OpenMRS Test Automation Framework

Cameron Dey,

Evan Tanner,

Brandon Priester

Table of Contents:

Introduction:

For CSCI 362, we had to pick an open source project and build a testing framework for components of its API. Our team decided to choose OpenMRS as our open source project. OpenMRS is, “a patient-based medical record system focusing on giving providers a free customizable electronic medical record system” [1].The mission of OpenMRS, “is to improve health care delivery in resource-constrained environments by coordinating a global community to create and support this software”[2]. Through several iterations, or deliverables, we progressed through many challenges of creating a sophisticated testing framework for methods of the API. Seen in the deliverables below are the various iterations our team made to create the final testing framework.

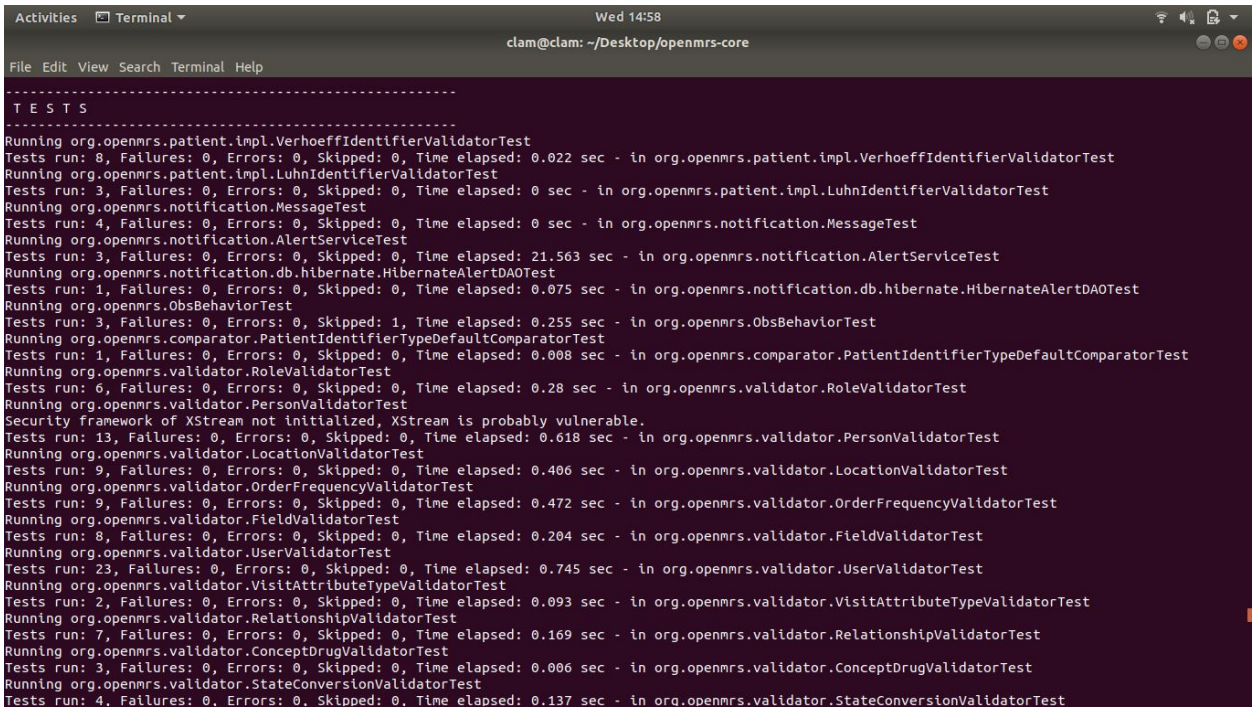
Specifications:

http://www.cs.cofc.edu/~bowring/classes/csci%20362/fall%202019/teamProject/TeamProjectsSpecifications.pdf

