Project Report

An Automated Testing Framework for Tanaguru's Contrast Finder

CSCI 362 - Software Engineering

Professor: Dr.Jim Bowring

Group Cougars-Roar:

Fabiola Atoche

Luke Bradley

Julian Smith

James Thurlow

<u>Table of contents</u>

<u>Introduction</u>

For our semester long software engineering project, we (Cougars-Roar) chose to test Tanaguru. Tanaguru is an online application that evaluates web page template drafts, individual pages, full sites or even web applications. It automates several accessibility tests used for evaluating if a website falls under the international regulation of the WCAG (Web Content Accessibility Guidelines). Tanaguru is upheld by using three qualities: transparency, efficiency and ease of use. Transparency lets the user know if there are any problems that could compromise results, efficiency attempts to save time without cutting corners on results, and ease of use makes understanding accessibility standards easy for the average user. The contrast finder within Tanaguru is an automated test that allows users to find contrasting colors for foregrounds and backgrounds of created sites. Inputs include RGB decimal or hexadecimal values. The application makes sure the contrast ratio is higher than the minimum standard set by the WCAG at 4:5:1. Even if a user doesn’t pick a correct ratio, the contrast finder gives a list of similar ratios that do correctly meet WCAG standards.

Detailed within our final report are our team updates on certain chapters (iterations) that span throughout the semester. The earlier chapters indicate beginning steps of implementation and testing, while the later chapters introduce test cases, test reports, fault injections, scripting and limitations. Our team progress is shown as the chapters continue on.

Chapter 1

We selected SugarsLabs for our team project.

SugarLabs, offers good documentation on how to install and build the project. SugarLabs community is still active and offer help. However, there was many issues during installation and building of SugarLabs. One of the main issues was having the wrong version of Ubuntu. Also, making sure that all dependencies were installed.

There were [multiple methods](https://developer.sugarlabs.org/dev-environment.md.html) for building SugarLabs, but the only successful way was utilizing the packaged style build. Tests, however, were unable to be run due to module load errors. Currently, we are attempting to fix the module file paths within the test files and further imports in an attempt to eventually run the tests.

Sugar Labs does have very well documented wiki which held a fair amount information on how their tests were created, as well as a guide for creating tests for their software.

One especially common error was being denied permission to the "/bin/bash: /bome/broot/.bash_profile" and "…/sugar-build/osbuild" packages when attempting to build the SugarLabs shell. This may be due to running an older version of Ubuntu, or at least an older version of the Fedora software that comes packaged with it.

Update

As a result of the SugarLabs version being incompatible with our Linux system version, we decided to switch over to the Tanaguru Contrast Finder. The setup found here, but it was not sufficient for describing the whole difficult process of implementing it.

Requirements for Tanaguru Contrast Finder:

1. Git
2. Tomcat
3. Maven

Getting Contrast Finder

```
$ git clone https://github.com/Tanaguru/Contrast-Finder.git
```

Getting Maven

```
$ apt-cache search maven
$ sudo apt-get install maven
```

Using Maven to Build Tanaguru Contrast Finder

```
$ cd Contrast-Finder
$ mvn clean install
```

All tests are run and the project is built. The deployable WAR file is found following the maven build command within /Contrast-Finder/contrast-finder-webapp/target/.

Setting up Tomcat and Deploying

The primary setup of Tomcat was done using Digital Ocean's Tomcat Setup Tutorial.

Unfortunately this was not enough to deploy the web application, for there were issues with the context of the webapp. In searching through the context.xml file inside of the webapp META-INF we found a Spring Framework spring-instrument that was not being loaded. This was because the jar containing this artifact was not installed within tomcat's library. So we installed the spring-framework-instrument jar file from here and then:

```
$ cd ~/Downloads/
$ mv spring-framework-4.3.11.jar /opt/tomcat/lib/
```

Once the jar is added to the tomcat library, we edited tomcat’s context.xml file to include the Loader, ClassLoader. The context.xml found in /opt/tomcat/conf should then have something like this:

```
<Context>
  <Loader
loaderClass="org.springframework.instrument.classloading.tomcat.TomcatInstrumentableClass
Loader"/>
</Context>
```

Once this was complete the last step was setting up the configuration for Tanaguru by doing the following:

```
$ mkdir /etc/contrast-finder/
$ cd ~/Contrast-Finder/contrast-finder-resources/src/main/resources
$ cp contrast-finder.conf /etc/contrast-finder/
```

Once this is done one may finally deploy the app:

```
$ cd ~/Contrast-Finder/contrast-finder-webapp/target
$ cp contrast-finder.war /opt/tomcat/webapps/ #contrast-finder is followed by whatever
version number
```

Then we went to localhost:8080 and the app was running and deployed.

