oct.12th - Oct.26th

- have the last 10 test cases completed.

Oct. 29th

- have Deliverable #3 complete and ready to present to the class.

Test Recording Procedures

We plan to log the time, data, and outcome of each test. We plan on implementing a script which outputs the results of each test to a text file.

Hardware/Software Requirements

The app will require a personal computer that is capable of running a Unix OS, openMRS, Tomcat/Jetty, Maven, MYSQL(for testing only) and Eclipse.

Constraints

The system must be built using an open-source Unix operating system, and all test cases must be coded in Java. There is also a time constraint as there is only so much we can do in a college semester.

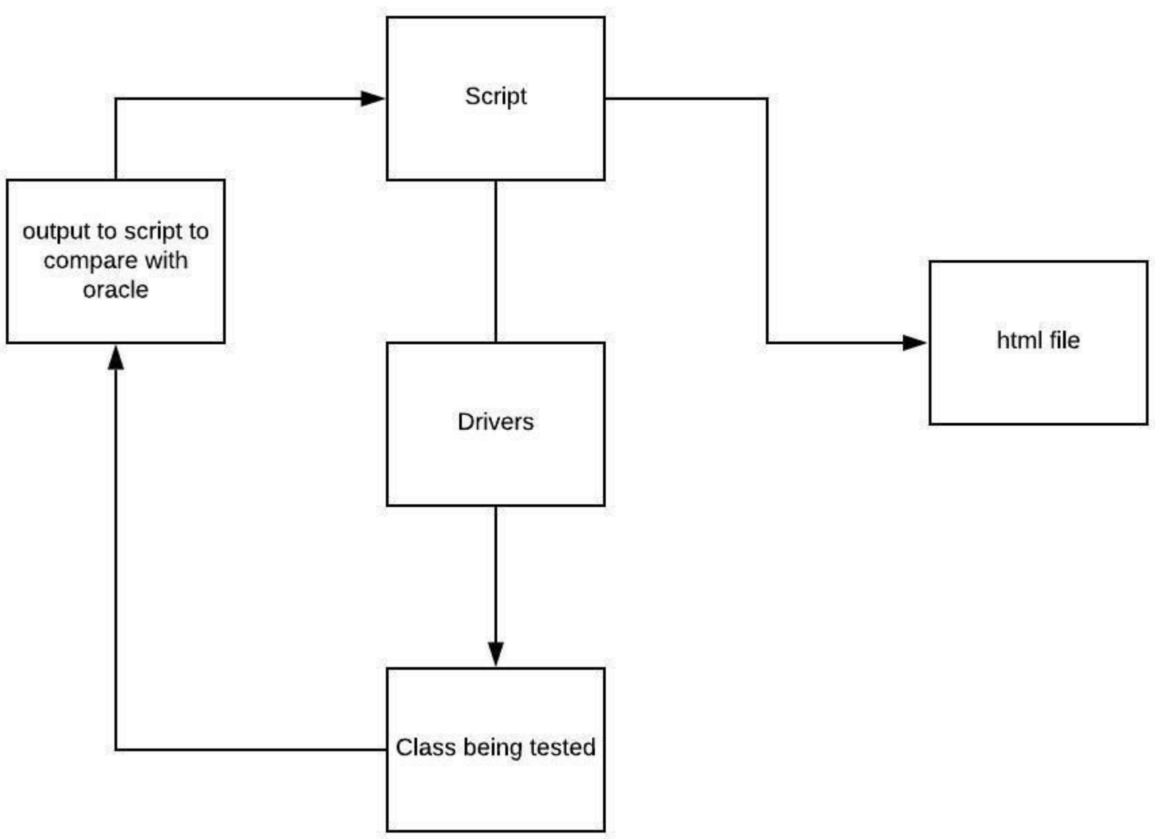
Chapter 3

Report

Since switching projects to OpenMRS things have been going easier. Unfortunately, last class we found out that we have lost a group member which has slowed us down a bit with redistributing his portion of work but now that we are the original Team 2 communication has been easier as well as work ethic. We have the script to run our drivers completed. Our only now issues are the drivers themselves. We are having trouble connecting the script with the drivers in order to produce meaningful results.

Architectural Framework

Our framework consists of a runTest.sh script that controls the flow of our automation and sets up an html file for test reporting. The script then reads in our testCase#.txt files contained within our testing Directory which calls our drivers to perform their tasks (which also read their inputs from the testCase#.txt files) the drivers then output their results to the html file created in the beginning. Finally the script compares the actual output with the oracle to determine if the test Passed or failed.

