Deliverable 2

Test Plan

The Testing Process

Testing will be carried out by running a runAllTests script from the linux command line, which will walk through the testCases folder and use the specification files to execute each test specific case. This will be displayed, and outputted to a HTML file comparing the results to what is expected.

Requirements Traceability

The Tanaguru software should be able to properly calculate the contrast between two colors, as well as produce a more favorable color combination if needed.

DistanceCalculator.java

calculate(Color colorToChange, Color colorToKeep)

The calculate method takes two java Color objects as input and then, using each's RGB values, finds the Euclidean distance between them.

ContrastChecker.java

getConstrastRatio(final Color fgColor, final Color bgColor)

The getConstrastRatio method takes two Color objects as input and then calculates the ratio between them, using the lighter color as the first in the calculation. The typo is from the original code.

ColorConverter.java

getHue(Color color)

The getHue method takes one Color object as input and then returns the calculated value for the hue.

offsetRgbColor(Color bgColor, int offsetRed, int offsetGreen, int offsetBlue)

The offsetRgbColor method takes as input a Color object and three ints representing the amount to offset the respective RGB values. This new Color is returned.

Tested Items

DistanceCalculator.java

calculate(Color colorToChange, Color colorToKeep)

Our tests include having the two far ends of color, white and black, as well as having the higher color in both the first and second input, having the same color as both inputs, and two tests of different colors in between the range of RGB.

ContrastChecker.java

getConstrastRatio(final Color fgColor, final Color bgColor)

Our tests include having the two far ends of color, white and black, as well as having the higher color in both the first and second input, having the same color as both inputs, and two tests of different colors in between the range of RGB.

ColorConverter.java

getHue(Color color)

Our tests include various RGB values between 0 and 255 with varying levels of each value.

offsetRgbColor(Color bgColor, int offsetRed, int offsetGreen, int offsetBlue)

Our tests include a variety of RGB values as well as various offset values including positive and negative.

rgb2Hex(Color color)

Our tests include a variety of RGB values including those who decimal and hexadecimal representations are the same and those that are different.

Testing Schedule

Our project schedule followed along with the set deadlines of the course throughout its completion.

September 22, 2020	Project Choice Made
October 6, 2020	Test Plan Created
November 5, 2020	Testing Framework Created
November 17, 2020	All Tests Implemented
November 24, 2020	Faults Injected into Code
December 3, 2020	Final Report Completed

Test Recording Procedures

When the testing framework is invoked, a html file will be created and opened in a browser to display the results of the tests in a readable format.

Hardware and Software Requirements

The testing framework is run from the Linux command line. Python and Java are both required. Both newer and older versions of each work.

Test Case Specification Template

The following is the format in which all test cases in our framework are to follow for implementation. The last two lines only appear if the project has been run before as it is the location the results of the test are stored.

```
1. Test ID
2. Tested class
3. Tested Method and Parameters
4. Test Case Driver
5. Arguments
6. Expected outcomes
7. Last Result from Running Test (These last two lines only appear if
the tests have been run before)
8. Success or Fail
```

All java test case executables must follow this order of arguments for commands for the script to properly use them:

```
TestMethodArguments ExpectedResults TestCaseFilePath
```

Where TestMethodArguments are taken from line 5 of the test case, ExpectedResults are taken from line 6, and TestCaseFilePath is a direct path to the test case for TestOutput.java to use to record the results.