Tests were then able to be run by us. These tests were mainly found to be validation oriented.

Chapter 2

Test Plan

1. **Introduction**

Purpose:
The Tanaguru Contrast-Finder module evaluates if the color used by web developers complies with the international regulation established by the WCAG (Web Content Accessibility Guidelines).
This test plan details our testing approach and framework that will execute the testing of Tanaguru Contrast-Finder. Contrast-Finder module is a tool that provides web developers the ability to review and evaluate the colors of a page or an entire web application. Contrast-Finder is reliable, intuitive and self-accessible.

Scope:
This test plan is defined to test methods included in the Contrast-Finder module.

Objective:
The objective of the test plan is to verify that the functionality of Contrast-Finder meets the specifications found on the Tanaguru GitHub documentation

2. **Testing Process**

The testing process will begin with the selection of methods to be tested.
Only Unit Testing will be performed.

The first step is to design test cases and store them in the folder testCases. The test cases also contain the test data and expected outputs, that will be used to test the methods.

A script "runAllTests. sh" is created to execute multiple test cases.

As tests are executed the results are evaluated as Pass/ Fail in the test report.

If a test is marked as Fail, either a failure in the method was detected or the test itself may have a failure. Both method and test case need to be reviewed.

3. **Requirements Traceability**

Tanaguru Contrast Finder is a tool which computes the contrast between two colors (background, foreground) and checks if the contrast is valid. Foreground is the text color and background is the page color.
Requirements:
- Both fields can be filled with a hexadecimal value between #000000 and #FFFFFF.
- Note that #ABC works also, it is automatically expanded in the form #AABBCC.
- Both fields can be filled without the '#'character, so FFF or FFFFFF works perfectly.
- The contrast ratio between foreground and background has three possible comparative values listed in order from worst to best accessibility: 3, 4.5 and 7 respectively.

1. Tested Items

Method getContrastRatio5DigitRound() from the ContrastChecker component
Method getLuminosity() from the ContrastChecker component
Method calculate() from the DistanceCalculator component
Method rgb2hsl() from the ColorConverter component
Method hex2rgb() from the ColorConverter component

2. Testing Schedule

Activity	Start	End	Team Member
Prepare Test Plan	September 26	October 3	Fabiola & Luke
Review Test Plan	October 3	October 24	Fabiola & James
Prepare Test Cases	September 26	November 14	All team members
Prepare Test Scripts	October 3	October 18	Julian
Execute 5 Test Cases	October 18	October 31	Julian
Review Testing	October 31	November 14	All team members
Create 5 Fault Injections	November 16	November 20	All team members
Complete Final Report	November 21	November 27	All team members

3. Test Recording Procedures

1. First, the driver traverses the test cases folder and receives a list of files (testcase#.txt)

2. Once it receives the files, it reads each individual test case and organizes the information in this type of format

    1. test ID
    2. requirement being tested
    3. component being tested
    4. method being tested
    5. test input(s)
    6. expected outcome(s)

    By setting up this format, we are able to retrieve each piece of information that we need for executing the driver

3. Next, we format each piece of information into their appropriate data type

4. Methods are executed to retrieve the actual result returned by the method

5. The actual value is compared to the expected value (as specified in test case text file) to produce a pass or fail result

6. Lastly, we format and write the results into a table for proper visual clarity.

1. **Software and Hardware requirements**
    - Java
    - Linux (Ubuntu 64-bit)
    - Terminal
    - Web browser (or preferred html viewing application)

2. **Constraints**
    - Besides daily production constraints, our only main team constraints have involved timing issues.