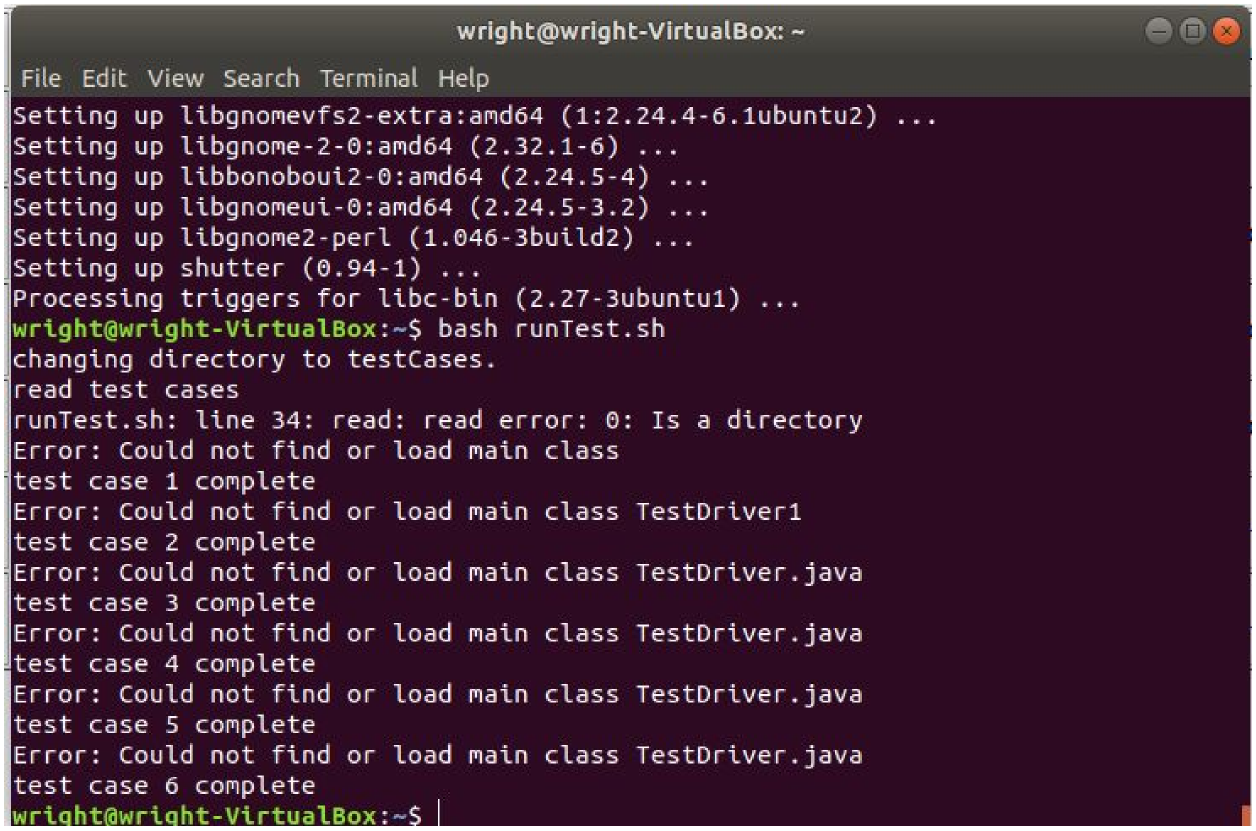


5 Test Cases

- isValid(String identifier) returning true
- isValid(String identifier) returning false
- isValid(String identifier) returning an UnallowedIdentifier exception
- getCheckDigit(String undecoratedIdentifier) converting a string into a check digit using ASCII and formula

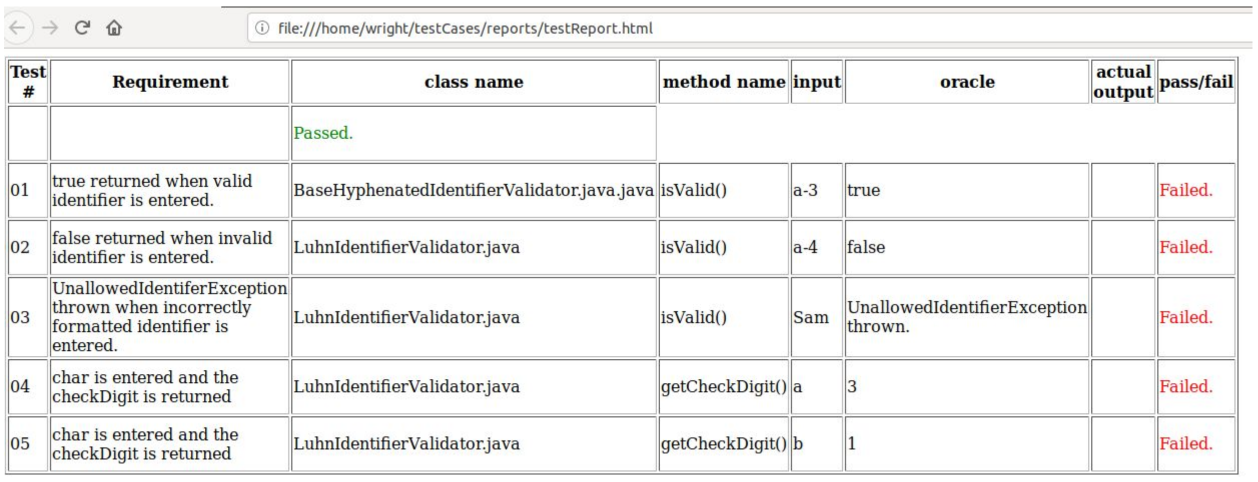
- getCheckDigit(String undecoratedIdentifier) converting a string into a check digit using ASCII and formula

