

The Boyoz - Final Report

Nathan Bell

Logan Sitar

Paul Joseph

Table of Contents:

Introduction.....	2
Chapter 1.....	3
- <i>Tanaguru Contrast Finder vs OpenMRS (pg. 3)</i>	
- <i>Testing OpenMRS (pg. 4-5)</i>	
- <i>Gained Experiences & Project Update (pg. 6)</i>	
Chapter 2.....	7
- <i>Testing Process, Requirements Traceability, & Tested Items (pg. 7)</i>	
- <i>Testing Schedule, Test Recording Procedures, Hardware and Software Requirements , & Constraints (pg. 8)</i>	
- <i>Unit Tests (pg. 8-10)</i>	
Chapter 3.....	11
- <i>Experiences, Description of Testing Framework, & How-To Documentation (pg. 11)</i>	
- <i>Example Test Cases (pg. 11-12)</i>	
Chapter 4.....	13
- <i>Experiences & Example Test Case (pg. 13)</i>	
- <i>Test Automation Code (pg. 14)</i>	
- <i>Example Output (pg. 15)</i>	
Chapter 5.....	16
- <i>Faults (pg. 16-18)</i>	
Chapter 6.....	19
- <i>Overall Experiences/Learnings, Self Evaluation of Team's Work, & Evaluation of Project Assignments (pg. 19)</i>	

Introduction:

In the process of developing an application, one must remember that the testing portion of the project is just as important as the development portion. This project stressed the importance of testing code and how to automate it.

To begin, an open source project was selected from a list, our top two contenders being OpenMRS and Tanaguru Contrast Finder. Although both projects had extensive documentation, readable code, and simple methods to test, we chose OpenMRS at first due to the successful deploying of the code using Maven. Once realizing the deployment of the code was not important for creating a test script, we switched to Tanaguru because the methods were easier to create test cases for.

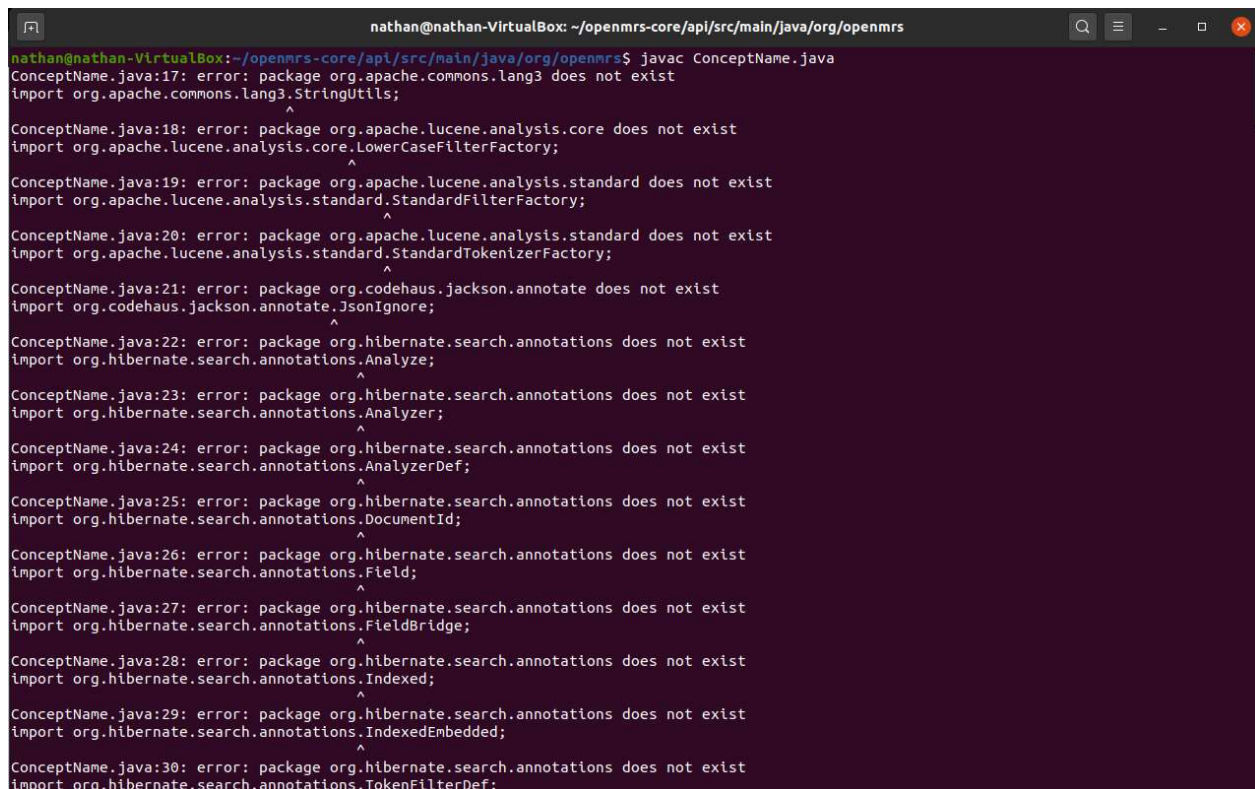
Once a testing framework was developed, twenty-five test cases were created. There were five different test cases for five different methods being tested from three different classes in the Tanaguru Contrast Finder open source code. An outline for each test case file can be seen on page eight under “Unit Tests”. A how-to for running the automated script is listed in Chapter 3 and the results are listed in Chapter 4 under “Example Output”.