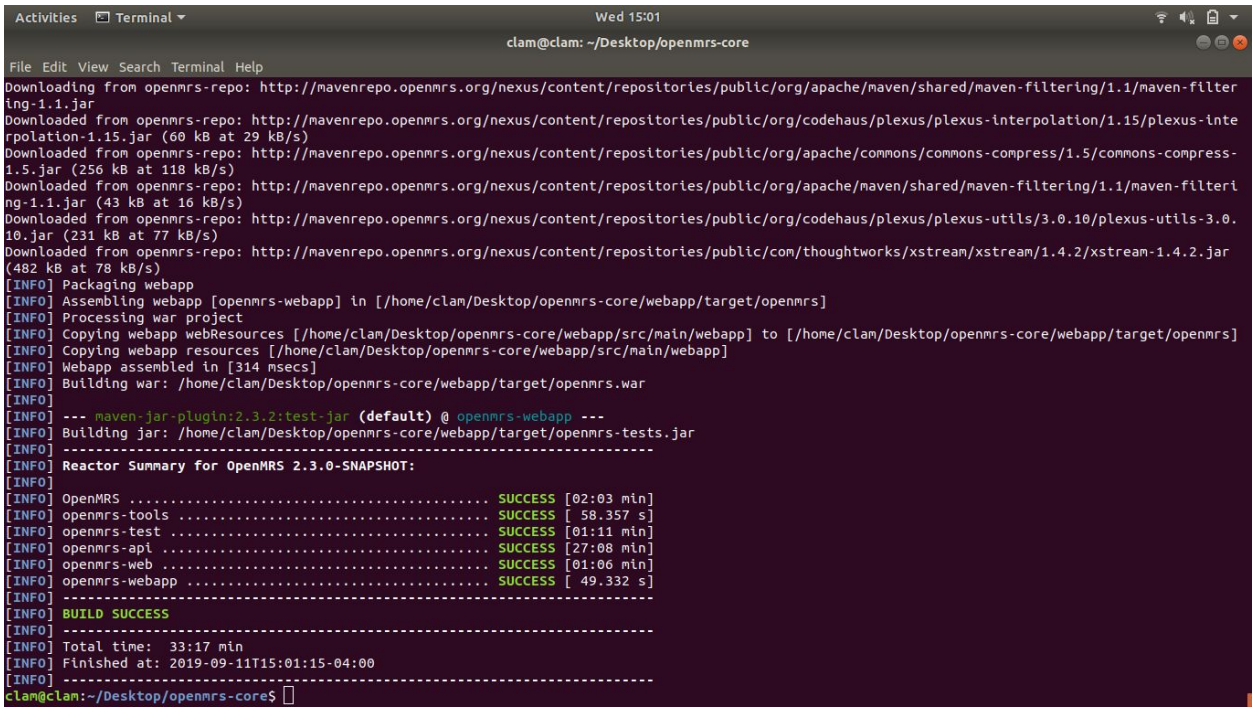
Team4: Deliverable 1

To begin, we cloned OpenMRS into our local repos. We then installed the prerequisite dependencies for the programs (Java 1.8 and Maven). We then used Maven to compile the openers-core package. During compilation, an astronomical amount of tests were ran. Some of the tests that ran included webUtilTest, moduleFileParserUnitTest, and PatientDaoTest. Our next team goal will be to create a database and implement future tests.

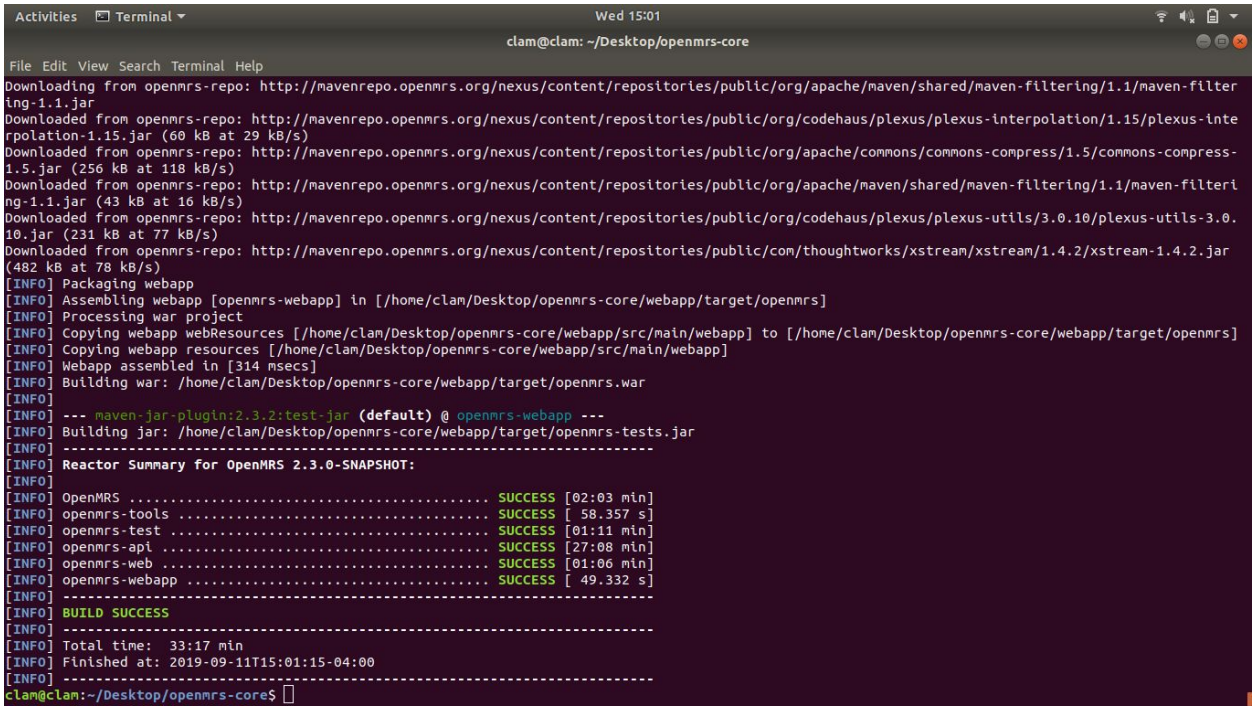
Tests Screenshot:



Finished Compilation Screenshot :



Installation Screenshot :



Team4: Deliverable 2

Team4 Test Plan

Scope

- We are developing only 25 test cases in a very small portion of the life cycle of OpenMRS

Approach

- Unit Testing
    - Unit Ranges (min, max)
    - I/O exceptions
    - Unit mismatch exceptions
- Component Testing
    - Testing integration of classes (IE, User class has a Person class)

Resources

- Drivers
- Github
- Software Dependencies
    - Java 8
    - Maven (any version)

Schedule

- Deliverable 1: September 12
    - Select, Clone, and Build project
    - Update Wiki
- Deliverable 2: October 1
    - Create a test plan
    - Identify 5 of the eventual 25 test cases we will develop
    - Update wiki
- Deliverable 3: October 29
    - Rework test plan if necessary
    - Design and build a testing framework
    - Implement our test plan
    - Update wiki
- Deliverable 4: November 12
    - Complete the design/implementation of our testing framework
    - Create the rest of the 25 total test cases
    - The testing framework should automatically use all 25 test cases
    - Update wiki
- Deliverable 5: November 19
    - Inject 5 faults into the code
    - Test the faults for failures
    - Record feedback
    - Update Wiki

Testing Tasks

- Identify bugs and errors

Degree of Tester Independence

- Team4 is not affiliated with any developers or associated organizations of OpenMRS. We are entirely independent of all associated parties to OpenMRS.

Test Environment

- Ubuntu 18.04LTS
- Java 8

Entry/Exit Criteria

- If the system catches the exceptions/errors it is a success.

Constraints:

- Team of three people
- Time (Other Classes/Work/Etc.)

References

- OpenMRS

Assumptions and Dependencies

- Not all methods will be tested
- The entire system will not be factored
- Component Tests will not be conducted and are expected to already function

---

Test Items

Test Case Template

ID #

Component

Method

input

expected output

---

1. **Class: OpenmrsUtil.java**

1

OpenmrsUtil.java

convertToInteger(Long longValue)

0

0

2

OpenmrsUtil.java

convertToInteger(Long longValue)

2147483647

2147483647

3

OpenmrsUtil.java

convertToInteger(Long longValue)

2147483648

2147483648 cannot be cast to Integer without changing its value.

4

OpenmrsUtil.java

convertToInteger(Long longValue)

-2147483648

-2147483648

5

OpenmrsUtil.java

convertToInteger(Long longValue)

-2147483649

-2147483649 cannot be cast to Integer without changing its value.

6

OpenmrsUtil.java

convertToInteger(Long longValue)

-Sup

Number Format Exception

Other Potential Methods to Test

Location.java

Boolean isTag(String tagToFind)

OpenmrsUtil.Java

boolean isStringInArray(String str, String[] arr)

boolean containsOnlyDigits(String test)

boolean containsDigit(String test)