Screenshot of testing report



wright@wright-VirtualBox: ~
File Edit View Search Terminal Help

```
Setting up libgnomevfs2-extra:amd64 (1:2.24.4-6.1ubuntu2) ...
Setting up libgnome-2-0:amd64 (2.32.1-6) ...
Setting up libbonoboui2-0:amd64 (2.24.5-4) ...
Setting up libgnomeui-0:amd64 (2.24.5-3.2) ...
Setting up libgnome2-perl (1.046-3build2) ...
Setting up shutter (0.94-1) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
wright@wright-VirtualBox:~$ bash runTest.sh
changing directory to testCases.
read test cases
runTest.sh: line 34: read: read error: 0: Is a directory
Error: Could not find or load main class
test case 1 complete
Error: Could not find or load main class TestDriver1
test case 2 complete
Error: Could not find or load main class TestDriver.java
test case 3 complete
Error: Could not find or load main class TestDriver.java
test case 4 complete
Error: Could not find or load main class TestDriver.java
test case 5 complete
Error: Could not find or load main class TestDriver.java
test case 6 complete
wright@wright-VirtualBox:~$
```


file:///home/wright/testCases/reports/testReport.html

Test #	Requirement	class name	method name	input	oracle	actual output	pass/fail
		Passed.					
01	true returned when valid identifier is entered.	BaseHyphenatedIdentifierValidator.java.java	isValid()	a-3	true		Failed.
02	false returned when invalid identifier is entered.	LuhnIdentifierValidator.java	isValid()	a-4	false		Failed.
03	UnallowedIdentiferException thrown when incorrectly formatted identifier is entered.	LuhnIdentifierValidator.java	isValid()	Sam	UnallowedIdentifierException thrown.		Failed.
04	char is entered and the checkDigit is returned	LuhnIdentifierValidator.java	getCheckDigit()	a	3		Failed.
05	char is entered and the checkDigit is returned	LuhnIdentifierValidator.java	getCheckDigit()	b	1		Failed.

Chapter 4

Experience

Since deliverable 3 we have made a vast improvement. Back when the last deliverable was due we were unable to get our driver and script to work together. Since then we have abandoned the methods we were attempting to test then, in favor of simpler methods that don't require outside classes and libraries to function. For a while we had a weird error that would return test failed on 3 test cases. We spent hours trying to fix this only to find that for some reason rewriting our test cases word for word character for character worked. I'm not sure why this worked or what caused a comparison of "false" vs "false" or "true" vs. "true" to come back as not equal. Since re-writing the suspect test cases everything has been going smoothly and out Automation Framework has been functioning as intended.

Framework

Our framework runs as such:

1. from the terminal navigate to or open a terminal in the ../2/TestAutomation directory.
2. from this directory we can run our script via the command bash runTest.sh
3. first the script navigates to ../2/TestAutomation/testcase directory, while also setting up our testRepo.html file in ../2/TestAutomation/reports
4. next the script begins filling in the html file with data collected from the testcase files.
5. our driver is then called and reads the test case file being worked on for inputs.
6. the driver then outputs to a testReport.txt file which the script then compares the output vs the oracle in the testcaseXX.txt file to determine if the test passed or failed.
7. the script repeats for as many files contained within the testCase directory.
8. finally once all test cases have been read and their reports added to the html file. the script then opens the html file for viewing.

Testcase template "testCase###.txt"

1. testCase number
2. requirement of what is being tested
3. classname.java
4. the method being tested's name
5. the input to be read by driver/script

6. the expected output (oracle)

Chapter 5

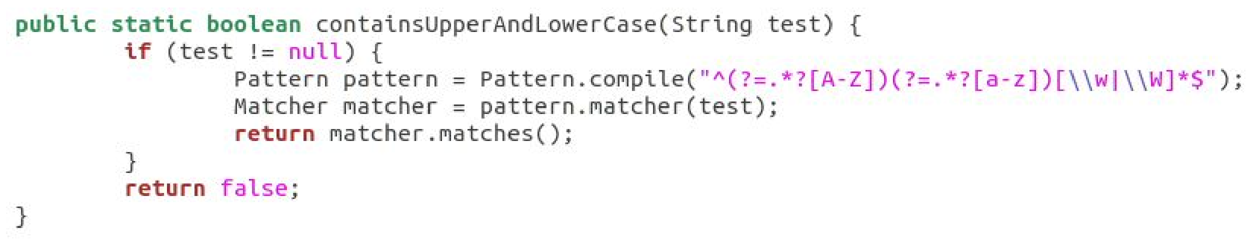
Fault Injection

For deliverable 5 we implemented 5 faults into our code that would most but not all of the corresponding test cases to fail.

---

1. The first fault injected to our code was in the containsUpperAndLowerCase method.

Original code.

