



OpenMRS
MEDICAL RECORD SYSTEM

Team 7 Final Report

College of Charleston

CSCI 362 Fall 2020

Dr. Bowring

Team Members: Jayton Fee, Violet Smith, Thomas Marshall

Automated Framework for testing OpenMRS, an open-source
project

TABLE OF CONTENTS:

0	Introduction - - - - -	3
1	Deliverable 1	
1.1	Failed Attempts - - - - -	4
1.2	Potential Success - - - - -	6
2	Deliverable 2	
2.1	Update - - - - -	8
2.2	Initial Test Cases - - - - -	8
2.3	Projected Test Case Schedule - - - - -	12
2.4	Requirements and Constraints - - - - -	13
3	Deliverable 3	
3.1	Update - - - - -	13
3.2	Test Architecture Outline - - - - -	13
3.3	Initial Test Cases - - - - -	14
3.4	Framework Instructions - - - - -	14
4	Deliverable 4	
4.1	Update - - - - -	15
4.2	The 25 Test Cases - - - - -	15
5	Deliverable 5	
5.1	Fault Injections - - - - -	18
6	Final Results	
6.1	Updates - - - - -	23
6.2	Testing Framework - - - - -	23
6.3	Requirements - - - - -	23
6.4	Instructions - - - - -	24
7	Reflections	
7.1	Self-Reflections - - - - -	25
7.2	Assignment Reflections and Critiques - - - - -	25

0. Introduction:

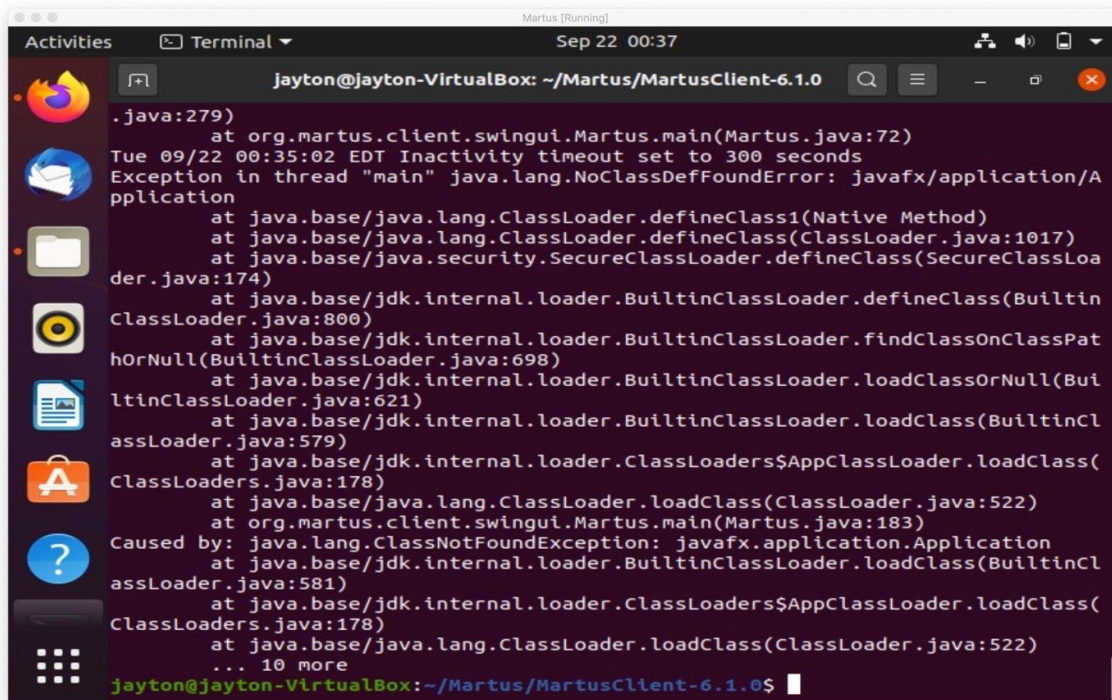
Our Team decided to work with the HFOSS project OpenMRS. OpenMRS is an open source electronic medical system. Their mission is to improve health care in limited resource environments by coordinating a global community to support and create this software. Please keep in mind, information found within deliverables were all documented at the time we were at each stage. Some or all of the information recorded for each deliverable has been changed in substantial ways. If you plan on using our format, only use the final results section for an accurate reference.

1. Deliverable 1:

1.1 Failed Attempts

Martus attempt:

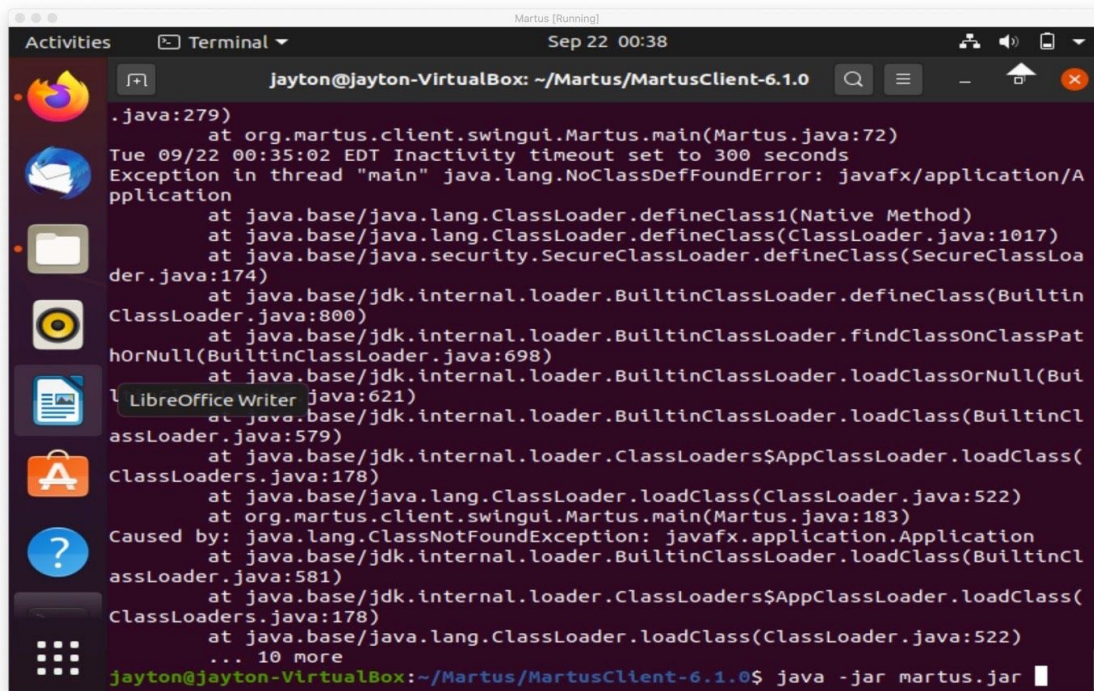
After installing and unzipping the Martus project our team attempted to run Martus on an Ubuntu Virtualbox. After installation of all the required software we attempted to run Martus and were not successful. We carefully browsed all supporting documentation from Marus without much help being provided. The attached screenshots document the error messages received from the terminal.