3. **Tests Cases**

Test 01

Requirement being tested: This method gets the contrast ratio between two colors
Component being tested: ContrastChecker
Method being tested: getContrastRatio5DigitRound(String fgColor, String bgColor)
Test input : 255, 255, 255; 255, 255, 255

Expected outcome: 1.0

Test 02
Requirement being tested: This method gets the contrast ratio between two colors
Component being tested: ContrastChecker
Method being tested: getContrastRatio5DigitRound(String fgColor, String bgColor)
Test input : 255, 255, 255; 0, 0, 0
Expected outcome: 21.0

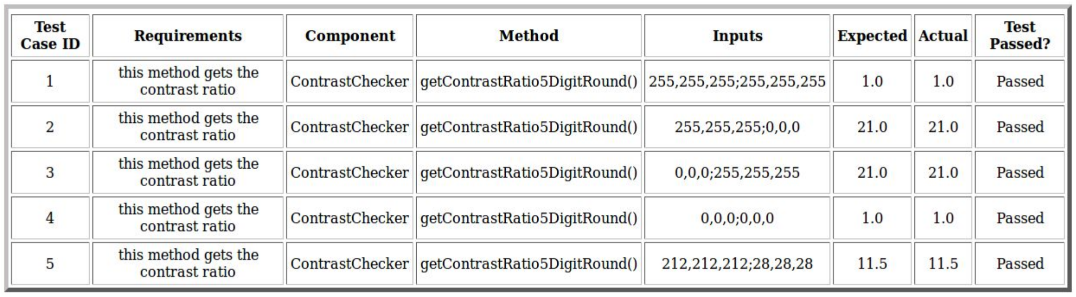
Test 03
Requirement being tested: This method gets the contrast ratio between two colors
Component being tested: ContrastChecker
Method being tested: getContrastRatio5DigitRound(String fgColor, String bgColor)
Test input : 0, 0, 0; 255, 255, 255
Expected outcome: 21.0

Test 04
Requirement being tested: This method gets the contrast ratio between two colors
Component being tested: ContrastChecker
Method being tested: getContrastRatio5DigitRound(String fgColor, String bgColor)
Test input : 0, 0, 0; 0, 0, 0
Expected outcome: 1.0

Test 05
Requirement being tested: This method gets the contrast ratio between two colors
Component being tested: ContrastChecker
Method being tested: getContrastRatio5DigitRound(String fgColor, String bgColor)
Test input : 212, 212, 212; 28, 28, 28
Expected outcome: 11.5

Chapter 3

The task assigned was to create an automatic testing framework for our first 5 test cases. Said testing framework was formatted and presented as follows:



Test Case ID	Requirements	Component	Method	Inputs	Expected	Actual	Test Passed?
1	this method gets the contrast ratio	ContrastChecker	getContrastRatio5DigitRound()	255,255,255;255,255,255	1.0	1.0	Passed
2	this method gets the contrast ratio	ContrastChecker	getContrastRatio5DigitRound()	255,255,255;0,0,0	21.0	21.0	Passed
3	this method gets the contrast ratio	ContrastChecker	getContrastRatio5DigitRound()	0,0,0;255,255,255	21.0	21.0	Passed
4	this method gets the contrast ratio	ContrastChecker	getContrastRatio5DigitRound()	0,0,0;0,0,0	1.0	1.0	Passed
5	this method gets the contrast ratio	ContrastChecker	getContrastRatio5DigitRound()	212,212,212;28,28,28	11.5	11.5	Passed

Overview:

A testing framework for Tanaguru's Contrast Finder

Framework Directory Structure and Descriptions:

```
Cougar-Roar/
    TestAutomation/
        docs/ #nothing yet
        oracles/ #nothing yet (testCases acts as an oracle)
        project/ #contains Tanaguru's Contrast-Finder source code
            /src
            /bin
        reports/ #contains the output of testcases in a html format
            report.html
        scripts/ #contains the runAllTests script which activates the Drivers and manages files
            runAllTests.sh
        temp/ #contains individual test case reports which are appended into the reports.html
        testCases/ #contains 25 test case files which contain passabke values for the drivers
        testCasesExecutables/ #contains
            ContrastCheckerDriver.java
    README.md
    Makefile
```

How to execute testing:

```
git clone

cd Cougars-Roar

make run
```

If you experience the following error:

```
./runAllTests.sh: line 10: javac: command not found

./runAllTests.sh: line 10: javac: command not found

cat: '../temp/*': No such file or directory
```