Sample Output and Test Case Specification

Tanaguru TestAutomation Results

Test ID	Tested class	Tested Method and Parameters	Driver	Arguments	Expected outcomes	Last Result from Running Test	Success or Fail
calculate_1	DistanceCalculator.java	calculate(Color(255,255,255), Color(0,0,0))	testcasesexecutables.TestCalculate	255 255 255 0 0 0	441.67	367.77	Fail
calculate_2	DistanceCalculator.java	calculate(Color(0,0,0), Color(255,255,255))	testcasesexecutables.TestCalculate	0 0 0 255 255 255	441.67	367.77	Fail
calculate_3	DistanceCalculator.java	calculate(Color(0,0,0), Color(0,0,0))	testcasesexecutables.TestCalculate	0 0 0 0 0 0	0.0	0.0	Pass
calculate_4	DistanceCalculator.java	calculate(Color(255,200,34),Color(255,255,17))	testcasesexecutables.TestCalculate	255 200 34 255 255 17	57.57	54.45	Fail
calculate_5	DistanceCalculator.java	calculate(Color(200,7,16),Color(57,57,57))	testcasesexecutables.TestCalculate	200 7 16 57 57 57	156.94	139.77	Fail
getConstrastRatio_1	ContrastChecker.java	getConstrastRatio(Color(0,0,0),Color(0,0,0))	testcasesexecutables.TestGetConstrastRatio	0 0 0 0 0 0	1.0	1.0	Pass
getConstrastRatio_2	ContrastChecker.java	getConstrastRatio(Color(0,0,0),Color(255,255,255))	testcasesexecutables.TestGetConstrastRatio	0 0 0 255 255 255	21.0	21.0	Pass
getConstrastRatio_3	ContrastChecker.java	getConstrastRatio(Color(255,255,255),Color(0,0,0))	testcasesexecutables.TestGetConstrastRatio	255 255 255 0 0 0	21.0	21.0	Pass
getConstrastRatio_4	ContrastChecker.java	getConstrastRatio(Color(57,57,57),Color(213,247,156))	testcasesexecutables.TestGetConstrastRatio	57 57 57 213 247 156	9.686825265872823	9.686825265872823	Pass
getConstrastRatio_5	ContrastChecker.java	getConstrastRatio(Color(187,5,190),Color(68,106,159))	testcasesexecutables.TestGetConstrastRatio	187 5 190 68 106 159	1.0184255955300985	1.0184255955300985	Pass
getHue_1	ColorConverter.java	getHue(Color(128,128,0))	testcasesexecutables.TestGetHue	128 128 0	60.0	60.0	Pass
getHue_2	ColorConverter.java	getHue(Color(0,0,0))	testcasesexecutables.TestGetHue	0 0 0	0.0	0.0	Pass
getHue_3	ColorConverter.java	getHue(Color(45,50,255))	testcasesexecutables.TestGetHue	45 50 255	238.57144	238.57144	Pass
getHue_4	ColorConverter.java	getHue(Color(200,255,0))	testcasesexecutables.TestGetHue	200 255 0	72.94118	72.94118	Pass
getHue_5	ColorConverter.java	getHue(Color(2,0,241))	testcasesexecutables.TestGetHue	2 0 241	240.49792	240.49792	Pass
offsetRgbColor_1	ColorConverter.java	offsetRgbColor(Color(128,128,0), 1, 2, 3)	testcasesexecutables.TestOffsetRGBColor	128 128 0 1 2 3	129 130 3	129 130 3	Pass
offsetRgbColor_2	ColorConverter.java	offsetRgbColor(Color(0,255,255), 255, -10, -10)	testcasesexecutables.TestOffsetRGBColor	0 255 255 255 -10 -10	255 245 245	255 245 245	Pass
offsetRgbColor_3	ColorConverter.java	offsetRgbColor(Color(120, 120, 120), 60, -60, 60)	testcasesexecutables.TestOffsetRGBColor	120 120 120 60 -60 60	180 60 180	180 60 180	Pass
offsetRgbColor_4	ColorConverter.java	offsetRgbColor(Color(0, 0, 0), 255, 255, 255)	testcasesexecutables.TestOffsetRGBColor	0 0 0 255 255 255	255 255 255	255 255 255	Pass
offsetRgbColor_5	ColorConverter.java	offsetRgbColor(Color(255,245,245), -255, -245, -245)	testcasesexecutables.TestOffsetRGBColor	255 245 245 -255 -245 -245	0 0 0	0 0 0	Pass
rgb2Hex_1	ColorConverter.java	rgb2Hex(Color(0,1,2))	testcasesexecutables.TestRGB2Hex	0 1 2	000102	000102	Pass
rgb2Hex_2	ColorConverter.java	rgb2Hex(Color(128,128,12))	testcasesexecutables.TestRGB2Hex	128 128 12	80800C	80800C	Pass
rgb2Hex_3	ColorConverter.java	rgb2Hex(Color(0,255,0))	testcasesexecutables.TestRGB2Hex	0 255 0	00FF00	00FF00	Pass
rgb2Hex_4	ColorConverter.java	rgb2Hex(Color(0,255,255))	testcasesexecutables.TestRGB2Hex	0 255 255	00FFFF	00FFFF	Pass
rgb2Hex_5	ColorConverter.java	rgb2Hex(Color(255,255,255))	testcasesexecutables.TestRGB2Hex	255 255 255	FFFFFF	FFFFFF	Pass

For a closer and more in-depth look, visit our github and look at the individual test case files or the sample output documents.

Report

When looking through the Tanaguru source code, it became ever evident that documentation was lacking. Luckily we were able to discern the purpose for things with what little we had along with tracing the code. Some examples of this are when making the test cases for the calculate method, we found that what the documentation says the method returns is incorrect. It says it uses the distance formula but the code itself uses cubic operations rather than squared. Because of this we decided to base our expected outcomes on the Euclidean formula, even though that means tests will fail. And the getConstrastRatio has one of multiple typos seen throughout the code. This makes our testing framework all the more useful.