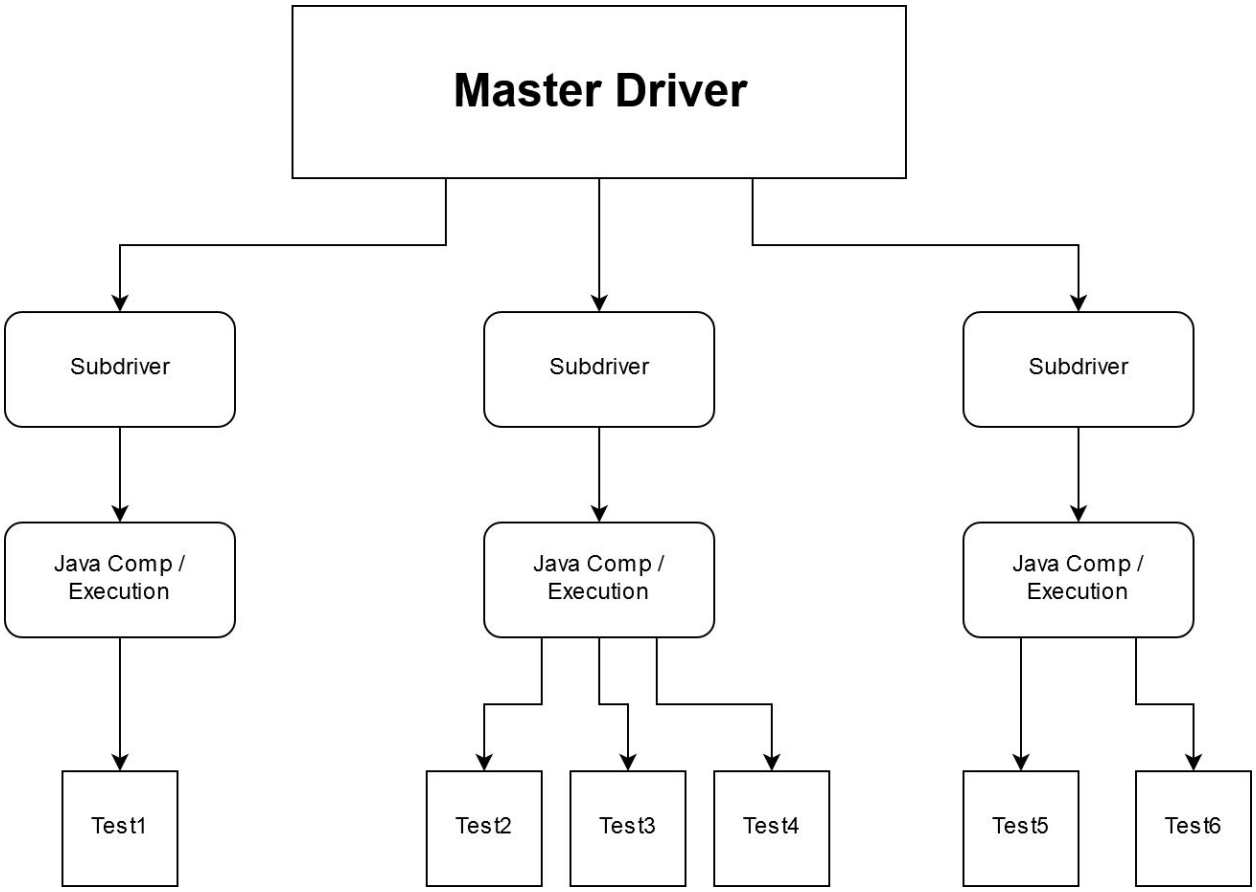
boolean nullSafeEqualsIgnoreCase(String s1, String s2)

Integer convertToInteger(Long longValue)

Team4: Deliverable 3

Architectural Description

This is a testing framework that is composed of a master driver that operates from the scripts folder which executes the sub-drivers. The sub-drivers are responsible for compiling the java code and executing them with their respective input that is held within the testCases folder.



Full How-To Description

Step 0: Make sure you are using Ubuntu 18. Install Java 8 if you have not already and set it as your primary Java version.

Step 1: Clone the Team4 github repository onto your machine.

Step 2: Navigate into the Team4 repository and open the TestAutomation directory on the shell.

Step 3: From the TestAutomation directory, type: ./scripts/runAllTests.sh

Step 4: An html report should open automatically with the full documentation for each test that was run, including the results.

Our Test Cases So Far

TestNumber: 1

Class: OpenmrsUtil.java

Method: convertToInteger(Long longValue)

Input: 0

Oracle: 0

TestNumber: 2

Class: OpenmrsUtil.java

Method: convertToInteger(Long longValue)

Input: 2147483647

Oracle: 2147483647

TestNumber: 3

Class: OpenmrsUtil.java

Method: convertToInteger(Long longValue)

Input: 2147483648

Oracle: 2147483648 cannot be cast to Integer without changing its value.

TestNumber: 4

Class: OpenmrsUtil.java

Method: convertToInteger(Long longValue)

Input: -2147483648

Oracle: -2147483648

TestNumber: 5

Class: OpenmrsUtil.java

Method: convertToInteger(Long longValue)

Input: -2147483649

Oracle: -2147483649 cannot be cast to Integer without changing its value.