Run the following command:

```
java -version
```

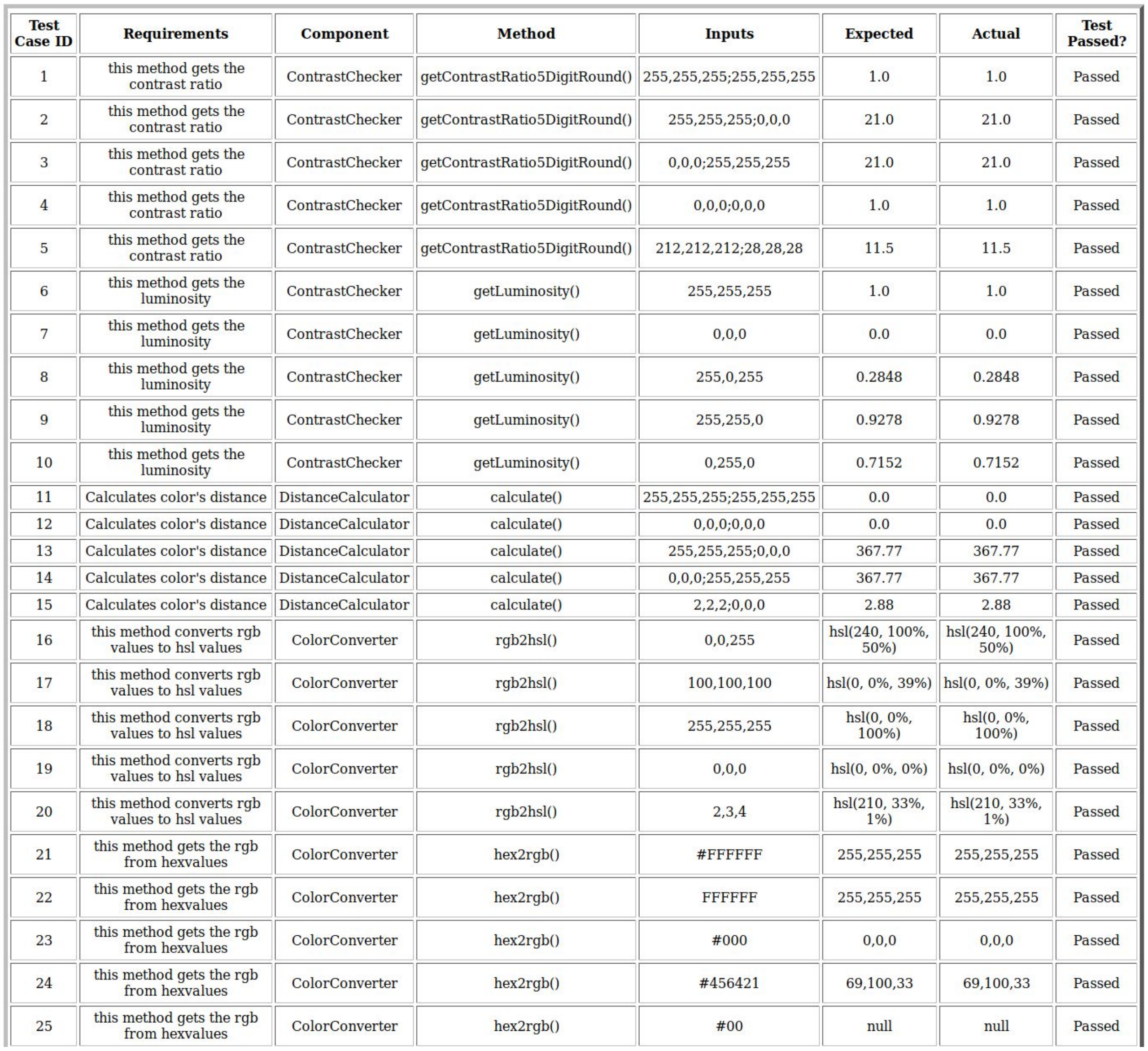
If nothing is returned then run:

```
sudo apt-get install default-jdk
```

If the issue is not corrected submit an issue to our GitHub:
https://github.com/CSCI-362-02-2017/Cougars-Roar

Chapter 4

We added twenty more testcases that will test:
Method getLuminosity() from the ContrastChecker component
Method calculate() from the DistanceCalculator component
Method rgb2hsl() from the ColorConverter component
Method hex2rgb() from the ColorConverter component
Also, two test drivers were created to execute the testcases. The unexpected outcome was getLuminosity method did not round up decimal numbers. Our strategy was to change the testcases' inputs to obtain results with numbers that only had up to four decimal places.
After executing the script we finally got a successful testing result.



