The Boyoz: Tanaguru Contrast-Finder

Automated Testing Suite

By Nathan Bell, Paul Joseph, and Logan Sitar

Choosing a Project

- After doing research, we narrowed our search down to OpenMRS and Tanaguru
- At first we choose OpenMRS because of the success we had deploying the code, yet we were having trouble finding meaningful methods to test.
- After finding out code deployment was not required, we settled on Tanaguru merely because the methods were easier to test

OpenMRS

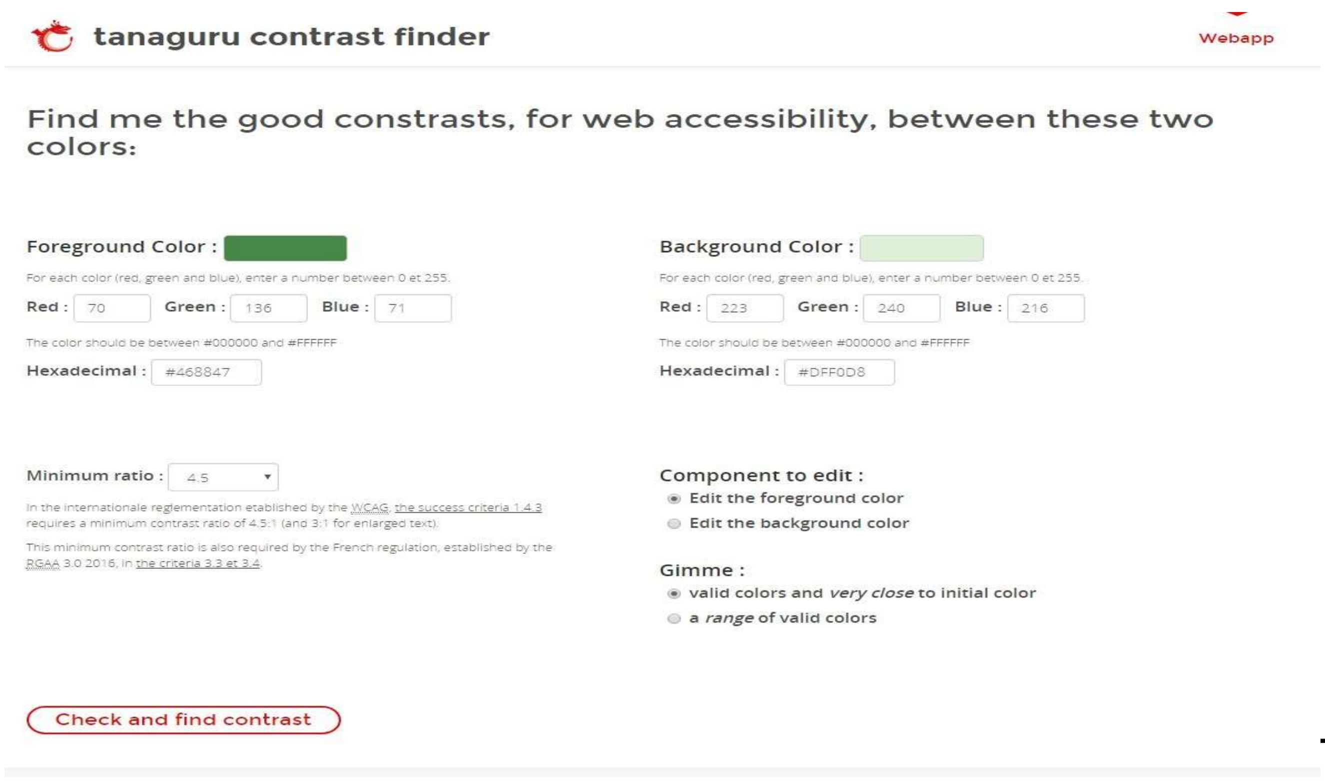
What is OpenMRS?

- OpenMRS is a global community of truly dedicated, talented, and generous contributors who build and maintain the OpenMRS platform and other, foundational OpenMRS technical products.
- It is a platform that countries and implementers use to create a customized EMR system in response to actual needs on the ground.

What is Tanaguru?

- Tanaguru is an open source website assessment tool that is dedicated to enhancing web quality and accessibility. They also focus on reliability and automation.
- It finds the best contrasts between two colors so a website is easily readable

Tanaguru Example


tanaguru contrast finder

Webapp

Find me the good constrasts, for web accessibility, between these two colors:

Foreground Color : 🟩

For each color (red, green and blue), enter a number between 0 et 255.