Finally, faults were injected into the code to ensure the script would recognize when there was a different output than the expected output. The different faults injected can be seen in Chapter 5 and how the outcome of the tests is reflected in the results column with red text for failures and green for successes.

Chapter #1

Tanaguru Contrast Finder vs OpenMRS:

After finally understanding just exactly what we are expected to do with our choice of open source project for this class, we are considering switching what project we are going to test. While digging through the files of OpenMRS looking for java files that we could compile and test we came into quite a few issues. The main issue being that we seemingly cannot compile individual files. When typing “javac FileName.java” into terminal the output nearly always looks like this

A terminal window with a dark background and light text. The title bar reads "nathan@nathan-VirtualBox: ~/openmrs-core/apl/src/main/java/org/openmrs". The command prompt shows the user running "javac ConceptName.java". The output consists of ten lines of error messages, each starting with "ConceptName.java:17:" through "ConceptName.java:30:". Each error message indicates that a specific package does not exist, such as "package org.apache.commons.lang3 does not exist" and "package org.apache.lucene.analysis.core does not exist". Each error message is followed by an arrow pointing to the line number in the source file.

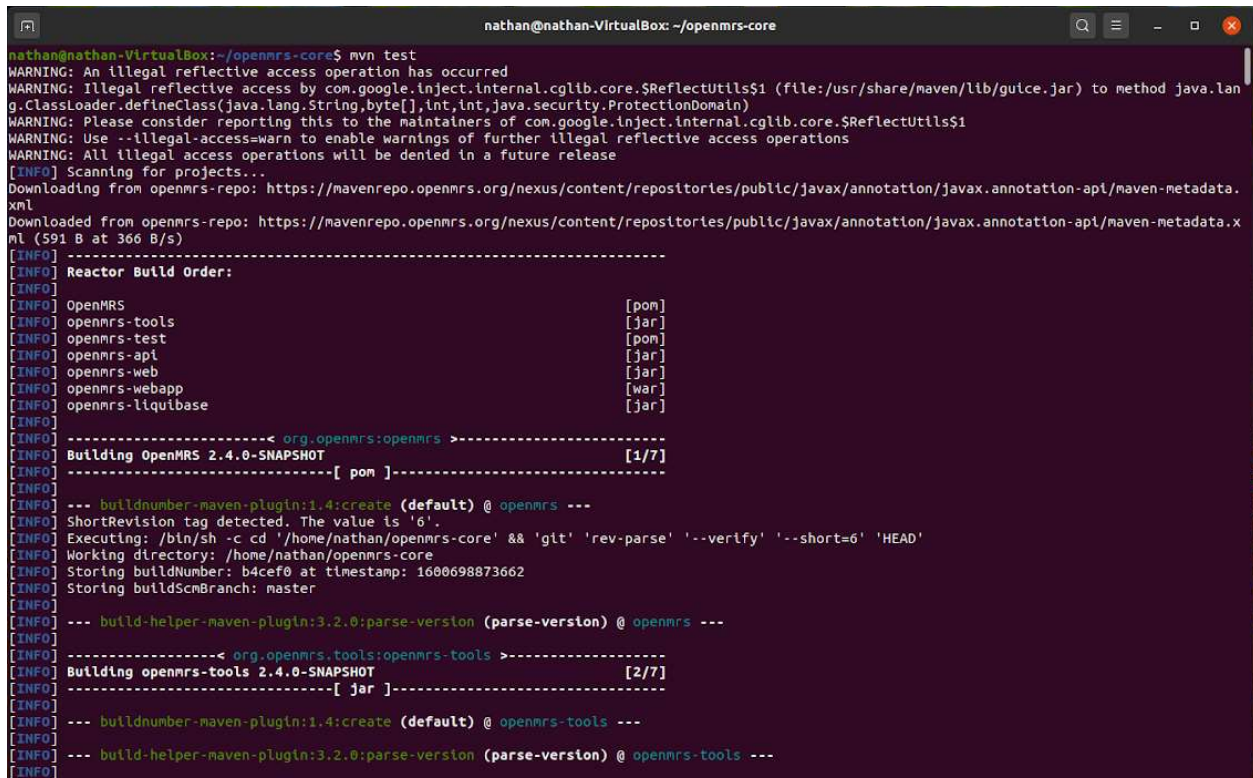
```
nathan@nathan-VirtualBox: ~/openmrs-core/apl/src/main/java/org/openmrs
nathan@nathan-VirtualBox:~/openmrs-core/apl/src/main/java/org/openmrs$ javac ConceptName.java
ConceptName.java:17: error: package org.apache.commons.lang3 does not exist
import org.apache.commons.lang3.StringUtils;
^
ConceptName.java:18: error: package org.apache.lucene.analysis.core does not exist
import org.apache.lucene.analysis.core.LowerCaseFilterFactory;
^
ConceptName.java:19: error: package org.apache.lucene.analysis.standard does not exist
import org.apache.lucene.analysis.standard.StandardFilterFactory;
^
ConceptName.java:20: error: package org.apache.lucene.analysis.standard does not exist
import org.apache.lucene.analysis.standard.StandardTokenizerFactory;
^
ConceptName.java:21: error: package org.codehaus.jackson.annotate does not exist
import org.codehaus.jackson.annotate.JsonIgnore;
^
ConceptName.java:22: error: package org.hibernate.search.annotations does not exist
import org.hibernate.search.annotations.Analyze;
^
ConceptName.java:23: error: package org.hibernate.search.annotations does not exist
import org.hibernate.search.annotations.Analyzer;
^
ConceptName.java:24: error: package org.hibernate.search.annotations does not exist
import org.hibernate.search.annotations.AnalyzerDef;
^
ConceptName.java:25: error: package org.hibernate.search.annotations does not exist
import org.hibernate.search.annotations.DocumentId;
^
ConceptName.java:26: error: package org.hibernate.search.annotations does not exist
import org.hibernate.search.annotations.Field;
^
ConceptName.java:27: error: package org.hibernate.search.annotations does not exist
import org.hibernate.search.annotations.FieldBridge;
^
ConceptName.java:28: error: package org.hibernate.search.annotations does not exist
import org.hibernate.search.annotations.Indexed;
^
ConceptName.java:29: error: package org.hibernate.search.annotations does not exist
import org.hibernate.search.annotations.IndexEmbedded;
^
ConceptName.java:30: error: package org.hibernate.search.annotations does not exist
import org.hibernate.search.annotations.TokenFilterDef;
```