Test Case ID	Requirements	Component	Method	Inputs	Expected	Actual	Test Passed?
1	this method gets the contrast ratio	ContrastChecker	getContrastRatio5DigitRound()	255,255,255;255,255,255	1.0	1.0	Passed
2	this method gets the contrast ratio	ContrastChecker	getContrastRatio5DigitRound()	255,255,255;0,0,0	21.0	21.0	Passed
3	this method gets the contrast ratio	ContrastChecker	getContrastRatio5DigitRound()	0,0,0;255,255,255	21.0	21.0	Passed
4	this method gets the contrast ratio	ContrastChecker	getContrastRatio5DigitRound()	0,0,0;0,0,0	1.0	1.0	Passed
5	this method gets the contrast ratio	ContrastChecker	getContrastRatio5DigitRound()	212,212,212;28,28,28	11.5	11.5	Passed
6	this method gets the luminosity	ContrastChecker	getLuminosity()	255,255,255	1.0	1.0	Passed
7	this method gets the luminosity	ContrastChecker	getLuminosity()	0,0,0	0.0	0.0	Passed
8	this method gets the luminosity	ContrastChecker	getLuminosity()	255,0,255	0.2848	0.2848	Passed
9	this method gets the luminosity	ContrastChecker	getLuminosity()	255,255,0	0.9278	0.9278	Passed
10	this method gets the luminosity	ContrastChecker	getLuminosity()	0,255,0	0.7152	0.7152	Passed
11	Calculates color's distance	DistanceCalculator	calculate()	255,255,255;255,255,255	0.0	0.0	Passed
12	Calculates color's distance	DistanceCalculator	calculate()	0,0,0;0,0,0	0.0	0.0	Passed
13	Calculates color's distance	DistanceCalculator	calculate()	255,255,255;0,0,0	367.77	367.77	Passed
14	Calculates color's distance	DistanceCalculator	calculate()	0,0,0;255,255,255	367.77	367.77	Passed
15	Calculates color's distance	DistanceCalculator	calculate()	2,2,2;0,0,0	2.88	2.88	Passed
16	this method converts rgb values to hsl values	ColorConverter	rgb2hsl()	0,0,255	hsl(240, 100%, 50%)	hsl(240, 100%, 50%)	Passed
17	this method converts rgb values to hsl values	ColorConverter	rgb2hsl()	100,100,100	hsl(0, 0%, 39%)	hsl(0, 0%, 39%)	Passed
18	this method converts rgb values to hsl values	ColorConverter	rgb2hsl()	255,255,255	hsl(0, 0%, 100%)	hsl(0, 0%, 100%)	Passed
19	this method converts rgb values to hsl values	ColorConverter	rgb2hsl()	0,0,0	hsl(0, 0%, 0%)	hsl(0, 0%, 0%)	Passed
20	this method converts rgb values to hsl values	ColorConverter	rgb2hsl()	2,3,4	hsl(210, 33%, 1%)	hsl(210, 33%, 1%)	Passed
21	this method gets the rgb from hexvalues	ColorConverter	hex2rgb()	#FFFFFF	255,255,255	255,255,255	Passed
22	this method gets the rgb from hexvalues	ColorConverter	hex2rgb()	FFFFFF	255,255,255	255,255,255	Passed
23	this method gets the rgb from hexvalues	ColorConverter	hex2rgb()	#000	0,0,0	0,0,0	Passed
24	this method gets the rgb from hexvalues	ColorConverter	hex2rgb()	#456421	69,100,33	69,100,33	Passed
25	this method gets the rgb from hexvalues	ColorConverter	hex2rgb()	#00	null	null	Passed

Chapter 5

The task for deliverable 5 was to design and inject 5 faults into Tanaguru code. The following are the fault injections.