TestNumber: 6

Method: OpenmrsUtil.java

Class: convertToInteger(Long longValue)

Input: -Sup

Oracle: Number Format Exception

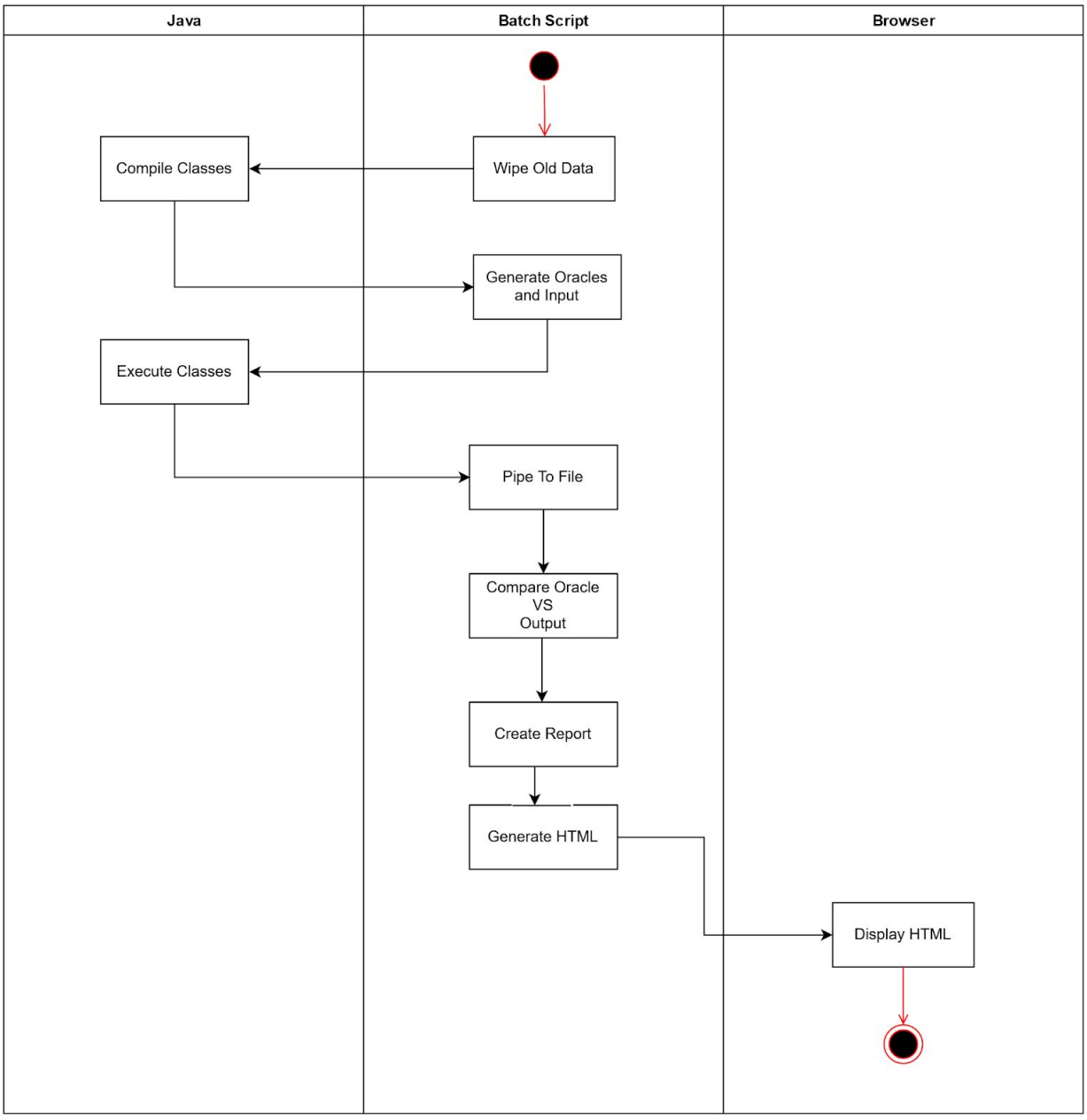
Team4: Deliverable 4

This is a testing framework that is composed of a git bash script (runAllTests.sh) and 6 java drivers. The java drivers access the desired methods to be tested from OpenMRS classes specified in the test cases and tests them given the specified input in the test cases. The bash script runs through four basic steps:

1. Compile all Java driver files
2. Read in each test case and run them using Java drivers
3. Compare oracles to driver outputs, form final report txt
4. Produce/open finalReport html

After running through each of these steps, a final report html should open in an internet browser.

Diagram of the updated architectural framework:



Full How-To Description

Step 0: Make sure you are using Ubuntu 18. Install Java 8 if you have not already and set it as your primary Java version.

Step 1: Clone the Team4 github repository onto your machine.

Step 2: Navigate into the Team4 repository and open the TestAutomation directory on the shell.