When we ran into this we decided to take a step back at our original choice of project: Tanaguru Contrast Finder. Our reason for not selecting the contrast finder originally was because we were having trouble deploying it with Tomcat. Compiling it was very easy. Compiling and running individual java classes also works fine and running them is very simple. If this is what we are to create test cases for and automate for the next deliverables. Unless there is something we are not understanding, it is likely best that we switch our choice of project to the Tanaguru Contrast finder.

Testing OpenMRS:

Here are some screenshots of us running the maven test command in openMRS. The same can easily be done with the Contrast Finder

Starting OpenMRS testing with maven:



```
nathan@nathan-VirtualBox: ~/openmrs-core
nathan@nathan-VirtualBox:~/openmrs-core$ mvn test
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.inject.internal.cglib.core.$ReflectUtils$1 (file:/usr/share/maven/lib/guice.jar) to method java.lang.ClassLoader.defineClass(java.lang.String,byte[],int,int,java.security.ProtectionDomain)
WARNING: Please consider reporting this to the maintainers of com.google.inject.internal.cglib.core.$ReflectUtils$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
[INFO] Scanning for projects...
Downloading from openmrs-repo: https://mavenrepo.openmrs.org/nexus/content/repositories/public/javax/annotation/javax.annotation-api/maven-metadata.xml
Downloaded from openmrs-repo: https://mavenrepo.openmrs.org/nexus/content/repositories/public/javax/annotation/javax.annotation-api/maven-metadata.xml (591 B at 366 B/s)
[INFO] -----
[INFO] Reactor Build Order:
[INFO]
[INFO] OpenMRS [pom]
[INFO] openmrs-tools [jar]
[INFO] openmrs-test [pom]
[INFO] openmrs-api [jar]
[INFO] openmrs-web [jar]
[INFO] openmrs-webapp [war]
[INFO] openmrs-liquibase [jar]
[INFO]
[INFO] -----< org.openmrs:openmrs >-----
[INFO] Building OpenMRS 2.4.0-SNAPSHOT [1/7]
[INFO] -----[ pom ]-----
[INFO]
[INFO] --- buildnumber-maven-plugin:1.4:create (default) @ openmrs ---
[INFO] ShortRevision tag detected. The value is '6'.
[INFO] Executing: /bin/sh -c cd '/home/nathan/openmrs-core' && 'git' 'rev-parse' '--verify' '--short=6' 'HEAD'
[INFO] Working directory: /home/nathan/openmrs-core
[INFO] Storing buildNumber: b4cef0 at timestamp: 1600698873662
[INFO] Storing buildScmBranch: master
[INFO]
[INFO] --- build-helper-maven-plugin:3.2.0:parse-version (parse-version) @ openmrs ---
[INFO]
[INFO] -----< org.openmrs.tools:openmrs-tools >-----
[INFO] Building openmrs-tools 2.4.0-SNAPSHOT [2/7]
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- buildnumber-maven-plugin:1.4:create (default) @ openmrs-tools ---
[INFO]
[INFO] --- build-helper-maven-plugin:3.2.0:parse-version (parse-version) @ openmrs-tools ---
[INFO]
```