Red : 70 Green : 136 Blue : 71

The color should be between #000000 and #FFFFFF

Hexadecimal : #468847

Background Color : 🟩

For each color (red, green and blue), enter a number between 0 et 255.

Red : 223 Green : 240 Blue : 216

The color should be between #000000 and #FFFFFF

Hexadecimal : #DFF0D8

Minimum ratio : 4.5

In the internationale reglementation established by the WCAG, the success criteria 1.4.3 requires a minimum contrast ratio of 4.5:1 (and 3:1 for enlarged text).

This minimum contrast ratio is also required by the French regulation, established by the RGAA 3.0 2016, in the criteria 3.3 et 3.4.

Component to edit :

● Edit the foreground color

○ Edit the background color

Gimme :

● valid colors and *very close* to initial color

○ a *range* of valid colors

Check and find contrast

Test Case Outline

Example Test Case

Id:

Requirement:

Class:

Method:

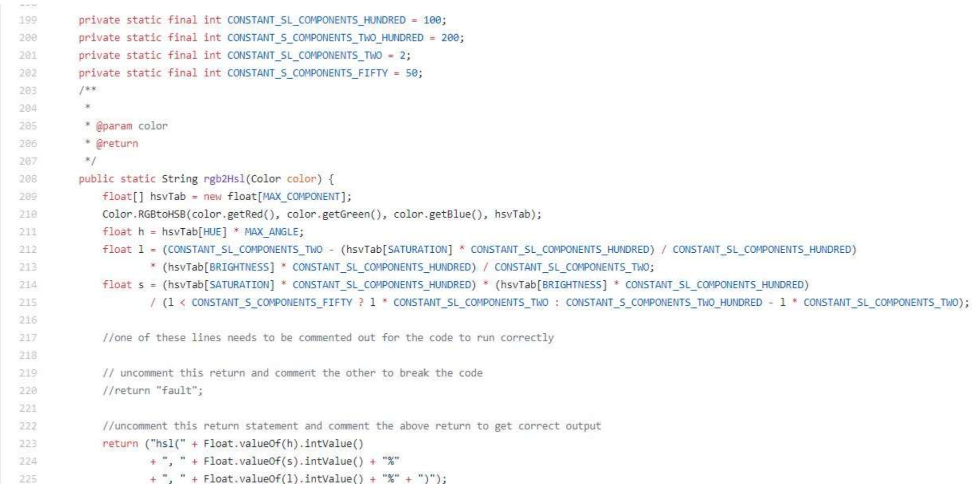
Input:

Expected Output:

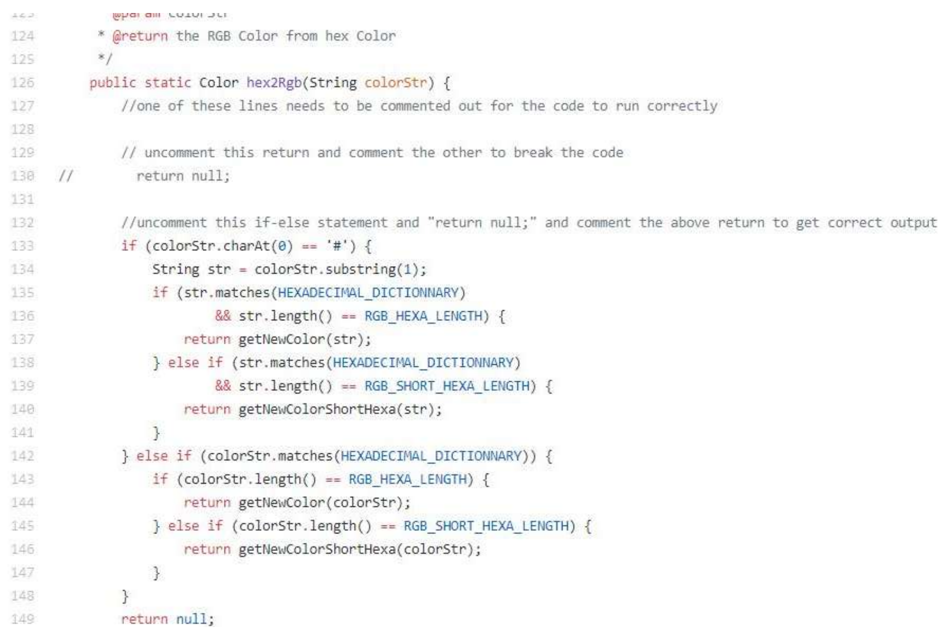
Class & Method Selection

- ColorConverter.java
- ContrastChecker.java
- DistanceCalculator.java

ColorConverter.java

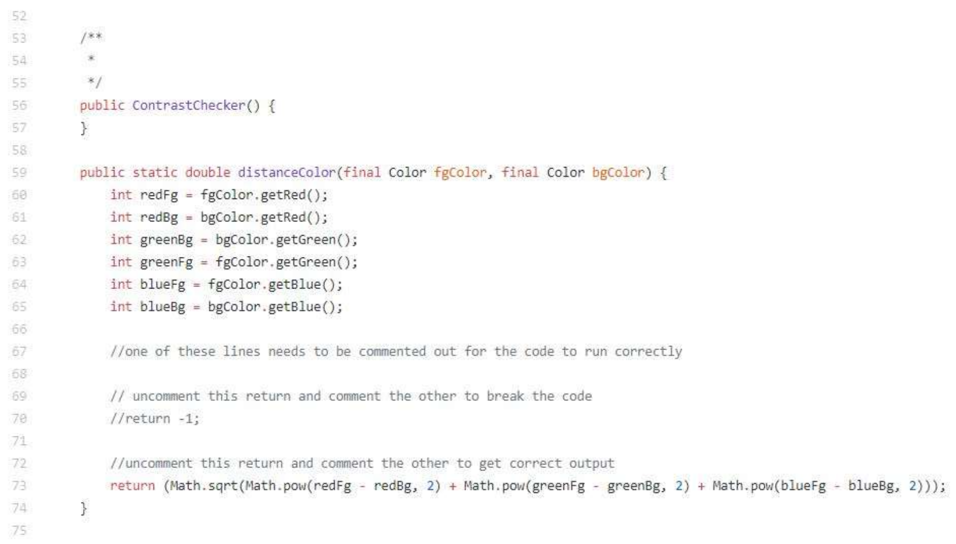


```java
    private static final int CONSTANT_SL_COMPONENTS_HUNDRED = 100;
    private static final int CONSTANT_S_COMPONENTS_TWO_HUNDRED = 200;
    private static final int CONSTANT_SL_COMPONENTS_TWO = 2;
    private static final int CONSTANT_S_COMPONENTS_FIFTY = 50;
    /**
     *
     * @param color
     * @return
     */
    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() + "%" + ")");
```

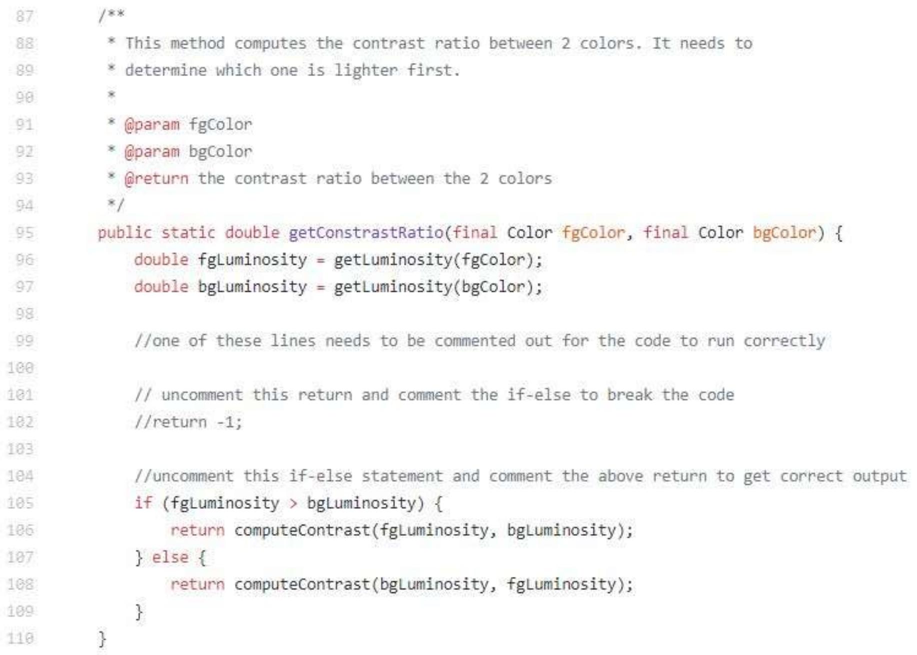


```java
     * @return the RGB Color from hex Color
     */
    public static Color hex2Rgb(String colorStr) {
        //one of these lines needs to be commented out for the code to run correctly

        // 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_HEXA_LENGTH) {
                return getNewColor(str);
            } else if (str.matches(HEXADECIMAL_DICTIONNARY)
                    && str.length() == RGB_SHORT_HEXA_LENGTH) {
                return getNewColorShortHexa(str);
            }
        } else if (colorStr.matches(HEXADECIMAL_DICTIONNARY)) {
            if (colorStr.length() == RGB_HEXA_LENGTH) {
                return getNewColor(colorStr);
            } else if (colorStr.length() == RGB_SHORT_HEXA_LENGTH) {
                return getNewColorShortHexa(colorStr);
            }
        }
        return null;
```