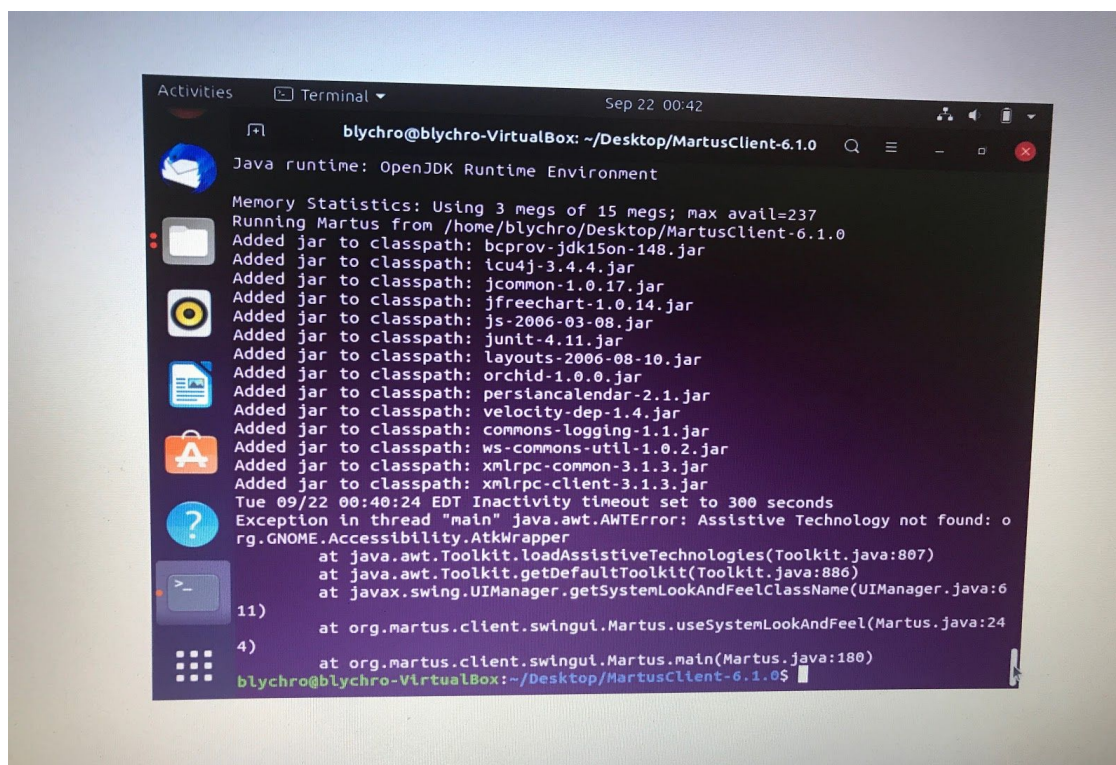
The screenshot shows a terminal window titled "Martus [Running]" with the command prompt "jayton@jayton-VirtualBox: ~/Martus/MartusClient-6.1.0". The terminal output displays a Java exception stack trace. The error is a "java.lang.NoClassDefFoundError: javafx/application/Application" occurring in the "main" thread. The stack trace includes the following frames (from top to bottom):

- at org.martus.client.swingui.Martus.main(Martus.java:72)
- Tue 09/22 00:35:02 EDT Inactivity timeout set to 300 seconds
- Exception in thread "main" java.lang.NoClassDefFoundError: javafx/application/Application
- at java.base/java.lang.ClassLoader.defineClass1(Native Method)
- at java.base/java.lang.ClassLoader.defineClass(ClassLoader.java:1017)
- at java.base/java.security.SecureClassLoader.defineClass(SecureClassLoader.java:174)
- at java.base/jdk.internal.loader.BuiltinClassLoader.defineClass(BuiltinClassLoader.java:800)
- at java.base/jdk.internal.loader.BuiltinClassLoader.findClassOnClassPathOrNull(BuiltinClassLoader.java:698)
- at java.base/jdk.internal.loader.BuiltinClassLoader.loadClassOrNull(BuiltinClassLoader.java:621)
- at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClassLoader.java:579)
- at java.base/jdk.internal.loader.ClassLoaders\$AppClassLoader.loadClass(ClassLoaders.java:178)
- at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:522)
- at org.martus.client.swingui.Martus.main(Martus.java:183)
- Caused by: java.lang.ClassNotFoundException: javafx.application.Application
- at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClassLoader.java:581)
- at java.base/jdk.internal.loader.ClassLoaders\$AppClassLoader.loadClass(ClassLoaders.java:178)
- at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:522)
- ... 10 more

The terminal prompt is "jayton@jayton-VirtualBox:~/Martus/MartusClient-6.1.0\$".



```
.java:279)
    at org.martus.client.swingui.Martus.main(Martus.java:72)
Tue 09/22 00:35:02 EDT Inactivity timeout set to 300 seconds
Exception in thread "main" java.lang.NoClassDefFoundError: javafx/application/A
pplication
    at java.base/java.lang.ClassLoader.defineClass1(Native Method)
    at java.base/java.lang.ClassLoader.defineClass(ClassLoader.java:1017)
    at java.base/java.security.SecureClassLoader.defineClass(SecureClassLoa
der.java:174)
    at java.base/jdk.internal.loader.BuiltinClassLoader.defineClass(Builtin
ClassLoader.java:800)
    at java.base/jdk.internal.loader.BuiltinClassLoader.findClassOnClassPat
hOrNull(BuiltinClassLoader.java:698)
    at java.base/jdk.internal.loader.BuiltinClassLoader.loadClassOrNull(Bui
l
1 LibreOffice Writer java:621)
    at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinCl
assLoader.java:579)
    at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(
ClassLoaders.java:178)
    at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:522)
    at org.martus.client.swingui.Martus.main(Martus.java:183)
Caused by: java.lang.ClassNotFoundException: javafx.application.Application
    at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinCl
assLoader.java:581)
    at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(
ClassLoaders.java:178)
    at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:522)
    ... 10 more
jayton@jayton-VirtualBox: ~/Martus/MartusClient-6.1.0$ java -jar martus.jar
```