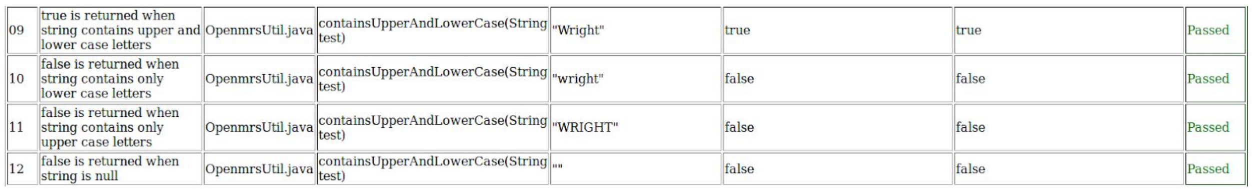


```
public static boolean containsUpperAndLowerCase(String test) {
        if (test != null) {
                Pattern pattern = Pattern.compile("^(?=.*?[A-Z])(?=.*?[a-z])[\\w|\\W]*$");
                Matcher matcher = pattern.matcher(test);
                return matcher.matches();
        }
        return false;
}
```

in this method, we removed "(?=.?[A-Z])" *from Pattern pattern =*

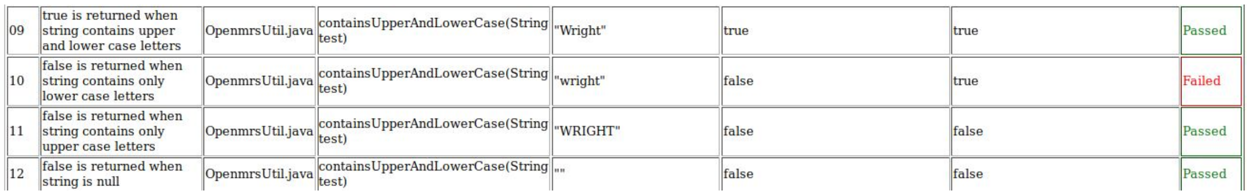
Pattern.compile("ˆ(?=.?[A-Z])(?=.*?[a-z])[\w/\W]*$");

Original results.



09	true is returned when string contains upper and lower case letters	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	"Wright"	true	true	Passed
10	false is returned when string contains only lower case letters	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	"wright"	false	false	Passed
11	false is returned when string contains only upper case letters	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	"WRIGHT"	false	false	Passed
12	false is returned when string is null	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	""	false	false	Passed

Test results after fault injection.



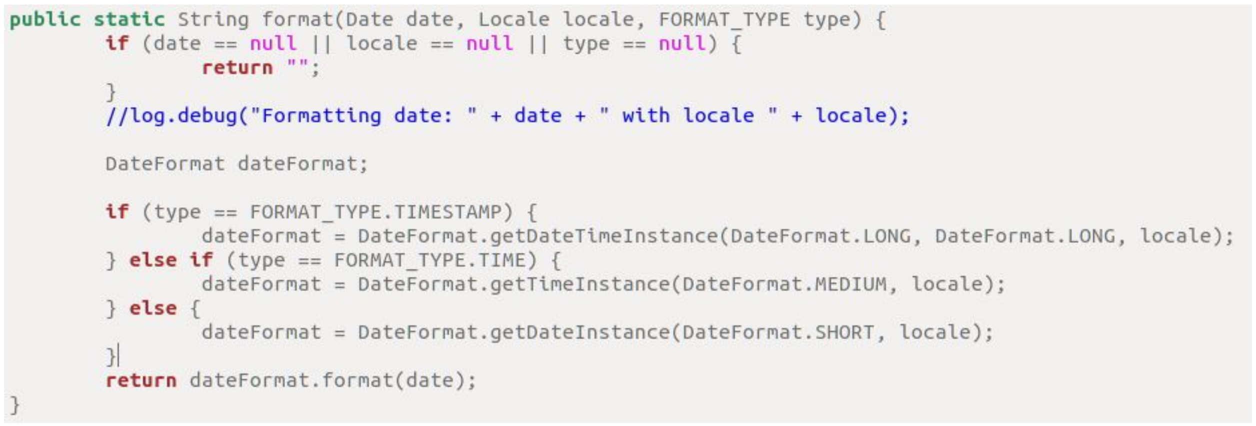
09	true is returned when string contains upper and lower case letters	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	"Wright"	true	true	Passed
10	false is returned when string contains only lower case letters	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	"wright"	false	true	Failed
11	false is returned when string contains only upper case letters	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	"WRIGHT"	false	false	Passed
12	false is returned when string is null	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	""	false	false	Passed

This fault as you can see causes the second test case to fail because the pattern didn't see any uppercase letters.