Example of an OpenMRS example test running:

```
Sep 21 10:52
nathan@nathan-VirtualBox: ~/openmrs-core

[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ openmrs-api ---
[INFO]
[INFO] T E S T S
[INFO]
[INFO] Running org.openmrs.AllergiesTest
[INFO] Tests run: 26, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 29.218 s - in org.openmrs.AllergiesTest
[INFO] Running org.openmrs.OrderTest
[INFO] Tests run: 64, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.307 s - in org.openmrs.OrderTest
[INFO] Running org.openmrs.EncounterTest
[INFO] Tests run: 63, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.402 s - in org.openmrs.EncounterTest
[INFO] Running org.openmrs.ProviderTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s - in org.openmrs.ProviderTest
[INFO] Running org.openmrs.ObsBehaviorTest
[WARNING] Tests run: 3, Failures: 0, Errors: 0, Skipped: 1, Time elapsed: 0.061 s - in org.openmrs.ObsBehaviorTest
[INFO] Running org.openmrs.TestOrderTest
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.002 s - in org.openmrs.TestOrderTest
[INFO] Running org.openmrs.ConceptReferenceTermTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.002 s - in org.openmrs.ConceptReferenceTermTest
[INFO] Running org.openmrs.annotation.OpenmrsProfileExcludeFilterWithModulesTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.049 s - in org.openmrs.annotation.OpenmrsProfileExcludeFilterWithModulesTest
[INFO] Running org.openmrs.annotation.OpenmrsProfileExcludeFilterTest
[WARNING] Tests run: 8, Failures: 0, Errors: 0, Skipped: 2, Time elapsed: 2.606 s - in org.openmrs.annotation.OpenmrsProfileExcludeFilterTest
[INFO] Running org.openmrs.annotation.StartModuleAnnotationTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.58 s - in org.openmrs.annotation.StartModuleAnnotationTest
[INFO] Running org.openmrs.annotation.StartModuleAnnotationReuseTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.292 s - in org.openmrs.annotation.StartModuleAnnotationReuseTest
[INFO] Running org.openmrs.annotation.OpenmrsProfileIncludeFilterTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.997 s - in org.openmrs.annotation.OpenmrsProfileIncludeFilterTest
[INFO] Running org.openmrs.PatientTest
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.026 s - in org.openmrs.PatientTest
[INFO] Running org.openmrs.UserTest
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s - in org.openmrs.UserTest
[INFO] Running org.openmrs.BaseCustomizableMetadataTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 3.255 s - in org.openmrs.BaseCustomizableMetadataTest
[INFO] Running org.openmrs.DrugOrderTest
[INFO] Tests run: 22, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.32 s - in org.openmrs.DrugOrderTest
[INFO] Running org.openmrs.util.SecurityTest
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.109 s - in org.openmrs.util.SecurityTest
[INFO] Running org.openmrs.util.ExceptionUtilTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.006 s - in org.openmrs.util.ExceptionUtilTest
[INFO] Running org.openmrs.util.DrugsByNameComparatorTest
```

OpenMRS maven end of testing output:

```
Sep 21 10:51
nathan@nathan-VirtualBox: ~/openmrs-core

[INFO] --- maven-resources-plugin:3.2.0:testResources (default-testResources) @ openmrs-liquibase ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Using 'UTF-8' encoding to copy filtered properties files.
[INFO] Copying 7 resources
[INFO] --- maven-compiler-plugin:3.8.1:testCompile (default-testCompile) @ openmrs-liquibase ---
[INFO] Nothing to compile - all classes are up to date
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ openmrs-liquibase ---
[INFO]
[INFO] T E S T S
[INFO]
[INFO] Running org.openmrs.liquibase.CoreDataTunerTest
[INFO] Tests run: 0, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.002 s - in org.openmrs.liquibase.CoreDataTunerTest
[INFO] Running org.openmrs.liquibase.AbstractSnapshotTunerTest
[INFO] Tests run: 0, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s - in org.openmrs.liquibase.AbstractSnapshotTunerTest
[INFO] Running org.openmrs.liquibase.MainTest
[INFO] Tests run: 0, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s - in org.openmrs.liquibase.MainTest
[INFO] Running org.openmrs.liquibase.SchemaOnlyTunerTest
[INFO] Tests run: 0, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s - in org.openmrs.liquibase.SchemaOnlyTunerTest
[INFO] Results:
[INFO] Tests run: 0, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] Reactor Summary for OpenMRS 2.4.0-SNAPSHOT:
[INFO] OpenMRS ..... SUCCESS [ 1.594 s]
[INFO] openmrs-tools ..... SUCCESS [ 0.973 s]
[INFO] openmrs-test ..... SUCCESS [ 0.033 s]
[INFO] openmrs-api ..... SUCCESS [13:43 min]
[INFO] openmrs-web ..... SUCCESS [32.778 s]
[INFO] openmrs-webapp ..... SUCCESS [ 1.579 s]
[INFO] openmrs-liquibase ..... SUCCESS [ 1.529 s]
[INFO] BUILD SUCCESS
[INFO] Total time: 14:25 min
[INFO] Finished at: 2020-09-21T10:48:54-04:00
[INFO]
nathan@nathan-VirtualBox: ~/openmrs-core$
```

Gained Experiences:

- Understanding Virtual Box software for virtual machines
 - allocating enough resources to the virtual machine
 - learning linux
- Working as a team
 - coordinating time to work together
 - splitting up work between members
- Installing and building packages from the terminal
 - compiling and running files through terminal to
- Building and compiling the open source project (OpenMRS)
- Running test cases
- Using Github Wiki
- Reinforcing git and github use

Project Update:

With the help of Dr. Bowring, we will decide between OpenMRS and Tanaguru Contrast Finder at the conclusion of this deliverable. If we choose to move one with OpenMRS we must find a way to compile the java files successfully and create a driver for it. If we choose Tanaguru then we will be one step ahead and already have the code compiled. We would just need to create the driver to test the code.