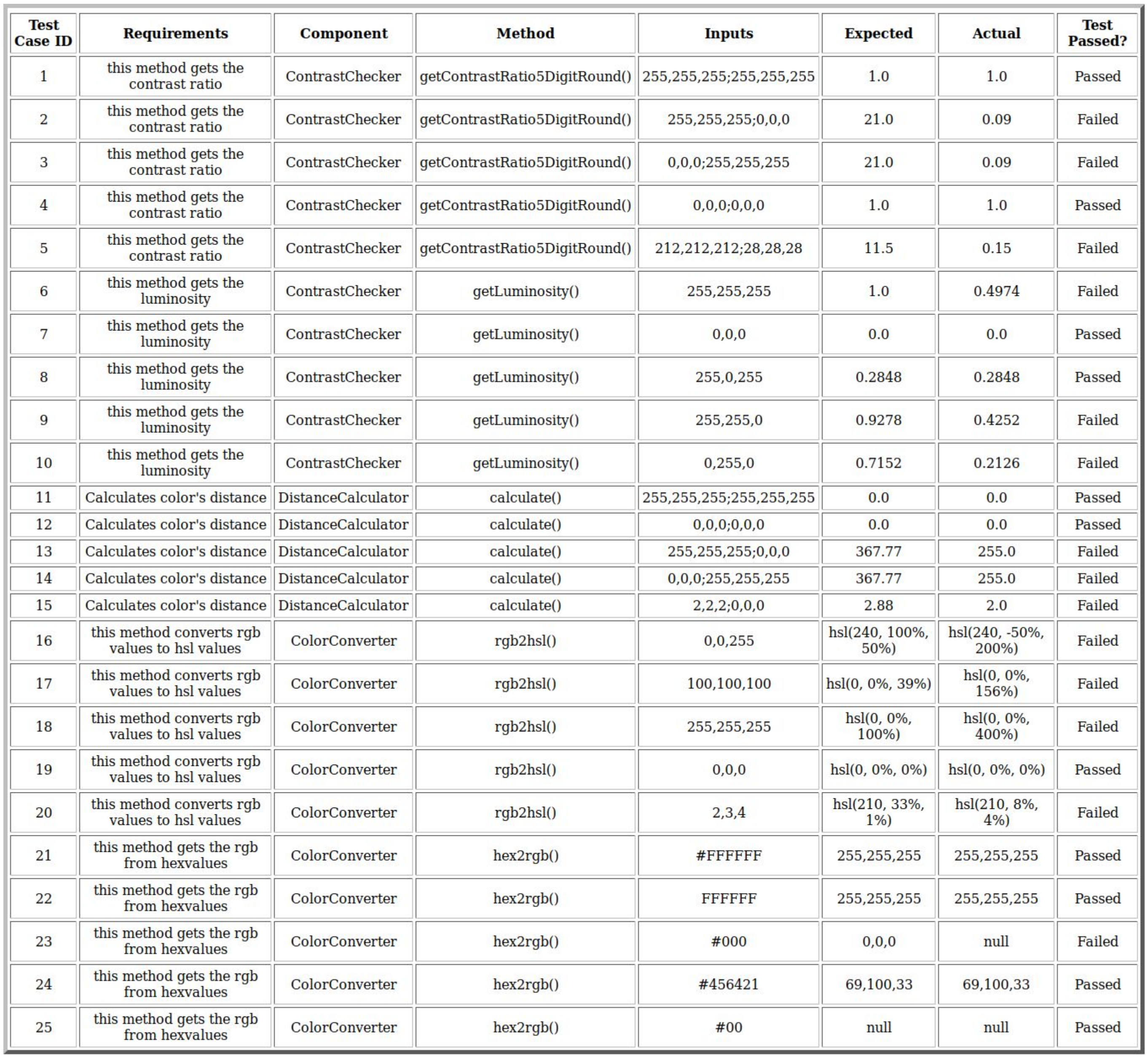
Component: ColorConverter
method : hex2Rgb()
changed: str.length() == RGB_SHORT_HEXA_LENGTH)
to: str.length() == RGB_HEXA_LENGTH)
In this case the method would not accept short_hex input
Expected outcome: fail testcase # 23

Component: ColorConverter
method : rgb2Hsl()
changed : * (CONSTANT_SL_COMPONENTS_HUNDRED)
to : + (CONSTANT_SL_COMPONENTS_HUNDRED)
Expected outcome: fail testscases # 16, 71, 18 and 20.