---

2. The second fault was injected into the format method.

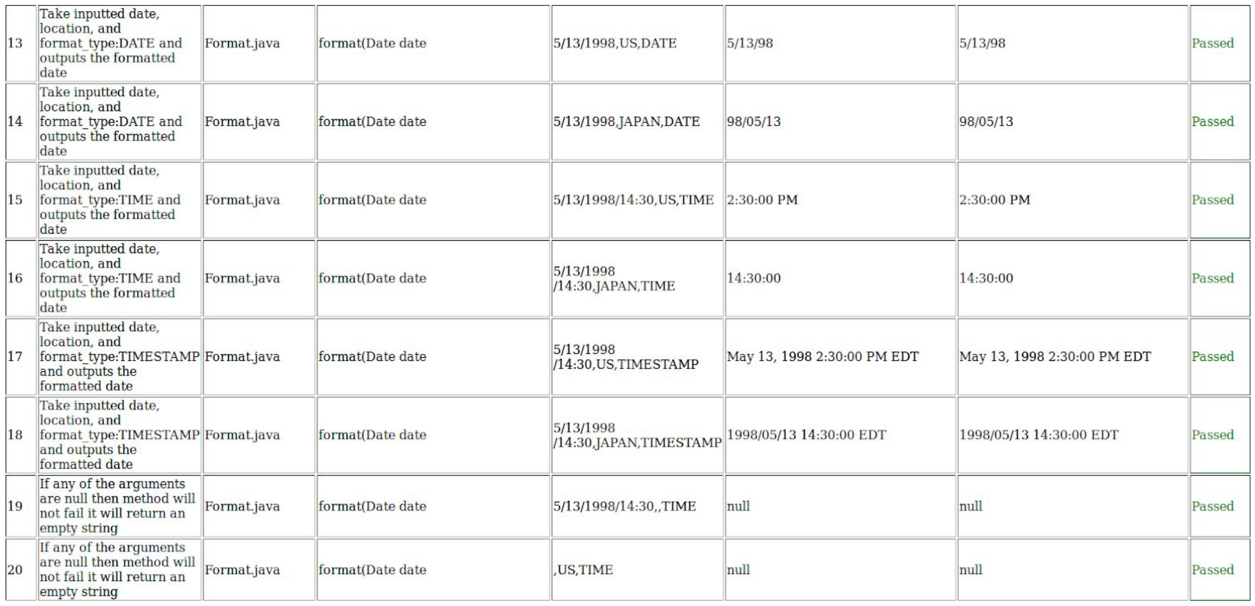
Original Format method



```
public static String format(Date date, Locale locale, FORMAT_TYPE type) {
        if (date == null || locale == null || type == null) {
                return "";
        }
        //log.debug("Formatting date: " + date + " with locale " + locale);

        DateFormat dateFormat;

        if (type == FORMAT_TYPE.TIMESTAMP) {
                dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG, DateFormat.LONG, locale);
        } else if (type == FORMAT_TYPE.TIME) {
                dateFormat = DateFormat.getTimeInstance(DateFormat.MEDIUM, locale);
        } else {
                dateFormat = DateFormat.getDateInstance(DateFormat.SHORT, locale);
        }
        return dateFormat.format(date);
}
```

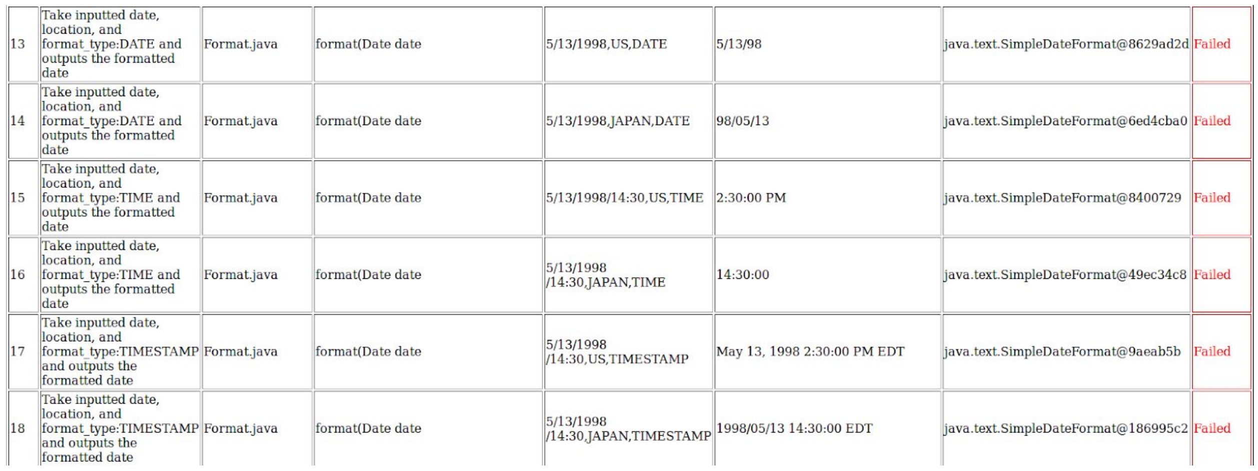
Here instead of returning "dateFormat.format(date);" we return dateFormat.toString(); This causes all format test cases to fail as it now causes a java.text error.