ContrastChecker.java

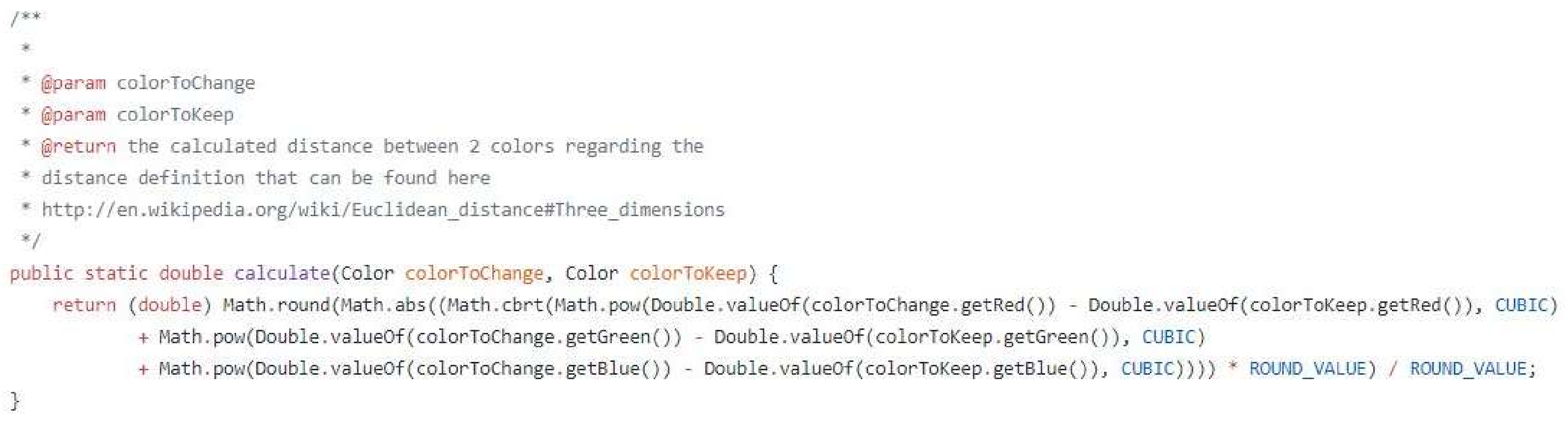


```java
/**
 *
 */
public ContrastChecker() {
}

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)));
}
```



```java
/**
 * This method computes the contrast ratio between 2 colors. It needs to
 * determine which one is lighter first.
 *
 * @param fgColor
 * @param bgColor
 * @return the contrast ratio between the 2 colors
 */
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);
    }
}
```

DistanceCalculator.java



```java
/**
 *
 * @param colorToChange
 * @param colorToKeep
 * @return the calculated distance between 2 colors regarding the
 * distance definition that can be found here
 * http://en.wikipedia.org/wiki/Euclidean_distance#Three_dimensions
 */
public static double calculate(Color colorToChange, Color colorToKeep) {
    return (double) Math.round(Math.abs((Math.cbrt(Math.pow(Double.valueOf(colorToChange.getRed()) - Double.valueOf(colorToKeep.getRed()), CUBIC)
            + Math.pow(Double.valueOf(colorToChange.getGreen()) - Double.valueOf(colorToKeep.getGreen()), CUBIC)
            + Math.pow(Double.valueOf(colorToChange.getBlue()) - Double.valueOf(colorToKeep.getBlue()), CUBIC)))) * ROUND_VALUE) / ROUND_VALUE;
}
```