Chapter #2

The Testing Process

Testing will be done with a script command which will:

- compile classes that will be used in testing
- read test case files and execute methods with specified input
- compare the method's result to the expected output
- log result in an html file

Requirements Traceability

Calculates distance between two colors

Finds the contrast of an inputted color object

Takes a color object and outputs a string of the HSL value (hue, saturation, lightness)

Takes a string input of a hexadecimal value and converts those values to a string
RGB

Calculates Euclidian distance between two colors

Tested Items

ConstrastChecker.java

- **distanceColor()**
- **getConstrastRatio()**

ColorConverter.java

- **rgb2Hsl()**
- **hex2Rgb()**

DistanceCalculator.java

- **calculate()**

Testing Schedule

09/22/2020-10/05/2020: Design a detailed test plan and develop test cases for multiple methods and classes.

10/13/2020: Finish test cases and update current plan

10/27/2020: Finish Automated Tests

Test Recording Procedures

Each test result will be parsed into an html file and upon completion the file will be displayed on the user's browser.

Hardware and Software Requirements

Hardware: Possessing a computer that can allocate enough space in memory to clone the open-source documents

Software: Utilising Git and GitHub, Java, OS Terminal

Constraints

Meetings with Team (connectivity, work conflicts, sickness, emergencies, etc.)

Unit Tests

Example Test Case

Id:

Requirement:

Class:

Method:

Input:

Output:

Test Case 1:

1

Takes a color object and outputs a string of the HSL value (hue, saturation, lightness)

ColorConverter.java

rgb2Hsl

0,0,0

“hsl(0.0, 0.0%, 0.0%)”

Test Case 2:

2

Takes a color object and outputs a string of the HSL value (hue, saturation, lightness)

ColorConverter.java

rgb2Hsl

255,255,255

“hsl(0.0, 0.0%, 100.0%)”

Test Case 3:

3

Takes a color object and outputs a string of the HSL value (hue, saturation, lightness)

ColorConverter.java

rgb2Hsl

1,2,3

“hsl(210.0, 50.0%, .8%)”

Test Case 4:

4

Takes a color object and outputs a string of the HSL value (hue, saturation, lightness)

ColorConverter.java

rgb2Hsl

0,0,123

“hsl(240.0, 100.0%, 24.1%)”

Test Case 5:

5

Takes a color object and outputs a string of the HSL value (hue, saturation, lightness)

```
ColorConverter.java  
rgb2Hsl  
81,54,200  
“hsl(292.2, 57.5%, 49.8%)”
```

Chapter #3

Experiences

Being the first time that any of us have worked on any test automation, I think we came away with a lot of knowledge. We gained experience in Linux commands, reading and understanding open-source code, creating a script to automate our test cases, printing results to an html file and then removing any necessary files after execution. We also learned that the windows and Linux shells differ in how they save files. The hardest part about this task was understanding the open-source methods and what their functions were.

Description of Framework

If you take a look at the testing script you can see that what we did was pretty simple. First off we run a separate setup script that formats the html results file that we'll be adding to later. Then we compile all java classes in our executable folder which has both our drivers and tested classes inside. After that we loop through all of the test case files. For each test case file we have a while loop that populates variables with the data from each line. That data is then read to know what driver to run the given input in. The output from that driver is passed into another variable which is then checked against the expected output from the test case and is declared either a pass or a fail. All of the data is then passed into the results file as a row of a table. When the script is done looping through every test case it opens the results file in firefox before clearing the executables folder of all of the compiled class files.

How-To Documentation

In order to run this automated test case, clone the "The-BoyoZ" repository from github, find the cloned folder in your directories and cd into TestAutomations. Once your directory is set to TestAutomations, run `./scripts/runAllTests.sh` from the command line. An html file should open in your browser with the test case results.

Example Test Cases

Test Case 1:

1

Converts a color object to a HSL value
ColorConverter.java
rgb2Hsl
000000
hsl(0, 0%, 0%)

Test Case 2:

2
Converts a color object to a HSL value
ColorConverter.java
rgb2Hsl
ffffff
hsl(0, 0%, 100%)

Test Case 3:

3
Converts a color object to a HSL value
ColorConverter.java
rgb2Hsl
1234fc
hsl(231, 97%, 52%)

Test Case 4:

4
Converts a color object to a HSL value
ColorConverter.java
rgb2Hsl
abc123
hsl(68, 69%, 44%)

Test Case 5:

5
Converts a color object to a HSL value
ColorConverter.java
rgb2Hsl
8f32a6
hsl(288, 53%, 42%)

Chapter #4

Experiences

- Creating Test Cases
- Creating Drivers
- Analyzing open-source code
- Bash scripting
 - File io
 - Piping
- Learning basic CSS and HTML
 - Creating tables
 - Changing text and background colors

Example Test Case

Test Case:

```
1
  Converts a color object to a HSL value
  ColorConverter.java
  rgb2Hsl
  000000
  hsl(0, 0%, 0%)
```