Original results



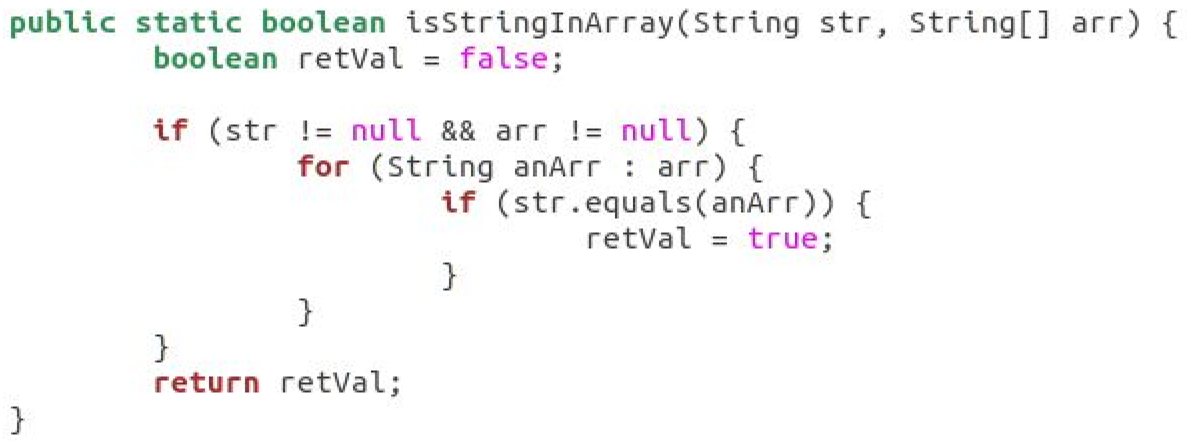
13	Take inputted date, location, and format_type:DATE and outputs the formatted date	Format.java	format(Date date	5/13/1998,US,DATE	5/13/98	5/13/98	Passed
14	Take inputted date, location, and format_type:DATE and outputs the formatted date	Format.java	format(Date date	5/13/1998,JAPAN,DATE	98/05/13	98/05/13	Passed
15	Take inputted date, location, and format_type:TIME and outputs the formatted date	Format.java	format(Date date	5/13/1998/14:30,US,TIME	2:30:00 PM	2:30:00 PM	Passed
16	Take inputted date, location, and format_type:TIME and outputs the formatted date	Format.java	format(Date date	5/13/1998 /14:30,JAPAN,TIME	14:30:00	14:30:00	Passed
17	Take inputted date, location, and format_type:TIMESTAMP and outputs the formatted date	Format.java	format(Date date	5/13/1998 /14:30,US,TIMESTAMP	May 13, 1998 2:30:00 PM EDT	May 13, 1998 2:30:00 PM EDT	Passed
18	Take inputted date, location, and format_type:TIMESTAMP and outputs the formatted date	Format.java	format(Date date	5/13/1998 /14:30,JAPAN,TIMESTAMP	1998/05/13 14:30:00 EDT	1998/05/13 14:30:00 EDT	Passed
19	If any of the arguments are null then method will not fail it will return an empty string	Format.java	format(Date date	5/13/1998/14:30,,TIME	null	null	Passed
20	If any of the arguments are null then method will not fail it will return an empty string	Format.java	format(Date date	,US,TIME	null	null	Passed

Test results after fault injection



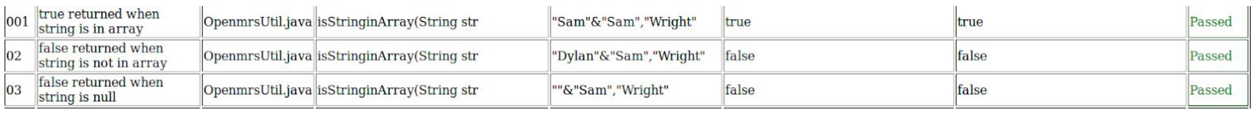
13	Take inputted date, location, and format_type:DATE and outputs the formatted date	Format.java	format(Date date	5/13/1998,US,DATE	5/13/98	java.text.SimpleDateFormat@8629ad2d	Failed
14	Take inputted date, location, and format_type:DATE and outputs the formatted date	Format.java	format(Date date	5/13/1998,JAPAN,DATE	98/05/13	java.text.SimpleDateFormat@6ed4cba0	Failed
15	Take inputted date, location, and format_type:TIME and outputs the formatted date	Format.java	format(Date date	5/13/1998/14:30,US,TIME	2:30:00 PM	java.text.SimpleDateFormat@8400729	Failed
16	Take inputted date, location, and format_type:TIME and outputs the formatted date	Format.java	format(Date date	5/13/1998 /14:30,JAPAN,TIME	14:30:00	java.text.SimpleDateFormat@49ec34c8	Failed
17	Take inputted date, location, and format_type:TIMESTAMP and outputs the formatted date	Format.java	format(Date date	5/13/1998 /14:30,US,TIMESTAMP	May 13, 1998 2:30:00 PM EDT	java.text.SimpleDateFormat@9aeab5b	Failed
18	Take inputted date, location, and format_type:TIMESTAMP and outputs the formatted date	Format.java	format(Date date	5/13/1998 /14:30,JAPAN,TIMESTAMP	1998/05/13 14:30:00 EDT	java.text.SimpleDateFormat@186995c2	Failed

3. With this fault inside the for loop in the method isStringinArray we changed the if statement "(str.equals(anArr))" to (!str.equals(anArr))" Original code