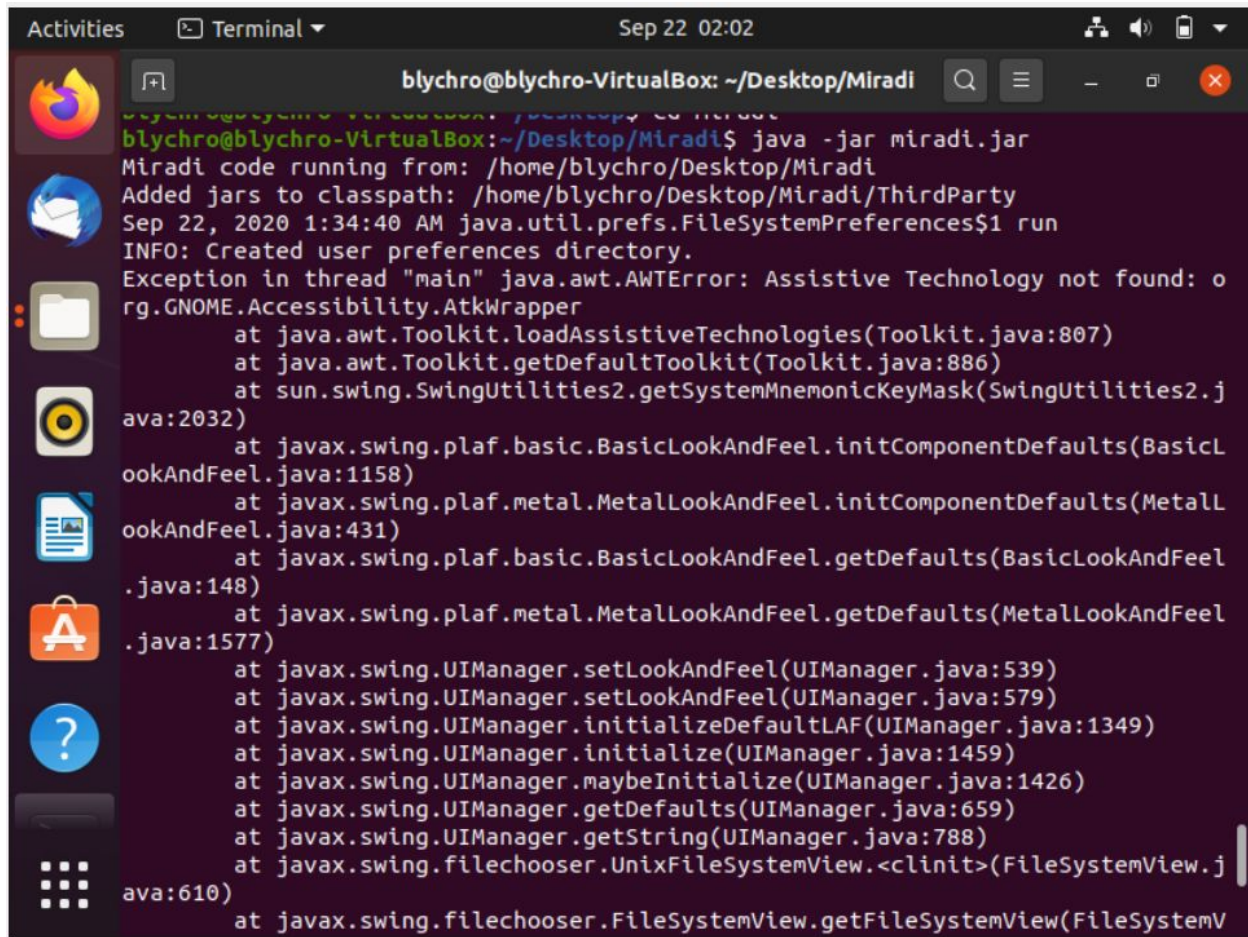


```
Activities Terminal Sep 22 00:42
blychro@blychro-VirtualBox: ~/Desktop/MartusClient-6.1.0
Java runtime: OpenJDK Runtime Environment
Memory Statistics: Using 3 megs of 15 megs; max avail=237
Running Martus from /home/blychro/Desktop/MartusClient-6.1.0
Added jar to classpath: bcprov-jdk15on-148.jar
Added jar to classpath: icu4j-3.4.4.jar
Added jar to classpath: jcommon-1.0.17.jar
Added jar to classpath: jfreechart-1.0.14.jar
Added jar to classpath: js-2006-03-08.jar
Added jar to classpath: junit-4.11.jar
Added jar to classpath: layouts-2006-08-10.jar
Added jar to classpath: orchid-1.0.0.jar
Added jar to classpath: persiancalendar-2.1.jar
Added jar to classpath: velocity-dep-1.4.jar
Added jar to classpath: commons-logging-1.1.jar
Added jar to classpath: ws-commons-util-1.0.2.jar
Added jar to classpath: xmlrpc-common-3.1.3.jar
Added jar to classpath: xmlrpc-client-3.1.3.jar
Tue 09/22 00:40:24 EDT Inactivity timeout set to 300 seconds
Exception in thread "main" java.awt.AWTError: Assistive Technology not found: o
rg.GNOME.Accessibility.AtkWrapper
    at java.awt.Toolkit.loadAssistiveTechnologies(Toolkit.java:807)
    at java.awt.Toolkit.getDefaultToolkit(Toolkit.java:886)
    at javax.swing.UIManager.getSystemLookAndFeelClassName(UIManager.java:6
11)
    at org.martus.client.swingui.Martus.useSystemLookAndFeel(Martus.java:24
4)
    at org.martus.client.swingui.Martus.main(Martus.java:180)
blychro@blychro-VirtualBox: ~/Desktop/MartusClient-6.1.0$ java -jar martus.jar
```

As a team we believe that assistance from Dr. Bowring will help with running the application. In the coming days we will also be attempting to run an earlier version of Martus that is better supported by Linux. As a backup plan we have also started working with Miradi, another H/FOSS project.