Test Automation Code

```
for file in testCases/*.txt #loop though all test cases
do
    i=0
while read line #fill an array with the data from the test cases
do
    lines[$i]="$line";
    i=$((i+1));
done < $file

#move values from array into variables
declare id=${lines[0]}
declare requirement=${lines[1]}
declare class=${lines[2]}
declare method=${lines[3]}
declare input=${lines[4]}
declare expectedOutput=${lines[5]}
declare output
declare passFail

#Adds Driver to the given method to get driver name
declare driver=$method"Driver"

cd testCaseExecutables #move to the location of the drivers

#Runs driver to get the output
output=$(java $driver $input)

#Check to see if test passed or failed
#pass fail messages are formatted to be an element of a table in html
if [ "$output" == "$expectedOutput" ]
then
    passFail="<td style=\"color:#228B22\">pass</td>"
else
    passFail="<td style=\"color:#FF0000\">fail</td>"
fi
```

This is the bulk of the automated testing script. A simple for loop is the driving factor behind looking in every test case to get the information needed for testing. As you can see the pass/fail check is formatted so that the results display will have a corresponding red/green color to the outcome.

Example Output

ID	Requirement	Class	Method	Input	ExpectedOutput	Output	Result
1	Converts a color object to a HSL value	ColorConverter.java	rgb2Hsl	000000	hsl(0, 0%, 0%)	hsl(0, 0%, 0%)	pass
2	Converts a color object to a HSL value	ColorConverter.java	rgb2Hsl	ffffff	hsl(0, 0%, 100%)	hsl(0, 0%, 100%)	pass
3	Converts a color object to a HSL value	ColorConverter.java	rgb2Hsl	1234fc	hsl(231, 97%, 52%)	hsl(231, 97%, 52%)	pass
4	Converts a color object to a HSL value	ColorConverter.java	rgb2Hsl	abc123	hsl(68, 69%, 44%)	hsl(68, 69%, 44%)	pass
5	Converts a color object to a HSL value	ColorConverter.java	rgb2Hsl	8f32a6	hsl(288, 53%, 42%)	hsl(288, 53%, 42%)	pass
6	Converts hexadecimal value to a RGB value	ColorConverter.java	hex2Rgb	000000	java.awt.Color[r=0,g=0,b=0]	java.awt.Color[r=0,g=0,b=0]	pass
7	Converts hexadecimal value to a RGB value	ColorConverter.java	hex2Rgb	ffffff	java.awt.Color[r=255,g=255,b=255]	java.awt.Color[r=255,g=255,b=255]	pass
8	Converts hexadecimal value to a RGB value	ColorConverter.java	hex2Rgb	2f4a2c	java.awt.Color[r=47,g=74,b=44]	java.awt.Color[r=47,g=74,b=44]	pass
9	Converts hexadecimal value to a RGB value	ColorConverter.java	hex2Rgb	123abc	java.awt.Color[r=18,g=58,b=188]	java.awt.Color[r=18,g=58,b=188]	pass
10	Converts hexadecimal value to a RGB value	ColorConverter.java	hex2Rgb	a98cef	java.awt.Color[r=169,g=140,b=239]	java.awt.Color[r=169,g=140,b=239]	pass
11	Calculates Euclidian distance between two colors	DistanceCalculator.java	calculate	000000 ffffff	441.67	367.77	fail
12	Calculates Euclidian distance between two colors	DistanceCalculator.java	calculate	123abc abc123	255.04	135.0	fail
13	Calculates Euclidian distance between two colors	DistanceCalculator.java	calculate	a1b2c3 1a2b3c	233.83	194.7	fail
14	Calculates Euclidian distance between two colors	DistanceCalculator.java	calculate	1e8c66 a34c6b	147.68	127.87	fail
15	Calculates Euclidian distance between two colors	DistanceCalculator.java	calculate	10af7d fd65c4	258.24	236.72	fail
16	Calculates distance between two colors	ContrastChecker.java	distanceColor	000000 ffffff	441.6729559300637	441.6729559300637	pass
17	Calculates distance between two colors	ContrastChecker.java	distanceColor	123abc abc123	255.03529167548558	255.03529167548558	pass
18	Calculates distance between two colors	ContrastChecker.java	distanceColor	a1b2c3 1a2b3c	233.82685902179844	233.82685902179844	pass
19	Calculates distance between two colors	ContrastChecker.java	distanceColor	1e8c66 a34c6b	147.68209099278084	147.68209099278084	pass
20	Calculates distance between two colors	ContrastChecker.java	distanceColor	10af7d 10af7d	0.0	0.0	pass
21	Finds the contrast ratio of two colors	ContrastChecker.java	getContrastRatio	000000 ffffff	21.0	21.0	pass
22	Finds the contrast ratio of two colors	ContrastChecker.java	getContrastRatio	123abc abc123	4.405327061494763	4.405327061494763	pass
23	Finds the contrast ratio of two colors	ContrastChecker.java	getContrastRatio	a1b2c3 1a2b3c	6.648415996056606	6.648415996056606	pass
24	Finds the contrast ratio of two colors	ContrastChecker.java	getContrastRatio	1e8c66 a34c6b	1.3141695081977176	1.3141695081977176	pass
25	Finds the contrast ratio of two colors	ContrastChecker.java	getContrastRatio	10af7d fd65c4	1.0516702848607316	1.0516702848607316	pass

Chapter #5

Faults

Overview of added faults:

In order to view the added faults, open The-Boyo directory, change to the TestAutomation directory, then change to the testCaseExecutables directory and the faults will be located in the following Java Classes.

