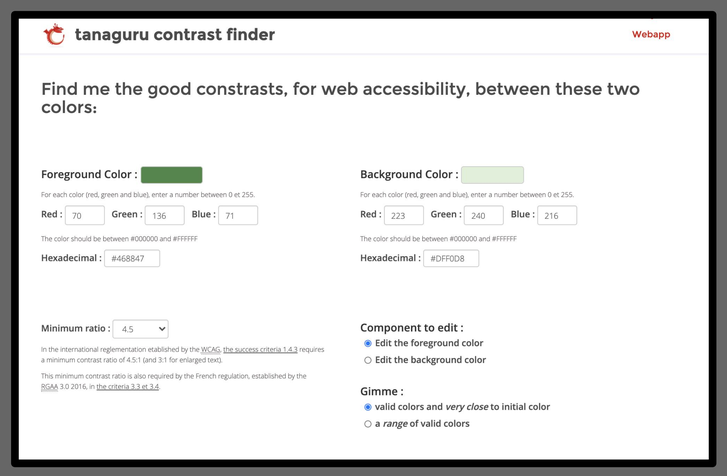
CSCI 362 Software Engineering: Testing Framework of the Tanaguru Contrast Finder

Ethan Graham, Daniel McBane, Chloe Stapleton

Tanaguru

The Tanaguru Contrast Finder is a web-based tool for finding suitable foreground and background colors for ease of use. The Contrast Finder is particularly useful in helping visually impaired readers and complies with the Web Content Accessibility Guidelines presented by the World Wide Web Consortium (W3C).

Our testing framework aims to find bugs and confirm reliability of calculations of the Tanaguru Contrast Finder.

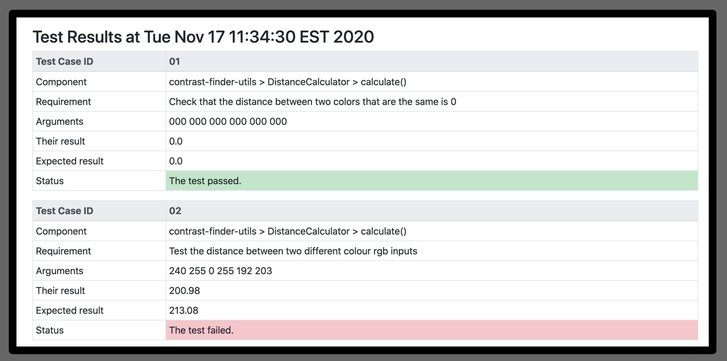


Testing Script

Our automatic testing framework consists of two files “runAllTests.sh” and “runtest.sh”

“runAllTests.sh” calls “runtest.sh” for each file in our test case directory. Before calling “runtest.sh” on each test case it creates an html file and sets up a table displays our test case and whether it passes or fails. Once its finished running “runtest.sh” for each test case it then opens the html file in the browser. Once it has successfully opened it then proceeds to delete the html file so that a new one is created each time it is run.

“runtest.sh” grabs information from the test case text file, it then updates the table in “runAllTests.sh” with the information from the test case. It then proceeds to find and compile the java files needed for the test case from the ../project/src directory where the Tanaguru file is and the ../testCaseExecutables directory where the driver file is. It then proceeds to compile the Tanaguru file in the same directory as the driver file. Once the files are compiled the script runs the driver file with the required inputs taken from the test case text file. When the file has been successfully run and has sent its output to the output.txt file it then cleans up the directories by deleting all the .class files and the output.txt file.



Test Results at Tue Nov 17 11:34:30 EST 2020

Test Case ID	01
Component	contrast-finder-utils > DistanceCalculator > calculate()
Requirement	Check that the distance between two colors that are the same is 0
Arguments	000 000 000 000 000 000
Their result	0.0
Expected result	0.0
Status	The test passed.

Test Case ID	02
Component	contrast-finder-utils > DistanceCalculator > calculate()
Requirement	Test the distance between two different colour rgb inputs
Arguments	240 255 0 255 192 203
Their result	200.98
Expected result	213.08
Status	The test failed.