Step 3: From the TestAutomation directory, type: ./scripts/runAllTests.sh

> Step 3.5: Choose whether to use the faulty jar or the standard

Step 4: An html report should open automatically with the full documentation for each test that was run, including the results.

Test Case Progress

Test:1

Class: OpenmrsUtil.java

Method: convertToInteger(Long longValue)

Requirement: Returns an Integer from a long.

Input: 0

Output:0

Test:2

Class: OpenmrsUtil.java

Method: convertToInteger(Long longValue)

Requirement: Returns an Integer from a long.

Input: 2147483647

Output: 2147483647

Test:3

Class: OpenmrsUtil.java

Method: convertToInteger(Long longValue)

Requirement: Returns an Integer from a Long

Input: 2147483648

Output: 2147483648 cannot be cast to Integer without changing its value.

Test:4

Class: OpenmrsUtil.java

Method: convertToInteger(Long longValue)

Requirement: Returns an Integer from a Long

Input: -2147483648

Output: -2147483648

Test:5

Class:OpenmrsUtil.java

Method: convertToInteger(Long longValue)

Requirement: Returns an Integer from a Long

Input: -2147483649

Output: -2147483649 cannot be cast to Integer without changing its value.

Test:6

Class: OpenmrsUtil.java

Method: containsOnlyDigits(String test)

Requirement: Returns true if String contains only digits

Input: 123451612454654646065056 4

Output: true

Test:7

Class: OpenmrsUtil.java

Method: containsOnlyDigits(String test)

Requirement: Returns true if String contains only digits

Input: a1012

Output: false

Test:8

Class: OpenmrsUtil.java

Method: containsOnlyDigits(String test)

Requirement: Returns true if String contains only digits

Input:

Output: false

Test:9

Class: OpenmrsUtil.java

Method: containsOnlyDigits(String test)

Requirement: Returns true if String contains only digits

Input: -

Output: false

Test:10

Class: OpenmrsUtil.java

Method: containsOnlyDigits(String test)

Requirement: Returns true if String contains only digits

Input: 12345?

Output:false

Test:11

Class: OpenmrsUtil.java

Method: containsOnlyDigits(String test)

Requirement:Returns true if String contains only digits

Input: 0

Output: true

Test:12

Class:OpenmrsUtil.java

Method:containsDigit(String test)

Requirement:Returns true if String contains any digits

Input: Pizza

Output: false

Test:13

Class: OpenmrsUtil.java

Method: containsDigit(String test)

Requirement: Returns true if String contains any digits

Input: 1432

Output: true

Test:14

Class: OpenmrsUtil.java

Method: containsDigit(String test)

Requirement: Returns true if String contains any digits

Input: test3

Output: true

Test:15

Class: OpenmrsUtil.java

Method: containsDigit(String test)

Requirement:Returns true if String contains any digits

Input: .!@#$!$%&*&%()~?

Output: false

Test:16

Class: OpenmrsUtil.java

Method: containsUpperAndLowerCase(String test)

Requirement: Returns true if contains upper and lower case letters

Input:

Output: false

Test:17

Class:OpenmrsUtil.java

Method: containsUpperAndLowerCase(String test)

Requirement: Returns true if contains upper and lower case letters

Input: dance gavin dance

Output:false

Test:18

Class: OpenmrsUtil.java

Method: containsUpperAndLowerCase(String test)

Requirement: Returns true if contains upper and lower case letters

Input: PourMelikeAPotOfCoffeeKindOfSlowAndKindaPoppy

Output: true

Test:19

Class: OpenmrsUtil.java

Method: containsUpperAndLowerCase(String test)

Requirement: Returns true if contains upper and lower case letters

Input: !!!Pour Me Like A Pot of Coffee&&*

Output: true

Test:20

Class: OpenmrsUtil.java

Method: containsUpperAndLowerCase(String test)

Requirement: Returns true if contains upper and lower case letters

Input: TOTA

Output: false

Test:21

Class: OpenmrsUtil.java

Method: isStringInArray(String str, String[] arr)

Requirement: Returns true if a String is included in a String array.

Input: Pizza Taco

Output: false

Test:22

Class: OpenmrsUtil.java

Method: isStringInArray(String str, String[] arr)

Requirement: Returns true if a String is included in a String array.

Input: Pizza Taco Pizza

Output: true

Test:23

Class: OpenmrsUtil.java

Method: isStringInArray(String str, String[] arr)

Requirement: Returns true if a String is included in a String array.

Input: Pizza

Output: false

Test:24

Class: OpenmrsUtil.java

Method: isStringInArray(String str, String[] arr)

Requirement: Returns true if a String is included in a String array.