Component ContrastChecker
method :getConstrastRatio5DigitRound()
changed :if(fgLuminosity < bgLuminosity)
to : if(fgLuminosity > bgLuminosity)
Expected Outcome: fail testcases # 2, 3, 5.

Component ContrastChecker
method : getLuminisity()
changed : getComposantValue(color.getGreen()) * GREEN_FACTOR
to : getComposantValue(color.getRed()) * GREEN_FACTOR
Expected Outcome: Fail Test Cases # 6, 9, 10.

Component: DistanceCalculator
method : calculate()
changed : + Math.pow(Double.valueOf(colorToChange.getBlue()) - Double.valueOf(colorToKeep.getBlue()), CUBIC)))) * ROUND_VALUE) / ROUND_VALUE
to : - Math.pow(Double.valueOf(colorToChange.getBlue()) - Double.valueOf(colorToKeep.getBlue()), CUBIC)))) * ROUND_VALUE) / ROUND_VALUE
Expected Outcome : Fail testcases # 13, 14, 15.



Test Case ID	Requirements	Component	Method	Inputs	Expected	Actual	Test Passed?
1	this method gets the contrast ratio	ContrastChecker	getContrastRatio5DigitRound()	255,255,255;255,255,255	1.0	1.0	Passed
2	this method gets the contrast ratio	ContrastChecker	getContrastRatio5DigitRound()	255,255,255;0,0,0	21.0	0.09	Failed
3	this method gets the contrast ratio	ContrastChecker	getContrastRatio5DigitRound()	0,0,0;255,255,255	21.0	0.09	Failed
4	this method gets the contrast ratio	ContrastChecker	getContrastRatio5DigitRound()	0,0,0;0,0,0	1.0	1.0	Passed
5	this method gets the contrast ratio	ContrastChecker	getContrastRatio5DigitRound()	212,212,212;28,28,28	11.5	0.15	Failed
6	this method gets the luminosity	ContrastChecker	getLuminosity()	255,255,255	1.0	0.4974	Failed
7	this method gets the luminosity	ContrastChecker	getLuminosity()	0,0,0	0.0	0.0	Passed
8	this method gets the luminosity	ContrastChecker	getLuminosity()	255,0,255	0.2848	0.2848	Passed
9	this method gets the luminosity	ContrastChecker	getLuminosity()	255,255,0	0.9278	0.4252	Failed
10	this method gets the luminosity	ContrastChecker	getLuminosity()	0,255,0	0.7152	0.2126	Failed
11	Calculates color's distance	DistanceCalculator	calculate()	255,255,255;255,255,255	0.0	0.0	Passed
12	Calculates color's distance	DistanceCalculator	calculate()	0,0,0;0,0,0	0.0	0.0	Passed
13	Calculates color's distance	DistanceCalculator	calculate()	255,255,255;0,0,0	367.77	255.0	Failed
14	Calculates color's distance	DistanceCalculator	calculate()	0,0,0;255,255,255	367.77	255.0	Failed
15	Calculates color's distance	DistanceCalculator	calculate()	2,2,2;0,0,0	2.88	2.0	Failed
16	this method converts rgb values to hsl values	ColorConverter	rgb2hsl()	0,0,255	hsl(240, 100%, 50%)	hsl(240, -50%, 200%)	Failed
17	this method converts rgb values to hsl values	ColorConverter	rgb2hsl()	100,100,100	hsl(0, 0%, 39%)	hsl(0, 0%, 156%)	Failed
18	this method converts rgb values to hsl values	ColorConverter	rgb2hsl()	255,255,255	hsl(0, 0%, 100%)	hsl(0, 0%, 400%)	Failed
19	this method converts rgb values to hsl values	ColorConverter	rgb2hsl()	0,0,0	hsl(0, 0%, 0%)	hsl(0, 0%, 0%)	Passed
20	this method converts rgb values to hsl values	ColorConverter	rgb2hsl()	2,3,4	hsl(210, 33%, 1%)	hsl(210, 8%, 4%)	Failed
21	this method gets the rgb from hexvalues	ColorConverter	hex2rgb()	#FFFFFF	255,255,255	255,255,255	Passed
22	this method gets the rgb from hexvalues	ColorConverter	hex2rgb()	FFFFFF	255,255,255	255,255,255	Passed
23	this method gets the rgb from hexvalues	ColorConverter	hex2rgb()	#000	0,0,0	null	Failed
24	this method gets the rgb from hexvalues	ColorConverter	hex2rgb()	#456421	69,100,33	69,100,33	Passed
25	this method gets the rgb from hexvalues	ColorConverter	hex2rgb()	#00	null	null	Passed