1.2 Potential Success

Miradi:



```
Activities Terminal Sep 22 02:02
blychro@blychro-VirtualBox: ~/Desktop/Miradi
blychro@blychro-VirtualBox: ~/Desktop/Miradi$ java -jar miradi.jar
Miradi code running from: /home/blychro/Desktop/Miradi
Added jars to classpath: /home/blychro/Desktop/Miradi/ThirdParty
Sep 22, 2020 1:34:40 AM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
Exception in thread "main" java.awt.AWTError: Assistive Technology not found: o
rg.GNOME.Accessibility.AtkWrapper
    at java.awt.Toolkit.loadAssistiveTechnologies(Toolkit.java:807)
    at java.awt.Toolkit.getDefaultToolkit(Toolkit.java:886)
    at sun.swing.SwingUtilities2.getSystemMnemonicKeyMask(SwingUtilities2.j
ava:2032)
    at javax.swing.plaf.basic.BasicLookAndFeel.initComponentDefaults(BasicL
ookAndFeel.java:1158)
    at javax.swing.plaf.metal.MetalLookAndFeel.initComponentDefaults(MetalL
ookAndFeel.java:431)
    at javax.swing.plaf.basic.BasicLookAndFeel.getDefaults(BasicLookAndFeel
.java:148)
    at javax.swing.plaf.metal.MetalLookAndFeel.getDefaults(MetalLookAndFeel
.java:1577)
    at javax.swing.UIManager.setLookAndFeel(UIManager.java:539)
    at javax.swing.UIManager.setLookAndFeel(UIManager.java:579)
    at javax.swing.UIManager.initializeDefaultLAF(UIManager.java:1349)
    at javax.swing.UIManager.initialize(UIManager.java:1459)
    at javax.swing.UIManager.maybeInitialize(UIManager.java:1426)
    at javax.swing.UIManager.getDefaults(UIManager.java:659)
    at javax.swing.UIManager.getString(UIManager.java:788)
    at javax.swing.filechooser.UnixFileSystemView.<clinit>(FileSystemView.j
ava:610)
    at javax.swing.filechooser.FileSystemView.getFileSystemView(FileSystemV
```

```
Activities Terminal Sep 22 01:43
blychro@blychro-VirtualBox: ~/Desktop/Miradi

at org.miradi.main.EAM.handleEamToMiradiMigration(EAM.java:606)
at org.miradi.main.EAM.initializeHomeDirectory(EAM.java:50)
at org.miradi.main.Miradi.main(Miradi.java:56)
blychro@blychro-VirtualBox:~/Desktop/Miradi$ java -Xms512m -jar miradi.jar
Miradi code running from: /home/blychro/Desktop/Miradi
Added jars to classpath: /home/blychro/Desktop/Miradi/ThirdParty
Exception in thread "main" java.awt.AWTError: Assistive Technology not found: o
rg.GNOME.Accessibility.AtkWrapper
at java.awt.Toolkit.loadAssistiveTechnologies(Toolkit.java:807)
at java.awt.Toolkit.getDefaultToolkit(Toolkit.java:886)
at sun.swing.SwingUtilities2.getSystemMnemonicKeyMask(SwingUtilities2.j
ava:2032)
at javax.swing.plaf.basic.BasicLookAndFeel.initComponentDefaults(BasicL
ookAndFeel.java:1158)
at javax.swing.plaf.metal.MetalLookAndFeel.initComponentDefaults(MetalL
ookAndFeel.java:431)
at javax.swing.plaf.basic.BasicLookAndFeel.getDefaults(BasicLookAndFeel
.java:148)
at javax.swing.plaf.metal.MetalLookAndFeel.getDefaults(MetalLookAndFeel
.java:1577)
at javax.swing.UIManager.setLookAndFeel(UIManager.java:539)
at javax.swing.UIManager.setLookAndFeel(UIManager.java:579)
at javax.swing.UIManager.initializeDefaultLAF(UIManager.java:1349)
at javax.swing.UIManager.initialize(UIManager.java:1459)
at javax.swing.UIManager.maybeInitialize(UIManager.java:1426)
at javax.swing.UIManager.getDefaults(UIManager.java:659)
at javax.swing.UIManager.getString(UIManager.java:788)
at javax.swing.filechooser.UnixFileSystemView.<clinit>(FileSystemView.j
ava:610)
```

While getting the initial download from Miradi was not an issue, we could not manage to make the program execute on Ubuntu. After trying a few different terminal commands, we attempted to run it in Windows, instead and managed to get it to run.

The screenshot shows the Miradi Desktop Software interface. The top navigation bar includes tabs for Project, Miradi Share, Team, Other Orgs, Scope, Location, Planning, TNC, WWF, WCS, RARE, and FOS. The main content area is titled 'Summary' and 'Progress Report'. It contains various fields for project information:

- Project Name: Miradi Marine Example 4.5
- Miradi Share Program: [Learn about Miradi Share](#)
- Primary Project Data Language: English
- Project Data Effective Date: 2016-09-01
- Project Filename: MarineExample_4_5_0
- External Project Ids: MiradiDatabase - 5463
- Project Number: EB 08-2
- Related Projects: Child project of the Western Island Conservation Area
- Project Website: www.eastbaymarine.org
- Project Description: (This is the example project used by Miradi Desktop Software. All information in this project is fictitious - any resemblance to a real-world project is purely coincidental.) The Eastern Bay Conservation Project (EBCP) is focused on working with the residents of Eastern Bay.
- Program Classifications: Date: 2018-06-30, Status: Minor Issues
- Progress Report: We have completed the project's first 3-year phase and are now in the 2nd year of the next phase. We are currently working on several key strategies as well as our monitoring plan. At this point, most of the

The test program provided demonstrates a way for an organization to document the work and projects and of its employees working on the aforementioned project. All project information is provided including project details, the project website, employees, and more. There is a tab to list employees along with their contact information, as well as tabs to display multiple organizations involved within the project and their information.

2. Deliverable 2:

2.1 Update

Since the first deliverable we have switched projects from Martus and Marati due to lack of a useful readme or wiki page. We are now using Open-MRS. Since switching we have been able to build the project and found methods to test.

2.2 Initial Test Cases

Test Number: 01

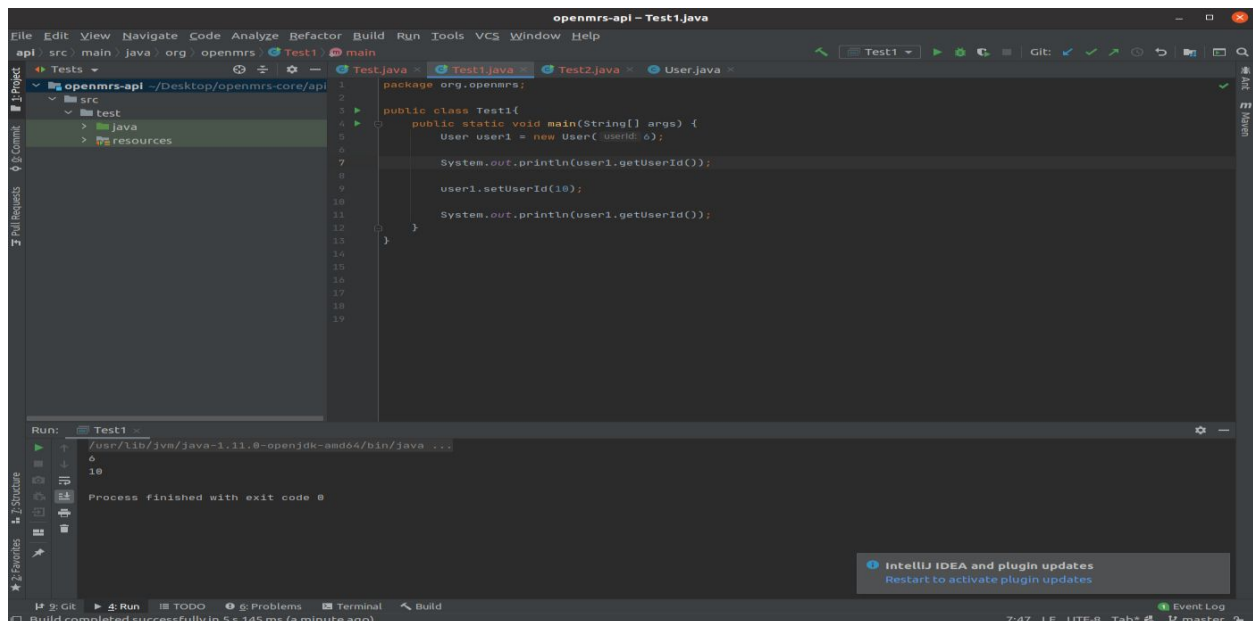
Requirement Being Tested: That setter method User only accepts Integers

Component Being Tested: User.java

Method Being Tested: public void setUserId(Integer userId)

Test Input: 10

Expected Outcomes: 10



The screenshot displays the IntelliJ IDEA IDE interface. The main editor window shows a Java file named `Test1.java` with the following code:

```
package org.openmrs;  
  
public class Test1 {  
    public static void main(String[] args) {  
        User user1 = new User( user1 );  
  
        System.out.println(user1.getUserId());  
  
        user1.setUserId(10);  
  
        System.out.println(user1.getUserId());  
    }  
}
```

The left sidebar shows the project structure with the following hierarchy:

- openmrs-api
 src
 test
 java
 resources

The bottom panel shows the Run configuration for `Test1` and the output of the execution:

```
Run: Test1  
/usr/lib/jvm/java-11.0-openjdk-amd64/bin/java ...  
10  
Process finished with exit code 0
```

The status bar at the bottom indicates that the build completed successfully in 55.145 ms (a minute ago).