Input: - eggs taco pizza 123 42 - cheese

Output: true

Test:25

Class: OpenmrsUtil.java

Method: isStringInArray(String str, String[] arr)

Requirement: Returns true if a String is included in a String array.

Input: input eggs taco pizza 123 42 cheese

Output: false

Final Test Report

Test Report

Today's Date: Wed Nov 20 2019 19:58:42 GMT-0500 (Eastern Standard Time)

Test #	Class	Method	Requirement	Input	Output	Oracle	Result
1	OpenmrsUtil.java	convertToInteger(Long longValue)	Returns an Integer from a Long (tests assume you're on a 32-bit system)	0	0	0	Pass
2	OpenmrsUtil.java	convertToInteger(Long longValue)	Returns an Integer from a Long (tests assume you're on a 32-bit system)	2147483647	2147483647	2147483647	Pass
3	OpenmrsUtil.java	convertToInteger(Long longValue)	Returns an Integer from a Long (tests assume you're on a 32-bit system)	2147483648	2147483648 cannot be cast to Integer without changing its value.	2147483648 cannot be cast to Integer without changing its value.	Pass
4	OpenmrsUtil.java	convertToInteger(Long longValue)	Returns an Integer from a Long (tests assume you're on a 32-bit system)	-2147483648	-2147483648	-2147483648	Pass
5	OpenmrsUtil.java	convertToInteger(Long longValue)	Returns an Integer from a Long (tests assume you're on a 32-bit system)	-2147483649	-2147483649 cannot be cast to Integer without changing its value.	-2147483649 cannot be cast to Integer without changing its value.	Pass
6	OpenmrsUtil.java	containsOnlyDigits(String test)	Returns true if String contains only digits	12345161245465646060650564	true	true	Pass
7	OpenmrsUtil.java	containsOnlyDigits(String test)	Returns true if String contains only digits	a1012	false	false	Pass
8	OpenmrsUtil.java	containsOnlyDigits(String test)	Returns true if String contains only digits		false	false	Pass
9	OpenmrsUtil.java	containsOnlyDigits(String test)	Returns true if String contains only digits	.	false	false	Pass
10	OpenmrsUtil.java	containsOnlyDigits(String test)	Returns true if String contains only digits	12345?	false	false	Pass
11	OpenmrsUtil.java	containsOnlyDigits(String test)	Returns true if String contains only digits	0	true	true	Pass
12	OpenmrsUtil.java	containsDigit(String test)	Returns true if String contains any digits	Pizza	false	false	Pass
13	OpenmrsUtil.java	containsDigit(String test)	Returns true if String contains any digits	1432	true	true	Pass
14	OpenmrsUtil.java	containsDigit(String test)	Returns true if String contains any digits	test3	true	true	Pass
15	OpenmrsUtil.java	containsDigit(String test)	Returns true if String contains any digits	.!@#$!$%&*&%()~?	false	false	Pass
16	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	Returns true if contains upper and lower case letters		false	false	Pass
17	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	Returns true if contains upper and lower case letters	dAnce gAvin dAnce	true	true	Pass
18	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	Returns true if contains upper and lower case letters	PourMelikeAPotOfCoffeeKindOfSlowAndKindaPoppy	true	true	Pass
19	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	Returns true if contains upper and lower case letters	!!!Pour Me Like A Pot of Coffee&&*	true	true	Pass
20	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	Returns true if contains upper and lower case letters	TOTa	true	true	Pass
21	OpenmrsUtil.java	isStringInArray(String str, String[] arr)	Returns true if a String is included in a String array.	Pizza Taco	false	false	Pass
22	OpenmrsUtil.java	isStringInArray(String str, String[] arr)	Returns true if a String is included in a String array.	Pizza Taco Pizza	true	true	Pass
23	OpenmrsUtil.java	isStringInArray(String str, String[] arr)	Returns true if a String is included in a String array.	Pizza	false	false	Pass
24	OpenmrsUtil.java	isStringInArray(String str, String[] arr)	Returns true if a String is included in a String array.	- eggs taco pizza 123 42 - cheese	true	true	Pass
25	OpenmrsUtil.java	isStringInArray(String str, String[] arr)	Returns true if a String is included in a String array.	input eggs taco pizza 123 42 cheese	false	false	Pass
26	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	Returns true if contains upper and lower case letters	aAbBcC	true	true	Pass

Team4: Deliverable 5

Injected Faults:

All faults are injected into the OpenmrsUtil.java

Fault 1: convertToInteger

No longer parses the value and immediately converts.