Requirements Traceability

distanceColor() - Calculates distance between two colors

getConstrastRatio() - Finds the contrast of an inputted color object

rgb2Hsl() - Converts a color object to a HSL value

hex2Rgb() - Converts hexadecimal value to a RGB value

calculate() - Calculates Euclidian distance between two colors

Test Case Changes

TestCase1.txt

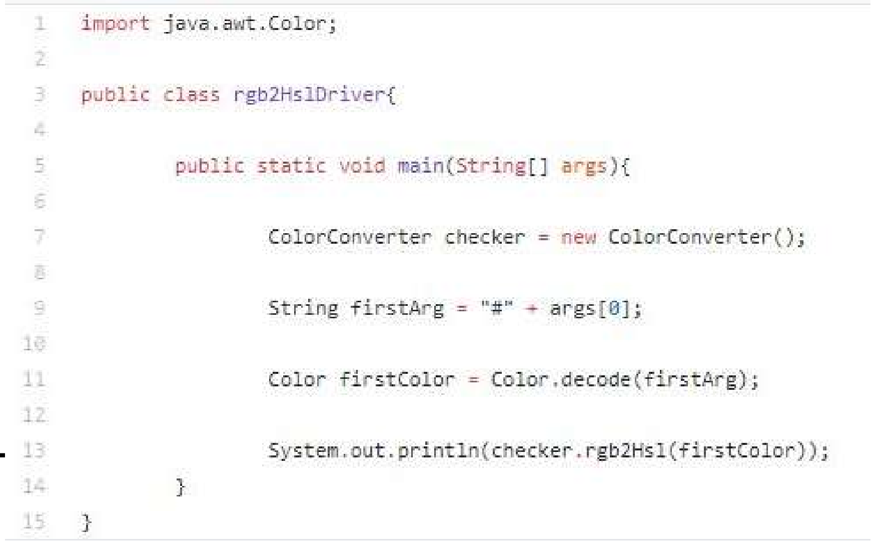
```
1
Takes a color object and outputs a string of the HSL value (hue, saturation, lightness)
ColorConverter.java
rgb2Hsl
0,0,0
“hsl(0.0, 0.0%, 0.0%)”
```

testCase_01.txt

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

Drivers

- calculateDriver.java
- distanceColorDriver.java
- getContrastRatioDriver.java
- hex2RgbDriver.java
- rgb2HslDriver.java



```java
import java.awt.Color;

public class rgb2HslDriver{

        public static void main(String[] args){

                ColorConverter checker = new ColorConverter();

                String firstArg = "#" + args[0];

                Color firstColor = Color.decode(firstArg);

                System.out.println(checker.rgb2Hsl(firstColor));
        }
}
```