Test Number: 02

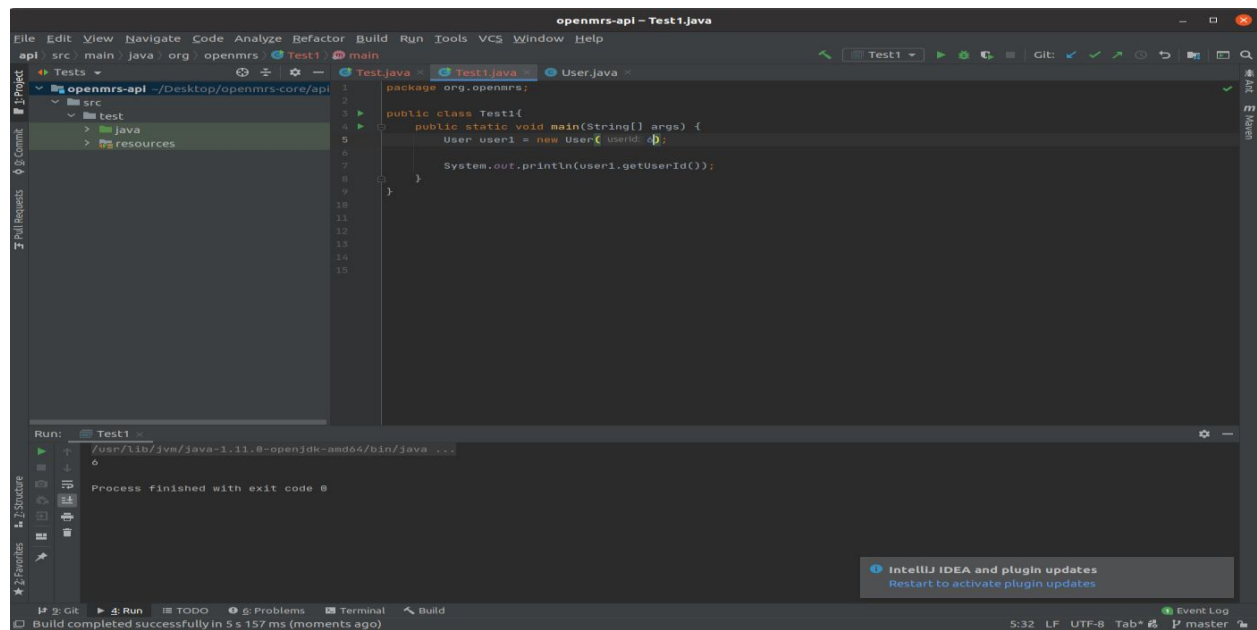
Requirement Being Tested: That getUserId actually returns userID

Component Being Tested: User.java

Method Being Tested: Public Integer getUserId()

Test Input: 6

Expected Outcomes: 6



Test Number: 03

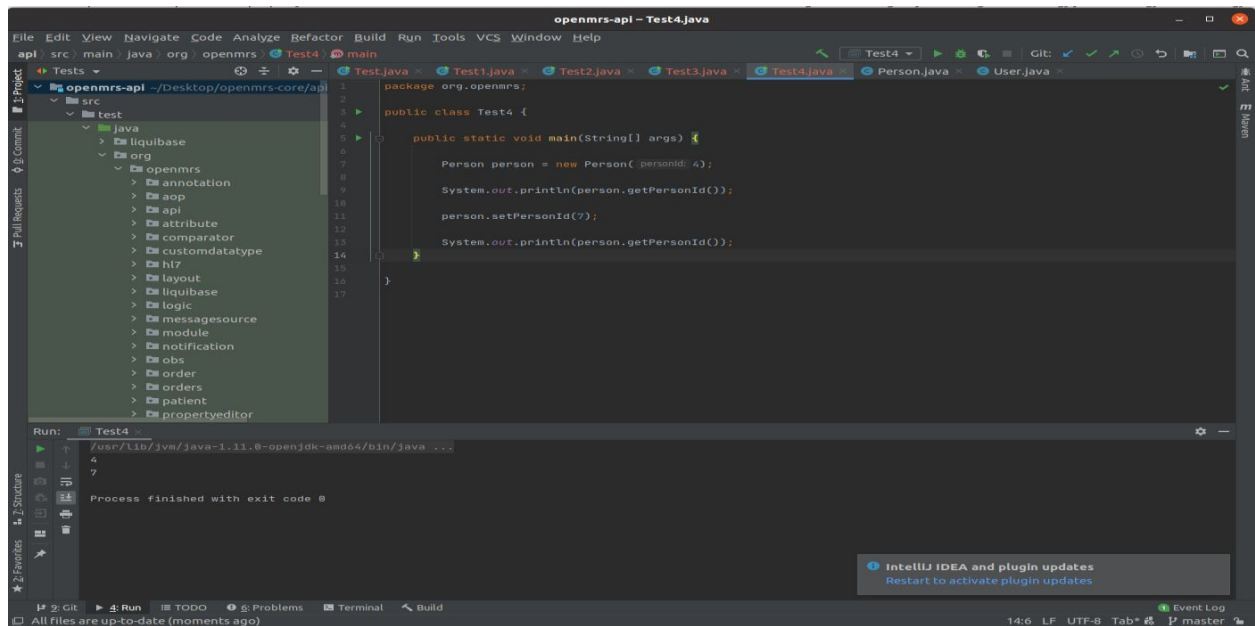
Requirement Being Tested: That setPersonId only takes an Integer

Component Being Tested: PersonID

Method Being Tested: Public Integer getUserID()