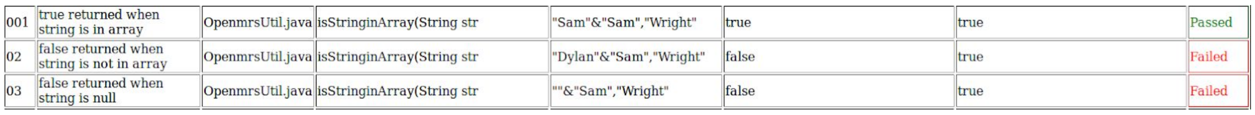
```
public static boolean isStringInArray(String str, String[] arr) {
        boolean retVal = false;

        if (str != null && arr != null) {
                for (String anArr : arr) {
                        if (str.equals(anArr)) {
                                retVal = true;
                        }
                }
        }
        return retVal;
}
```

Original results



001	true returned when string is in array	OpenmrsUtil.java	isStringinArray(String str	"Sam"&"Sam","Wright"	true	true	Passed
02	false returned when string is not in array	OpenmrsUtil.java	isStringinArray(String str	"Dylan"&"Sam","Wright"	false	false	Passed
03	false returned when string is null	OpenmrsUtil.java	isStringinArray(String str	""&"Sam","Wright"	false	false	Passed

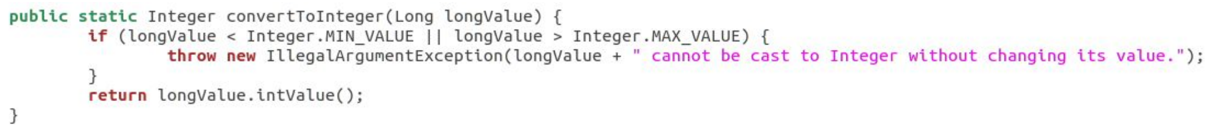
Test results after fault injection



001	true returned when string is in array	OpenmrsUtil.java	isStringinArray(String str	"Sam"&"Sam","Wright"	true	true	Passed
02	false returned when string is not in array	OpenmrsUtil.java	isStringinArray(String str	"Dylan"&"Sam","Wright"	false	true	Failed
03	false returned when string is null	OpenmrsUtil.java	isStringinArray(String str	""&"Sam","Wright"	false	true	Failed

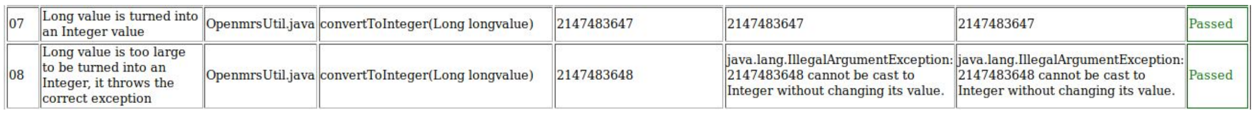
this causes the first test case to pass and the other two for this method to fail.

4. For convertToInteger(Long long) we changed the "||" to "&&" in the if statement Original code



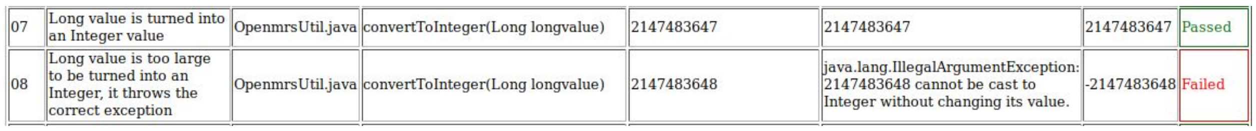
```
public static Integer convertToInteger(Long longValue) {
        if (longValue < Integer.MIN_VALUE || longValue > Integer.MAX_VALUE) {
                throw new IllegalArgumentException(longValue + " cannot be cast to Integer without changing its value.");
        }
        return longValue.intValue();
}
```

Original results



07	Long value is turned into an Integer value	OpenmrsUtil.java	convertToInteger(Long longvalue)	2147483647	2147483647	2147483647	Passed
08	Long value is too large to be turned into an Integer, it throws the correct exception	OpenmrsUtil.java	convertToInteger(Long longvalue)	2147483648	java.lang.IllegalArgumentException: 2147483648 cannot be cast to Integer without changing its value.	java.lang.IllegalArgumentException: 2147483648 cannot be cast to Integer without changing its value.	Passed

Test results after fault injection

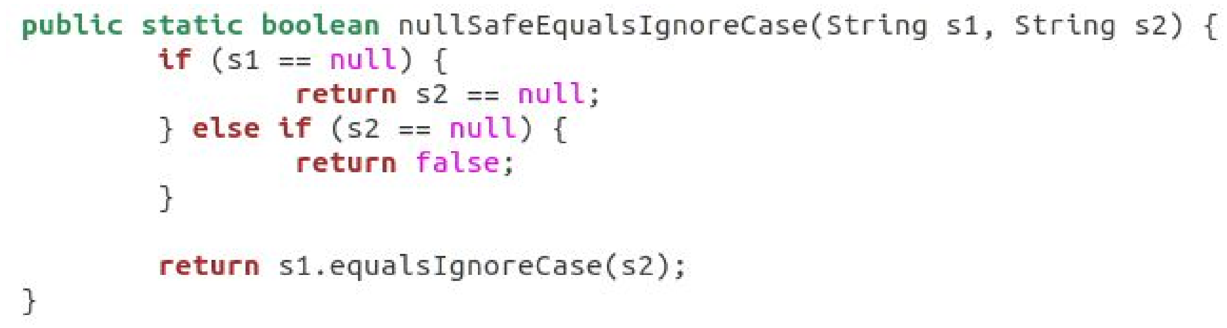


07	Long value is turned into an Integer value	OpenmrsUtil.java	convertToInteger(Long longvalue)	2147483647	2147483647	2147483647	Passed
08	Long value is too large to be turned into an Integer, it throws the correct exception	OpenmrsUtil.java	convertToInteger(Long longvalue)	2147483648	java.lang.IllegalArgumentException: 2147483648 cannot be cast to Integer without changing its value.	-2147483648	Failed

the fault caused the int to become negative because the long was too long to be converted to an int. Since the fault affected the if statement that caught this, it came out as negative instead of outputting and exception.