Fault 2: containsOnlyDigits

No longer checks for null strings

Fault 3: containsDigit

No longer checks for null strings

Fault 4: containsUpperAndLowerCase

No longer considers the letters a,b,c,A,B,C as valid letters

Fault 5: isStringInArray

No longer checks for null arrays or strings

Assignment Evaluations

Deliverable 1:

The building of OpenMrs first began with the cloning of its Github repository. The required dependencies included Java 1.8 and Maven. Using Maven, we compiled the Openmrs-core package that then automatically ran various pre-configured tests. While this deliverable was not very difficult, the download seemed impeded towards certain operating systems that our team members were originally inclined to use. For example, Ubuntu 16.04 had trouble running the repository even after downloading the required dependencies. To fix this problem, every team member decided to use Ubuntu 18.04 for the rest of the iterations to follow.

Deliverable 2:

The goal of deliverable 2 was to create a detailed test plan for our OpenMrs testing framework and then plan at least 5 of the eventual 25 test cases. Our test plan included the scope, approach, resources, schedule, testing tasks, test environments, entry/exit criteria, constraints, references, and assumptions/dependencies. The difficulty in this task stemmed from having to predict exactly how we should undertake the challenge of creating a sophisticated framework. With limited knowledge of OpenMrs's api, we had to make assumptions of how the framework should be abstracted and then implemented. After creating the test plan, we decided to pick convertToInteger as our first method for planning a test case.

Deliverable 3:

The goal of deliverable 3 was to design and build an automated testing framework that used our previous built test plan from deliverable 2. This deliverable was the first time we had to create a jar in order to use and pull from the openmrs-core api. While this may have seemed to be an easy task at first, finding the right method to accomplish this deemed to be the opposite. With some of the material in OpenMrs's documents suggesting the need for a server to interact with the API, it almost led us through an unnecessary route in accessing the API. Luckily, the jar was built through opening the module within the OpenMRS core; where it had to be accessed. Trying to build the API using the core wasn't possible without including all other modules. Further confusion was found on how to use the jar as it had to be built as a plugin which was a headache at that moment. But, after following a guide on how to create a plugin that issue was resolved. With much fortune, a way to compile the code was found after, however we did not find a way to compile it without being within the directory; Using relative paths threw errors, but being able to compile was more than enough for us.

Deliverable 4:

The goal of deliverable 4 was to complete the design and implementation of the testing framework our team specified in deliverable 3. In addition to this, we had to finish implementing the rest of the 25 total test cases. For this deliverable we had to complete the remaining sub-drivers in our program to execute the last 4 java methods whose output would be tested against their oracles. With the script already being designed to access and execute based on the line of each text file, we thought it would be a relatively easy deliverable. But in hindsight, we had to switch between potential methods from the previous deliverables to see which ones would work the best for our testing framework. It should also be noted here that the architecture of the framework was simplified in design during this phase. We removed the relationship of a master and a sub driver from the script. The script was then made into a standalone script which dealt with the large task. In essence, the modularity of the script was removed to simplify the process and to improve the efficiency.

Deliverable 5:

The goal of deliverable 5 was to design and inject 5 faults into OpenMrs's API to cause at least 5 of the test cases to fail. While changing the Java code to fail wasn't too difficult, the process of creating individual jars between periods of testing each unique mutant method became the actual headache. Not only this, but methods such as, StringInArray, would originally test for null in previous deliverables by checking for a carriage return. With fault injections, this method became very hard to test by not allowing a null to be passed through its parameter along with other strings. While all the Java code in the jar was implemented to fail as seen in deliverable 5 wiki, we were forced to use only convertToInteger and containsUpperAndLowerCase in order to obtain the 5 test case fails demonstrated in deliverable 5. It should also be noted that during this deliverable that a small appendage was made to the runAllTests script such that it is possible to run with or without the faulted jar.

Team Evaluations

Throughout this semester our team has overcome many challenges that software engineers commonly face when developing an api-oriented program. To begin, the use of version control proved to be an exceptionally useful tool for working with each other remotely. While we've used Github before, the constant need to push and pull has been great practice for our ability to develop as a group. While using github frequently, we learned about the various exceptions that arise when a local repository is mismatched with the master branch.

In addition to our gained experience with GitHub, we also obtained a lot of practice working with git bash scripting (something that none of us had a lot of experience with at the start of this project). Since we all had similar levels of experience with each tool involved in this project, we had a good opportunity to grow together as a team. Each new breakthrough in our project was a breakthrough for everyone, and this boosted our team's morale.

The first three deliverables we completed required many meetings outside of class since we were still figuring out how to effectively use each tool involved. The last two deliverables, however, were completed way more efficiently. We not only had a much greater understanding of how our testing framework worked but we also knew how to work together as a team very well.