Test Input: 7

Expected Outcomes: 7



Test Number: 04

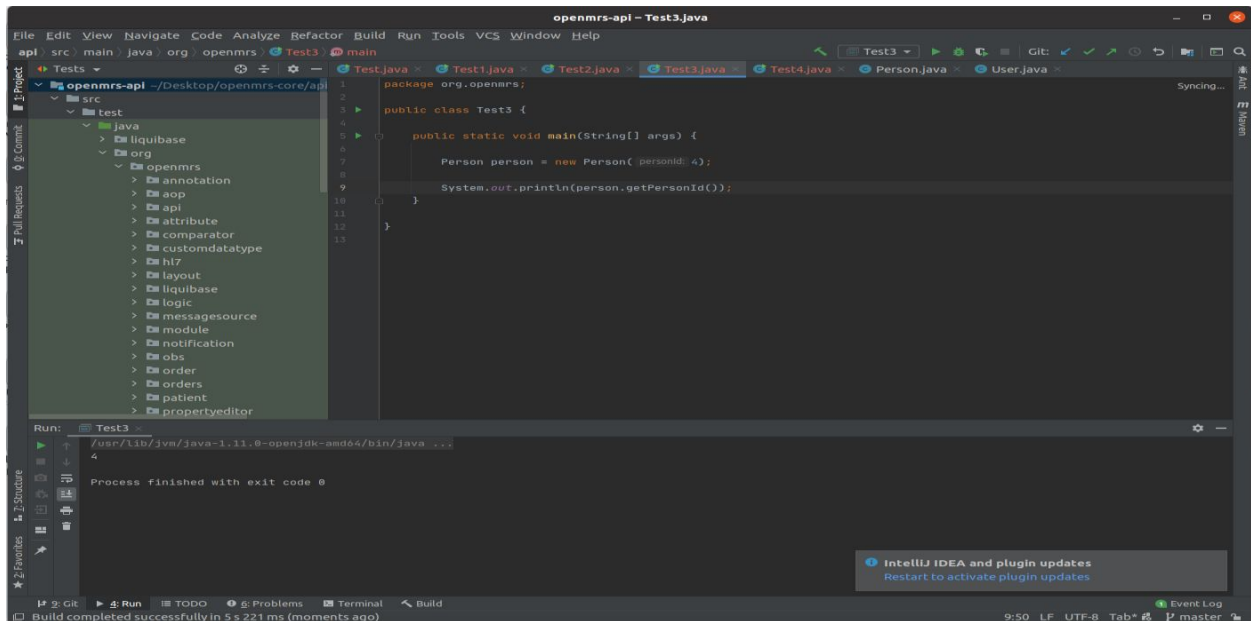
Requirement Being Tested: That getPersonId returns personId

Component Being Tested: Person.java

Method Being Tested: Public Integer getPersonId()

Test Input: 4

Expected Outcomes: 4



Test Number: 05

Requirement Being Tested: The Person can be reached through the User

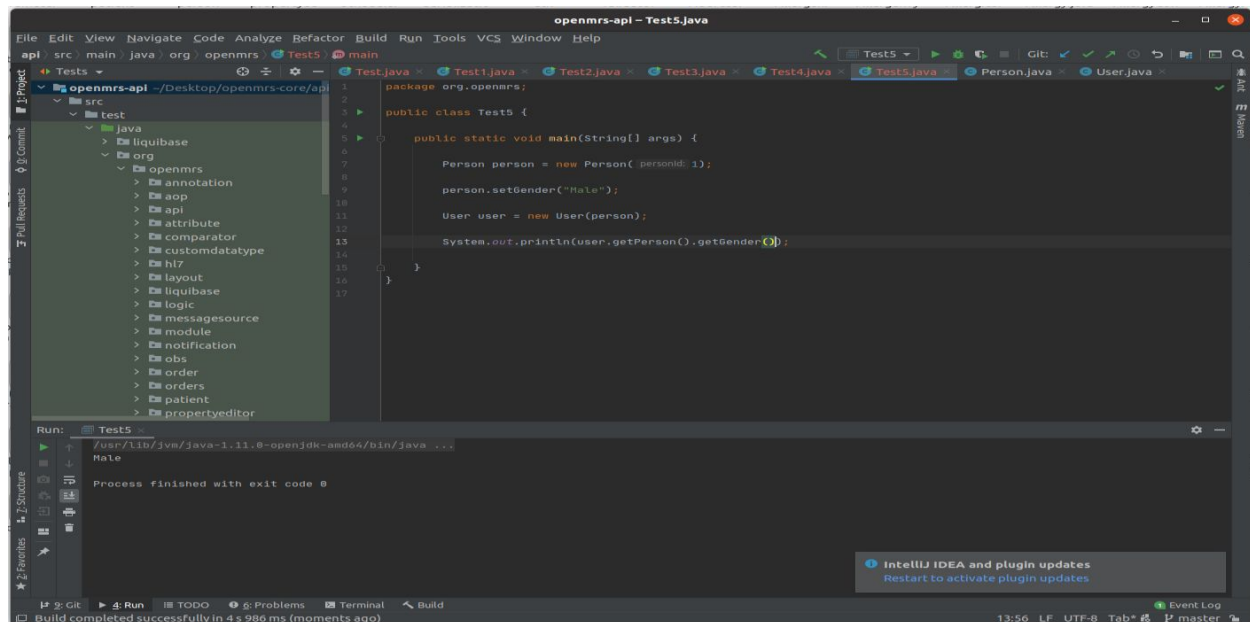
Component Being Tested: User.java

Method Being Tested: Public Person User(Person)

Test Input: "Male"

Expected Outcomes: "Male"

Purpose: While the return statement is returning the getGender() function, the purpose of the test was to confirm a user can be instantiated with a person.



2.3 Projected Test Case Schedule

Testing Schedule:

10/7 → Complete Test 01 and Test 02 in the command line

10/8→ Complete Test 03 and Test 04 in the command line

10/9 → Complete Test 05 and Test 06 in the command line

10/10→ Complete Test 07 and Test 08 in the command line

10/11 → Complete Test 09 and Test 10 in the command line

10/12→ Complete Test 11 and Test 12 in the command line

10/13 → Complete Test 13 and Test 14 in the command line

10/14→ Complete Test 15 and Test 16 in the command line

10/15 → Complete Test 17 and Test 18 in the command line

10/16→ Complete Test 19 and Test 20 in the command line

10/17 → Complete Test 21 and Test 22 in the command line

10/18→ Complete Test 23 and Test 24 in the command line

10/19 → Complete Test 25 in the command line

10/20-11/5 → Work on Creating an Automated testing framework from the tests run in the command line

2.4 Requirements and Constraints

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 and software requirements: We will be using linux, ubuntu, maven, and java 8 as our software tools. We are using our personal laptops as our hardware.

Constraints: One constraint would be only so much time is allotted for this team project. Another constraint would be our personal laptops do not run as fast as if we had supercomputers.

3. Deliverable 3:

3.1 Update

So far Team 7 has successfully created an automated testing framework. The issues that we have had so far is figuring out why our tests run in IntelliJ but when run through the command line the objects being used in the test are not recognized. For example when creating a User, the user object is not found. When this problem is solved the rest of our testing will fall into place.