Class: ConstrastChecker.java

Method: distanceColor() : Added a statement to always return -1

```
public static double distanceColor(final Color fgColor, final Color bgColor) {
    int redFg = fgColor.getRed();
    int redBg = bgColor.getRed();
    int greenBg = bgColor.getGreen();
    int greenFg = fgColor.getGreen();
    int blueFg = fgColor.getBlue();
    int blueBg = bgColor.getBlue();

    //one of these lines needs to be commented out for the code to run correctly

    // uncomment this return and comment the other to break the code
    //return -1;

    //uncomment this return and comment the other to get correct output
    return (Math.sqrt(Math.pow(redFg - redBg, 2) + Math.pow(greenFg - greenBg, 2) + Math.pow(blueFg - blueBg, 2)));
}
```

Method: getContrastRatio(): Added a statement to always return -1

```
public static double getConstrastRatio(final Color fgColor, final Color bgColor) {
    double fgLuminosity = getLuminosity(fgColor);
    double bgLuminosity = getLuminosity(bgColor);

    //one of these lines needs to be commented out for the code to run correctly

    // uncomment this return and comment the if-else to break the code
    //return -1;

    //uncomment this if-else statement and comment the above return to get correct output
    if (fgLuminosity > bgLuminosity) {
        return computeContrast(fgLuminosity, bgLuminosity);
    } else {
        return computeContrast(bgLuminosity, fgLuminosity);
    }
}
```

Class: ColorConverter.java

Method: rgb2Hsl()

Added a statement to return the string "Fault"

```

public static String rgb2Hsl(Color color) {
    float[] hsvTab = new float[MAX_COMPONENT];
    Color.RGBtoHSB(color.getRed(), color.getGreen(), color.getBlue(), hsvTab);
    float h = hsvTab[HUE] * MAX_ANGLE;
    float l = (CONSTANT_SL_COMPONENTS_TWO - (hsvTab[SATURATION] * CONSTANT_SL_COMPONENTS_HUNDRED) / CONSTANT_SL_COMPONENTS_HUNDRED)
        * (hsvTab[BRIGHTNESS] * CONSTANT_SL_COMPONENTS_HUNDRED) / CONSTANT_SL_COMPONENTS_TWO;
    float s = (hsvTab[SATURATION] * CONSTANT_SL_COMPONENTS_HUNDRED) * (hsvTab[BRIGHTNESS] * CONSTANT_SL_COMPONENTS_HUNDRED)
        / (l < CONSTANT_S_COMPONENTS_FIFTY ? l * CONSTANT_SL_COMPONENTS_TWO : CONSTANT_S_COMPONENTS_TWO_HUNDRED - l * CONSTANT_SL_COMPONENTS_TWO);

    //one of these lines needs to be commented out for the code to run correctly

    // uncomment this return and comment the other to break the code
    //return "fault";

    //uncomment this return statement and comment the above return to get correct output
    return ("hsl(" + Float.valueOf(h).intValue()
        + ", " + Float.valueOf(s).intValue() + "%"
        + ", " + Float.valueOf(l).intValue() + "%" + ")");
}

```

Method: hex2Rgb(): Added a statement to always return null

```

public static Color hex2Rgb(String colorStr) {
    //one of these lines needs to be commented out for the code to run correctly

    // uncomment this return and comment the other to break the code
    return null;

    //uncomment this if-else statement and "return null;" and comment the above return to get correct output
    if (colorStr.charAt(0) == '#') {
        String str = colorStr.substring(1);
        if (str.matches(HEXADECIMAL_DICTIONNARY)
            && str.length() == RGB_HEX_LENGTH) {
            return getNewColor(str);
        } else if (str.matches(HEXADECIMAL_DICTIONNARY)
            && str.length() == RGB_SHORT_HEX_LENGTH) {
            return getNewColorShortHexa(str);
        }
    } else if (colorStr.matches(HEXADECIMAL_DICTIONNARY)) {
        if (colorStr.length() == RGB_HEX_LENGTH) {
            return getNewColor(colorStr);
        } else if (colorStr.length() == RGB_SHORT_HEX_LENGTH) {
            return getNewColorShortHexa(colorStr);
        }
    }
    return null;
}

```

Class: DistanceCalculator.java

Method: calculate()

This method already did not correctly calculate the distance so it was not changed