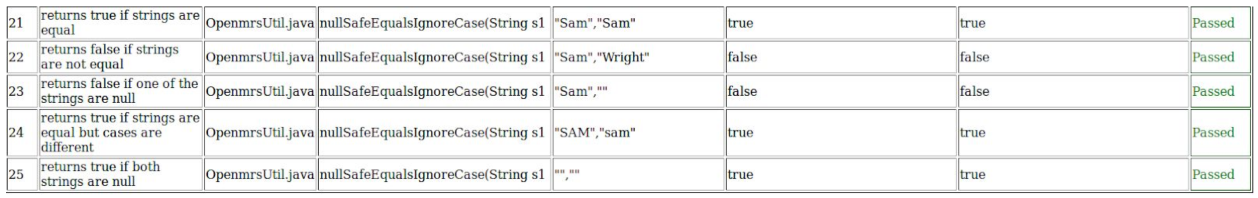
---

5. For our final fault we chose to change the ".equalsIgnoresCase" to just ".equals" in the method nullSafeEqualsIgnoreCase. Original code



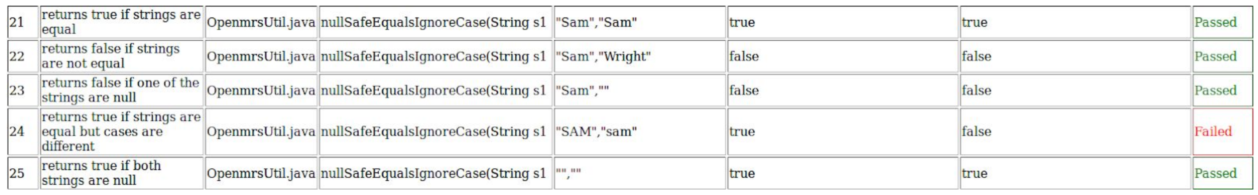
```
public static boolean nullSafeEqualsIgnoreCase(String s1, String s2) {
        if (s1 == null) {
                return s2 == null;
        } else if (s2 == null) {
                return false;
        }

        return s1.equalsIgnoreCase(s2);
}
```

Original results



21	returns true if strings are equal	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"Sam","Sam"	true	true	Passed
22	returns false if strings are not equal	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"Sam","Wright"	false	false	Passed
23	returns false if one of the strings are null	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"Sam",""	false	false	Passed
24	returns true if strings are equal but cases are different	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"SAM","sam"	true	true	Passed
25	returns true if both strings are null	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"",""	true	true	Passed

Test results after fault injection



21	returns true if strings are equal	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"Sam","Sam"	true	true	Passed
22	returns false if strings are not equal	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"Sam","Wright"	false	false	Passed
23	returns false if one of the strings are null	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"Sam",""	false	false	Passed
24	returns true if strings are equal but cases are different	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"SAM","sam"	true	false	Failed
25	returns true if both strings are null	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"",""	true	true	Passed

Chapter 6

Experiences

Many lessons were learned and many new skills were acquired during this project. Our team members had little to no experience with Bash, VirtualBox, Linux, and even Github. The team aspect of this project made the project more challenging. Communication was difficult at times and it was sometimes hard to find a time to meet up and work together on the project. Working with OpenMRS was another big hurdle that I wish we had more time to work on. We started with OpenMRS-core but a lot of classes and methods require imports from the other 5 OpenMRS repositories. Overall, this was a hefty task to complete that was very rewarding in the end.

Team Evaluation

When we picked our teams in class, Ryan was absent that day so the team was originally only Wright and Sam. Since we were the only team of two people, we decided on Team 2 as our name. Next class, Ryan is there and is put into our group. Ryan did leave us on very short notice but we are glad it happened when it did instead of later in the semester when we were in the thick of it. It was fate that Team 2 would be returned back to its original form that it was named after.

In the early stages (deliverable 1 and 2), we could have met more frequently and been more organized on github. We think that would have made this project go a lot more smoothly. After losing Ryan, we had to redistribute the work between us and figure out a plan. This departure from the group made us realize we had a lot of work to do and that we had to get it going. Communication became much easier between us and we met up more frequently. We have worked hard to get every deliverable in on time.

Evaluation of Project Assignment

This project was difficult. When we first started no one on our team members had any extensive experience in bash, github, or linux. However, from working on this project, as a team, we feel we are much better equipped to handle tasks of this size going forward. This project has also given us a good idea of what it takes to be a cohesive team, and the importance of solid communication.