3.2 Test Architecture Outline

In our github, the TestAutomation is the main folder where our framework lives. Once opened you'll see docs, oracles, project, scripts, reports, temp, testCases, and testCasesExecutables.

scripts: this directory is where our driver file and main scripts are located that execute our tests

driver.sh: this file is our driver file, runs all needed scripts and commands needed to run our tests and opens the final results for viewing after the scripts complete

reports: this directory is where our driver will create our finalReport file

Eventually this folder will hold our final testing report

3.3 Initial Test Cases:

Test ID	Requirement	Component	Method	Tested Input	Expected outcome
01	The setter method, User, only takes integers.	User.java	Public void setUserId(Integer userId)	10	10
02	getUserId returns the user Id	User.java	Public Integer getUserId()	6	6
03	The setter method, setPersonId, only takes integers.	Person.java	Public Integer setPersonId()	7	7
04	getPersonId returns the person Id	Person.java	Public Integer getPersonId()	4	4
05	The person can be reached through the user	User.java	Public person User(Person)	"Male"	"Male"

3.4 Frame Work Instructions:

Step 1: Clone the repository "git clone <https://github.com/csci-362-01-2020/Team-7>"

Step 2: Navigate to the directory where the driver is stored. You will need to copy this into the top directory of openmrs.

Step 3: In the top directory you will be able to run the script driver.sh(bash driver.sh)

4. Deliverable 4:

4.1 Updates:

We have finally managed to get our test cases to run from the command line. The issue was the openmrs project was relocating the .class files after compiling and we were trying to call them from their .java file locations. Tests 1-5 have been revamped to no longer be getters or setters and be more meaningful. The script has also been heavily overhauled to no longer be reliant on a java file and can run each test case from its own code. All test cases started hard coded but have since been updated to accept arguments. The script now uses test case text files to locate and implement argument inputs. Lastly, the finalReport.html file has been properly formatted and is no longer a large, jumbled, string of text.

4.2 25 Test Cases:

Test ID	Requirement	Component	Method	Tested Input	Expected outcome	Pass/Fail
01	Tests the method compareNatural Ascii when the value for s is greater than the value for t	NaturalStrings.java	compareNatural Ascii(String s, String t)	s: "adc" t: "abc"	A value greater than 0	Pass (returns 2)
02	Tests the method compareNatural Ascii when the value for t is greater than the value for s	NaturalStrings.java	compareNatural Ascii(String s, String t)	s: "adc" t: "abc"	A value less than 0	Pass (returns -2)
03	Tests the method compareNatural ignoreCaseAscii when the value for s is equal to the value for t	NaturalStrings.java	compareNatural ignoreCaseAscii(String s, String t)	s: "ABC" t: "abc"	0	Pass
04	Tests the method compareNatural Ascii when the value for t is greater than the value for s	NaturalStrings.java	compareNatural Ascii(String s, String t)	s: "Abc" t: "adC"	A value less than 0	Pass(return s -32)
05	The person can	User.java	getGender()	"Male"	"Male"	Pass

	be reached through the user					
06	Tests the compareTo method on how it handles when the first low inputted is greater than the second low and the first high is greater than the second high	DoubleRange.java	compareTo(DoubleRange other)	Range1 = low 78.9, high 90.1 Range2 = low 34.0, high 56.0	1	Pass
07	Tests the compareTo method on how it handles when the first low inputted is less than the second low and the first high value is greater than the second high value	DoubleRange.java	compareTo(DoubleRange other)	Range1 = low 34.0, high 90.1 Range2 = low 34.0, high 56.0	-1	Pass
08	Tests the compareTo method on how it handles when the first low value inputted is equal to the second low value and the first high value is equal to the second high value	DoubleRange.java	compareTo(DoubleRange other)	Range1 = low 34.0, high 34.0 Range2 = low 34.0, high 34.0	0	Pass
09	Test the fromSpecification method when a string is given as an argument	LocaleUtility.java	fromSpecification(String localeSpecification)	"eg_US_36"	"eg_US_36"	Pass
10	Test inputs a role and confirms the role was added	User.java	hasRole(String r)	"CEO"	True	Pass
11	Test inputs a role and confirms if the role makes the user a superuser	User.java	isSuperUser()	"System Developer"	True	Pass
12	Test inputs a	User.java	isSuperUser()	"Graphic	False	Pass

	role and confirms if the role makes the user a superuser			Designer"		
13	Test if the method contains can detect if the double value is in the range	DoubleRange.java	contains(double d)	20.9	True	Pass
14	Test inputs two allergens and sees if they're the same	Allergy.java	hasSameAllergen(Allergy allergy)	"DRUG"	True	Pass
15	Test inputs two allergens and sees if they're the same	Allergy.java	hasSameAllergen(Allergy allergy)	"DRUG", "FOOD"	False	Pass
16	Test if the method contains can detect if the double value is in the range, if the range is inclusive	DoubleRange.java	contains(double d)	10.0	True	Pass
17	Tests if the method supportsPropertyName returns false when a designated property name is not entered	LocaleUtility.java	supportsPropertyName(String propertyName)	"hello"	False	Pass
18	Tests if the method CompareNaturalAscii returns 0 if the two string arguments are equal	NaturalStrings.java	compareNaturalAscii(String s, String t)	s: "abc" t: "abc"	0	Pass
19	Test if the method contains can detect if the double value is not in the range	DoubleRange.java	contains(double d)	0.0	False	Pass
20	Tests if two Providers are the same	ProviderByPersonNameComparator.java	compare(Provider provider1, Provider provider2)	Provider(1), Provider(1)	0	Pass
21	Tests the formatPercentage method when	Format.java	formatPercentage(double pct)	null	" "	Pass

	a null value is passed as an argument					
22	Tests the format method when a null value is passed as an argument	Format.java	format(double d)	null	“ “	Pass
23	Tests the format method when a double is passed as an argument	Format.java	format(double d)	0.98	0.98	Pass
24	Tests the formatPercentage method when a double is passed as an argument	Format.java	formatPercentage(double pct)	0.98	98%	Pass
25	Tests if two Users are the same	UserByNameComparator.java	compare(User user1, User user2)	User(1) User(1)	1	Pass

5. Deliverable 5:

5.1 Fault Injections:

Code Injection 1:

File: UserByNameComparator.java

Line: 35

Injection: In the if statement change '==' to '!='.

Expected Test Failure: Test 25

Outcome before error:

Test ID 25

- Requirements: Tests if two Users are the same
- Component: UserByNameComparator.java
- Method: compare(User user1, User user2)
- Test Input: 1 1
- Expected Outcome: 1
- Actual Outcome: 1

Passed

Outcome after error:

Test ID 25

- Requirements: Tests if two Users are the same
- Component: UserByNameComparator.java
- Method: compare(User user1, User user2)
- Test Input: 1 1
- Expected Outcome: 1
- Actual Outcome: -1

Failed

Code Injection 2:

File: DoubleRange.java

Line: 120

Injection: In the if statement change '<=' to '>='.

Expected Test Failure: Test 13

Outcome before error:

Test ID 13

- Requirements: Test inputs a double and checks if the given double is inside the range.
- Component: DoubleRange.java
- Method: contains(double d)
- Test Input: 20.9
- Expected Outcome: true
- Actual Outcome: true

Passed

Outcome after error:

Test ID 13

- Requirements: Test inputs a double and checks if the given double is inside the range.
- Component: DoubleRange.java
- Method: contains(double d)
- Test Input: 20.9
- Expected Outcome: true
- Actual Outcome: false

Failed

Code Injection 3:

File: Format.java

Line: 37

Injection: In the if statement change '==' to '!='.