ID	Requirement	Class	Method	Input	ExpectedOutput	Output	Result
1	Converts a color object to a HSL value	ColorConverter.java	rgb2Hsl	000000	hsl(0, 0%, 0%)	fault	fail
2	Converts a color object to a HSL value	ColorConverter.java	rgb2Hsl	ffffff	hsl(0, 0%, 100%)	fault	fail
3	Converts a color object to a HSL value	ColorConverter.java	rgb2Hsl	1234fc	hsl(231, 97%, 52%)	fault	fail
4	Converts a color object to a HSL value	ColorConverter.java	rgb2Hsl	abc123	hsl(68, 69%, 44%)	fault	fail
5	Converts a color object to a HSL value	ColorConverter.java	rgb2Hsl	8f32a6	hsl(288, 53%, 42%)	fault	fail
6	Converts hexadecimal value to a RGB value	ColorConverter.java	hex2Rgb	000000	java.awt.Color[r=0,g=0,b=0]	java.awt.Color[r=0,g=0,b=0]	pass
7	Converts hexadecimal value to a RGB value	ColorConverter.java	hex2Rgb	ffffff	java.awt.Color[r=255,g=255,b=255]	java.awt.Color[r=255,g=255,b=255]	pass
8	Converts hexadecimal value to a RGB value	ColorConverter.java	hex2Rgb	2f4a2c	java.awt.Color[r=47,g=74,b=44]	java.awt.Color[r=47,g=74,b=44]	pass
9	Converts hexadecimal value to a RGB value	ColorConverter.java	hex2Rgb	123abc	java.awt.Color[r=18,g=58,b=188]	java.awt.Color[r=18,g=58,b=188]	pass
10	Converts hexadecimal value to a RGB value	ColorConverter.java	hex2Rgb	a98cef	java.awt.Color[r=169,g=140,b=239]	java.awt.Color[r=169,g=140,b=239]	pass
11	Calculates Euclidian distance between two colors	DistanceCalculator.java	calculate	000000 ffffff	441.67	367.77	fail
12	Calculates Euclidian distance between two colors	DistanceCalculator.java	calculate	123abc abc123	255.04	135.0	fail
13	Calculates Euclidian distance between two colors	DistanceCalculator.java	calculate	a1b2c3 1a2b3c	233.83	194.7	fail
14	Calculates Euclidian distance between two colors	DistanceCalculator.java	calculate	1e8c66 a34c6b	147.68	127.87	fail
15	Calculates Euclidian distance between two colors	DistanceCalculator.java	calculate	10af7d fd65c4	258.24	236.72	fail
16	Calculates distance between two colors	ContrastChecker.java	distanceColor	000000 ffffff	441.6729559300637	441.6729559300637	pass
17	Calculates distance between two colors	ContrastChecker.java	distanceColor	123abc abc123	255.03529167548558	255.03529167548558	pass
18	Calculates distance between two colors	ContrastChecker.java	distanceColor	a1b2c3 1a2b3c	233.82685902179844	233.82685902179844	pass
19	Calculates distance between two colors	ContrastChecker.java	distanceColor	1e8c66 a34c6b	147.68209099278084	147.68209099278084	pass
20	Calculates distance between two colors	ContrastChecker.java	distanceColor	10af7d 10af7d	0.0	0.0	pass
21	Finds the contrast ratio of two colors	ContrastChecker.java	getContrastRatio	000000 ffffff	21.0	21.0	pass
22	Finds the contrast ratio of two colors	ContrastChecker.java	getContrastRatio	123abc abc123	4.405327061494763	4.405327061494763	pass
23	Finds the contrast ratio of two colors	ContrastChecker.java	getContrastRatio	a1b2c3 1a2b3c	6.648415996056606	6.648415996056606	pass
24	Finds the contrast ratio of two colors	ContrastChecker.java	getContrastRatio	1e8c66 a34c6b	1.3141695081977176	1.3141695081977176	pass
25	Finds the contrast ratio of two colors	ContrastChecker.java	getContrastRatio	10af7d fd65c4	1.0516702848607316	1.0516702848607316	pass

The faults only affect the method they are inserted into. As you can see here the fault in the rgb2Hsl method has been activated which changes the output and result only of the five rgb2Hsl test cases. The calculate method fails every time due to an error in the code.

Chapter #6

Overall Experiences & Learnings:

Being the first time that any of us have worked on any test automation, I think we came away with a lot of knowledge. We gained experience in Linux commands, reading and understanding open-source code, building drivers to run code for testing, creating a script to automate our test cases, printing results to an html file and then removing any necessary files after execution. We also learned that the windows and Linux shells differ in how they save files. In addition, we learned how to create tables and edit text and background colors using CSS and HTML. Having a finished product that runs smoothly, it is safe to say that we have gained much experience throughout the semester.

Self Evaluation of Team's Work:

Throughout the semester, our team has worked well together to coordinate days to meet online and work on our project virtually. Each member did their part to communicate which section of the project they would be working on. All in all everyone contributed to meeting the criteria of each deliverable. We all worked well together as a team to figure out how to make our program efficiently run, even though none of us had experience with what we were doing, nor were we given instructions on how it should be done.

Evaluation of Project Assignments:

Overall, The-BoyoZ thought every assignment on this project was valuable and aided in creating an efficient automated testing script. The assignments that were new and had the most impact on us would be the creation of the test cases, the automated script and the printing of results on a html page. Somethings that we were already aware of but the assignments strengthened our knowledge of were Git and GitHub. Working strictly remotely, The-BoyoZ lived through branching, merging, pushing and pulling code to and from a repository. This is a valuable skill that is used throughout the Computer Engineering field so it was beneficial practicing consistently with it. We also valued being thrown in the deep end from the start and having to figure things out on our own. This project was a great simulation of what the rest of our Computer Engineering careers could potentially be like.