Chapter 6

Fabiola Atoche
Looking back, at the beginning of the class, I can see the huge progress I have experienced. First I learned to use Git and Github, and to appreciate how powerful this version control system is, having a repository that allows everyone to collaborate on the same project. Second, I learned to write bash scripts and execute them. Although creating the script for the project was difficult and I was able to complete the task with the help of my teammates, I was able to finally see how important and powerful it is to master this skill. Third, working in a team, assigning roles and tasks to each team member as well as defining deadlines and sticking to them. Fourth, writing the test plan and test cases. I was a very busy this term, so it seemed to go by very fast and I feel I increased my knowledge in class as well as outside of the classroom.

James Thurlow
At the beginning of class back in August I remember being pretty intimidated by the entire structure for this software engineering class along with its semester long project expectations. It started out with a couple of bumps, but with a lot of trial and error I feel I have really come a long way and learned a ton. I knew what Github was, but I didn’t know how to navigate and use it properly. Now I know where to go and how to properly collaborate on the application. Same thing with Git; i used help from teammates and tutorials on Codecademy to help me learn that barrier and use it through my terminal. Scripting is something i struggled with a lot during the semester, but i was able to make proper strides in continuing to learn more about it after not knowing anything going in. Testing and re-testing code was something I learned how to do in earlier computer science classes, but building reports upon those tests was new to me and I learned how to do that this semester. Lastly, this was my first big team software project that I had ever worked on. I was relatively new to computer science when I first got to college, and after a couple years of learning the realms I

finally contributed to a large project that was most certainly worthwhile. I had always played sports growing up so I knew how to operate on a team, but working in a software environment was totally different and it helped me develop my team building skills.

Luke Bradley
At the beginning of this class, I was really unsure what the semester project was even really about, much less how to do it, and I had little idea of how to create and run a script for anything. I already had experience with Github in a couple of earlier classes, but using Ubuntu/Linux, VirtualBox, and actual Git in command line was new to me, as was Bash scripting, and I feel like I learned a lot this semester just from experimenting with those and seeing how they worked. While I had done team collaborations with computer science peers in the past, this was the first time that I had worked on a project of this caliber with other people, and it definitely helped the scope not seem nearly as daunting to have it split up among the team. Communication was sometimes difficult, but whenever we couldn't meet up due to schedule conflicts we were still able to work on the project remotely and keep each other updated. I think overall I learned quite a bit about scripting, operating Linux and command line, and working with others from this project this semester, and had a good bit of fun doing it as well.

Julian Smith
Throughout the semester, by help of our professor, Dr. Bowring, and my team, I was able to experience and enjoy working on this project in a team environment. Fortunately, I was already experience in a good bit of what most would have to learn. GitHub, Java, Tomcat, command line and Test-Driven development are all things that I have gained a fair amount of experience in over the years. Fortunately our team got along nicely and we were able to allocate roles and apply other software development techniques which led to a smooth production. It felt very nice to be able to supply my teammates with knowledge as well as learn with them certain things such as the bash

scripting. It was very interesting to learn about the equations for calculating many color properties, as well as learn about all the open source software available for improving people's daily lives whether through accessibility, education or other sorts of the like.

Self-Evaluation

As a team we did a good job. We by no means were anywhere close to perfect, but we were able to work well enough together to accomplish what we set out to do at the beginning of the semester. Everyone contributed to the project in their own ways and we fit the puzzle pieces together. Communication was often tough, and meeting up to work on the project together at points was difficult. Once we all understood how to collaborate over Github it made things way easier and we were able to coordinate most changes on the project using our group message.

Evaluations and Suggestions

We thought the project was fair for the most part so really these suggestions should be taken lightly. In the beginning of the semester if we had had some lessons on scripting to give us some sort of idea of where to start that would have been a boost and saved some time. I noticed a lot of the projects people chose had not been updated in a long time on Github, maybe if we were able to work on something that people had recently edited or contributed to would have been cool to work on an actually help them out via our testing.