Expected Test Failure: Test 22

Outcome before error:

Test ID 22

- Requirements: Tests the format method when a null value is passed as an argument
- Component: Format.java
- Method: format(double d)
- Test Input: null
- Expected Outcome: ""
- Actual Outcome: ""

Passed

Outcome after error:

Method fails to complete

Exception in thread "main" java.lang.NullPointerException
at org.openmrs.util.Format.formatPercentage(Format.java:40)

at org.openmrs.Test21.main(Test21.java:17)

Exception in thread "main" java.lang.NullPointerException
at org.openmrs.util.Format.formatPercentage(Format.java:40)
at org.openmrs.Test21.main(Test21.java:17)

Code Injection 4:

File: Format.java

Line: 45

Injection: In the return statement change 'return " " + (d)' to 'return " + (d)"'.

Expected Test Failure: Test 23

Outcome before error:

Test ID 23

- Requirements: Tests the format method when a double is passed as an argument
- Component: Format.java
- Method: format(double d)

- Test Input: 0.98
- Expected Outcome: 0.98
- Actual Outcome: 0.98

Passed

Outcome after error:

Test ID 23

- Requirements: Tests the format method when a double is passed as an argument
- Component: Format.java
- Method: format(double d)
- Test Input: 0.98
- Expected Outcome: 0.98
- Actual Outcome: + (d)

Failed

Code Injection 5:

File: DoubleRange.java

Line: 133

Injection: In the final return statement change 'true' to 'false'

Expected Test Failure: Test 16

Outcome before error:

Test ID 16

- Requirements: Test inputs a double and checks if the given double is the same as the minimum if it is considered inside the range.
- Component: DoubleRange.java
- Method: contains(double d)
- Test Input: 10.0
- Expected Outcome: true
- Actual Outcome: true

Passed

Outcome after error:

Test ID 16

- Requirements: Test inputs a double and checks if the given double is the same as the minimum if it is considered inside the range.
- Component: DoubleRange.java
- Method: contains(double d)
- Test Input: 10.0
- Expected Outcome: true
- Actual Outcome: false

Failed

As expected, only tests that called methods with injected errors were changed, all other tests remained the same as before.

6. Final Report:

6.1 Updates:

The script has been edited to be more efficient. Instead of finding the driver in its location then using a naming system to get the right test case, the script now starts at the testCase directory and uses the information inside each file to locate the necessary driver and use the inputs needed. Many unnecessary if statements and loops were completely removed by these changes. Tests 13, 16, and 19 had some hard code that was taken out and added as arguments. Requirements were also modified to be more accurate to their intentions.

6.2 Testing Framework:

- The current testing framework is designed to be completely relative, so provided the user follows our instructions properly, no adjustments to the script should be necessary to account for different users.
- The scripts starts by wiping the finalResults.html and result.txt files to prepare for the new outputs
- Next, the script navigates to the test case text files and iterates through each file in the testCases directory. To help the user know how far the script has gotten it also prints what test case it is currently viewing as well.
- The script will create an array of seven parts, each one being a line found in each text file.
- Using the name of the specified driver, within the current test case, the script will navigate to the executable files within the openMRS project and run the predetermined driver with the given inputs from the test case.
- The output from each executable is then recorded to the results.txt file and the full content of the test case is formatted into html code and recorded in the finalResults.html file. The actual and expected outputs are compared, and the scripts records whether the test passed or failed.
- After every test file in the testCases directory has been executed and recorded, the scripts final task is to automatically open the finalResults.html in the user's preferred browser.

6.3 Requirements:

Ubuntu

One of the requirements for the class was that our project runs on a Linux Operating System. We have chosen to use Ubuntu

Java

OpenMRS is a Java application which is why you need to install a Java JDK.
If you want to build the master branch you will need a Java JDK of minimum version 8.

Maven

Install the build tool [Maven](#).

You need to ensure that Maven uses the Java JDK needed for the branch you want to build.

To do so execute: `mvn -version`

Git

Install the version control tool git and clone this repository with:

`git clone https://github.com/csci-362-01-2020/Team-7`

and the OpenMRS repository with:

`git clone https://github.com/openmrs/openmrs-core.git`

6.4 Instructions:

1. Download the TestAutomation folder and the openmrs-core repository from the openmrs project on github: <https://github.com/openmrs>.
2. Make sure the TestAutomation folder and the openmrs-core folder are both located in the same directory.
3. Copy all of the files located inside the TestAutomation/testCasesExecutables directory and paste them inside the openmrs-core/api/src/main/java/org/openmrs
4. Go into the OpenMRS code and comment out all instances of Log and for the security class comment out the first private method in openmrs-core/api/src/main/java/org/openmrs/util/naturalStrings java
5. Open the terminal and use the cd command to enter the top level of the openmrs-core directory
6. Once there, make sure you have maven and a java jdk of version 8 or higher installed and then compile the project with the command: `mvn clean install` (As a tip, once the system begins running the tests, it is finished compiling the project and the built in tests can be skipped by using `ctrl+c`. This is recommended because the tests can take up to 30 minutes to complete)
7. Once the compilation is complete, use the cd command to navigate to the TestAutomation directory
8. Grant yourself access to the file with the command: `chmod 777 scripts/runAllTests.sh`
9. To run the automated tests script use the command: `./scripts/runAllTests.sh`
10. The script should be fully automated from this point and automatically open your preferred browser with the results of each test once completed.

7. Reflections:

7.1 Self-Reflections:

Looking back on the assignment as a whole, we spent a lot of our time trying to get the openMRS project to cooperate with our drivers. It made us feel like we were constantly much farther behind than we actually were. Because we managed to use IntelliJ early on, we managed to keep up better than we would have been able to if the command line was our only way of testing our drivers. Our lack of knowledge when it came to BASH scripting made building the automation difficult, at first, but after doing our research and putting many small techniques together, the script came together nicely. It also meant, when the script had to be reworked to make it considerably more efficient, the adjustment time span was much faster than the initial building time frame.

Knowing what we know now about the assignment we shouldn't have procrastinated as often as we did. Our procrastination caused us to have to stay up until the late hours of the night the days before deliverable days. This project taught us how to write a script and how to work with an open source project. This project also allowed us to further our knowledge in working with github.

7.2 Assignment Reflections and Critiques:

Each deliverable did a good job of checkpointing the steps of the end result, but it did feel a bit heavy near the end. The first few deliverables had weeks between each one, but the final 3 deliverables were all due in the span of 3 and a half weeks. I think, maybe they should be more evenly distributed throughout the semester and have each deliverable add a few more test cases, so it becomes evident if a group is struggling to understand if they are doing the test cases correctly and can make adjustments earlier on. Deliverable 4 had the fully constructed automation and 20 of the 25 test cases due.