Test Cases

TestID	Requirement	Component	Method	Inputs	Outcomes
1	Find distance between two colors	DistanceCalculator.java	calculate(color1,color2)	000 000 000, 000 000 000	A value of 0 should be returned
2	Find distance between two colors	DistanceCalculator.java	calculate(color1,color2)	240 255 000, 255 192 203	A value of 213.08 should be returned
3	Find distance between two colors	DistanceCalcultor.java	calculate(color1,color2)	180 180 000, 180 180 060	A value of 60 should be returned
4	Find distance between two colors	DistanceCalculator.java	calculate(color1,color2)	255 255 255, 000 000 000	A value of 441.67 should be returned
5	Find Euclidean distance between two colors	ContrastChecker.java	distanceColor(color1,color2)	000 000 000, 000 000 000	A value of 0 should be returned
6	Find Euclidean distance between two colors	ContrastChecker.java	distanceColor(color1,color2)	240 255 000, 255 192 203	A value of 213.0 should be returned
7	Find Euclidean distance between two colors	ContrastChecker.java	distanceColor(color1,color2)	180 180 000, 180 180 060	A value of 60 should be returned
8	Find Euclidean distance between two colors	ContrastChecker.java	distanceColor(color1,color2)	255 255 255, 000 000 000	A value of 441.67 should be returned
9	Check method gives an Error from invalid input	DistanceCalculator.java	calculate(color1,colour2)	000 000 000, GGG GGG GGG	An Error should be returned
10	Check method gives Error from invalid input	DistanceCalculator.java	calculate(color1,color2)	000 000 000, -111 111 111	An Error should be returned
11	Convert RGB value to Hex value	ColorConverter.java	rgb2Hex(R/G/B Value)	128 000 128	A value of #800080 should be returned
12	Convert RGB value to Hex value	ColorConverter.java	rgb2Hex(R/G/B Value)	128 000 032	A value of #800020
13	Convert RGB value to Hex value	ColorConverter.java	rgb2Hex(R/G/B Value)	015 082 186	A value of #0F52BA should be returned
14	Convert Hex value to RGB value	ColorConverter.java	hex2Rgb(Hex Value)	#800080	java.awt.Color[r=128,g=0,b =128] should be returned
15	Convert Hex value to RGB value	ColorConverter.java	hex2Rgb(Hex Value)	#008000	java.awt.Color[r=0,g=128,b=0] should be returned
16	Check the methods response to non-numeric input	ColorConverter..java	hex2Rgb(Hex Value)	#00G000	An Error is expected
17	Check methods response to non-numeric input	ColorConverter.java	hex2Rgb(Hex Value)	128 000 FFF	An Error is expected
18	Check methods response to negative input	ColorConverter.java	hex2Rgb(Hex Value)	-128 000 000	An Error is expected
19	Check methods response to invalid input	ColorConverter.java	hex2Rgb(Hex Value)	“apple”	An Error is expected
20	Check methods response to invalid input	ColorConverter.java	rgb2Hex(R/G/B Value)	“apple”	An Error is expected
21	Calculate RGB number with offset	ColorConverter.java	offsetRgb(RGB1,RGB2)	200 200 200, 0 0 0	java.awt.Color[r=200,g=200,b=200] should be returned
22	Calculate RGB number with offset	ColorConverter.java	offsetRgb(RGB1,RGB2)	155 211 007, 6 8 25	java.awt.Color[r=161,g=219,b=32] should be returned
23	Calculate RGB number with offset	ColorConverter.java	offsetRgb(RGB1,RGB2)	155 211 007, -60 -8 -2	java.awt.Color[r=95,g=203,b=5] should be returned
24	Offset rgb number	ColorConverter.java	offsetRgb(RGB1,RGB2)	250 190 65, 6 0 0	An Error is expected
25	Offset rgb number	ColorConverter.java	offsetRgb(RGB1,RGB2)	250 190 65, 0 0 0	An Error is expected

Injecting Faults

To make sure our tests are properly working, we inject strategic faults and analyze the results of our automated testing framework.

Fault 1:
In the ColorConverter class we inserted a fault into the offsetRgbColor() method on line 118.
Fault injected: Subtract the blue offset rather than adding it

Fault 2:
In the ContrastChecker class we inserted a fault into the distanceColor(() method on line 66.
Fault injected: Incorrectly multiply red values instead of subtracting them to find the Euclidean distance

Fault 3:
In the ColorConverter class we inserted a fault into the rgbToHex() method on line 189.
Fault injected: Add an extra 0 into the formatter

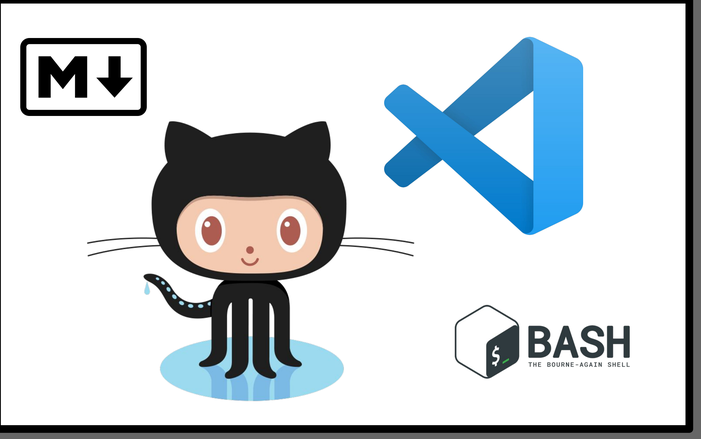
Fault 4:
In the ColorConverter class we inserted a fault into the getNewColor() method on line 166.
Fault injected: Parse the Hex color string incorrectly from R to B instead of R to G

Fault 5:
In the DistanceCalculator class we inserted a fault into the calculate method on line 30.
Fault injected: Set CUBIC variable to 0 instead of 3

With all five faults in code simultaneously, 10/25 test cases will pass.

Lessons Learned

- Bash scripting / parsing text files
- The purpose of a driver
- Github / Git version control
- Markdown
- Java compiling via the command line
- VSCode
- The importance of testing!



Acknowledgements

A special thanks to Dr. Bowring for his guidance and feedback as well as the Tanaguru project and its open source status for the code (found at https://github.com/Tanaguru/Contrast-Finder).