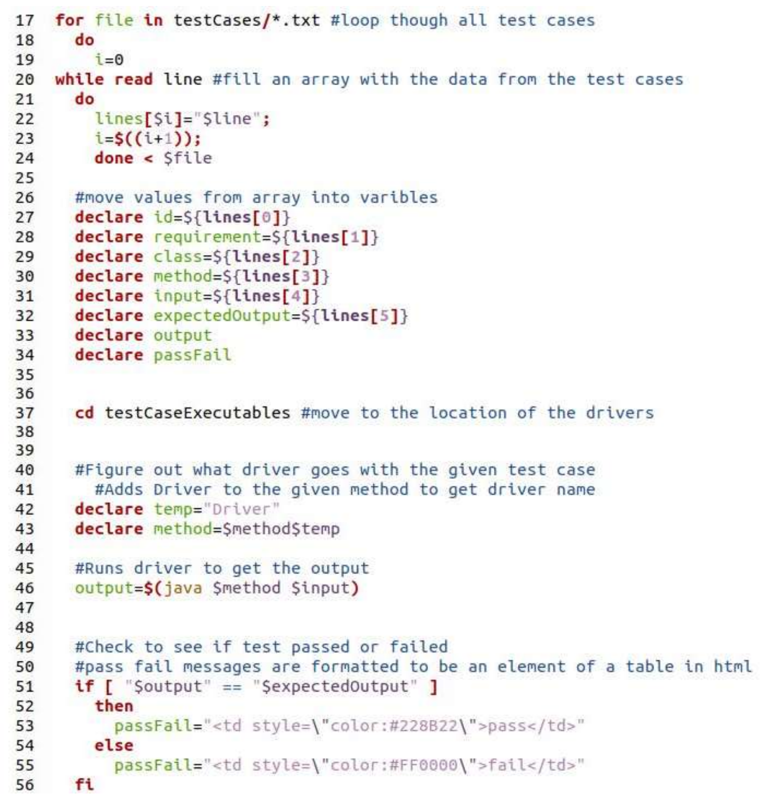
Script

For each test case file

Read each line into a

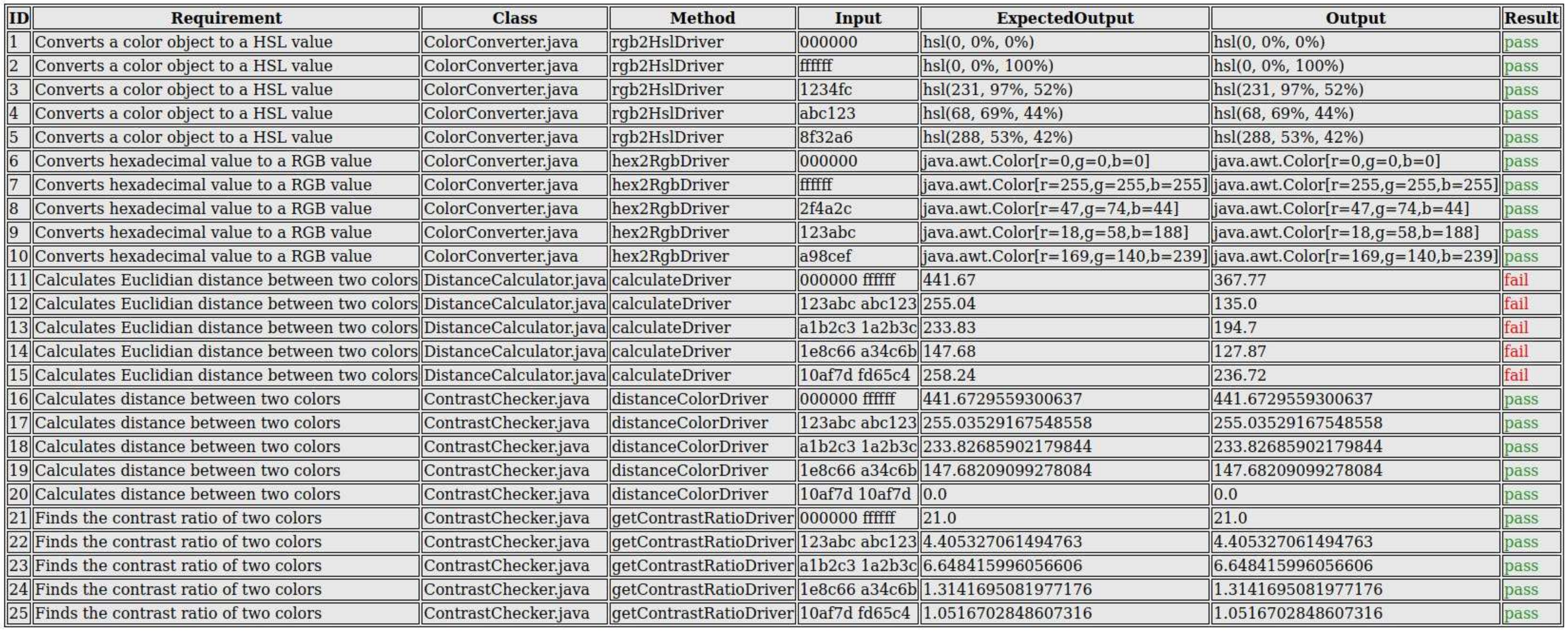
Run the driver with the input

Check it it matches output



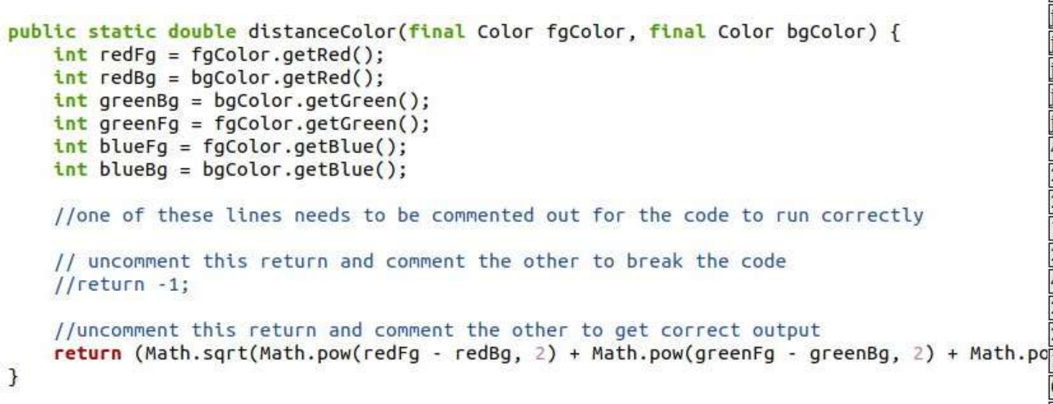
```
for file in testCases/*.txt #loop though all test cases
  do
    i=0
while read line #fill an array with the data from the test cases
  do
    lines[$i]="$line";
    i=$((i+1));
    done < $file

  #move values from array into varibles
  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


  cd testCaseExecutables #move to the location of the drivers


  #Figure out what driver goes with the given test case
    #Adds Driver to the given method to get driver name
  declare temp="Driver"
  declare method=$method$temp

  #Runs driver to get the output
  output=$(java $method $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
```

Output

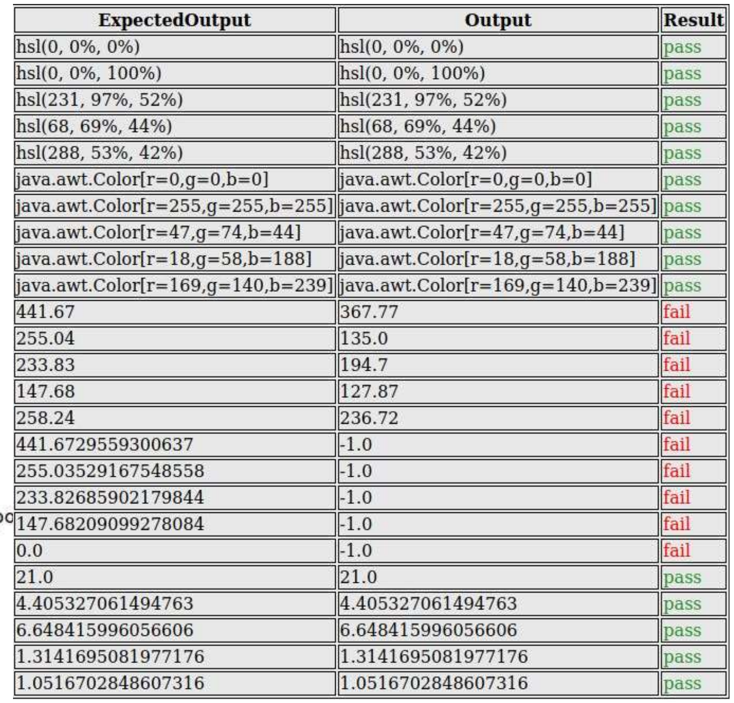


ID	Requirement	Class	Method	Input	ExpectedOutput	Output	Result
1	Converts a color object to a HSL value	ColorConverter.java	rgb2HslDriver	000000	hsl(0, 0%, 0%)	hsl(0, 0%, 0%)	pass
2	Converts a color object to a HSL value	ColorConverter.java	rgb2HslDriver	ffffff	hsl(0, 0%, 100%)	hsl(0, 0%, 100%)	pass
3	Converts a color object to a HSL value	ColorConverter.java	rgb2HslDriver	1234fc	hsl(231, 97%, 52%)	hsl(231, 97%, 52%)	pass
4	Converts a color object to a HSL value	ColorConverter.java	rgb2HslDriver	abc123	hsl(68, 69%, 44%)	hsl(68, 69%, 44%)	pass
5	Converts a color object to a HSL value	ColorConverter.java	rgb2HslDriver	8f32a6	hsl(288, 53%, 42%)	hsl(288, 53%, 42%)	pass
6	Converts hexadecimal value to a RGB value	ColorConverter.java	hex2RgbDriver	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	hex2RgbDriver	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	hex2RgbDriver	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	hex2RgbDriver	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	hex2RgbDriver	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	calculateDriver	000000 ffffff	441.67	367.77	fail
12	Calculates Euclidian distance between two colors	DistanceCalculator.java	calculateDriver	123abc abc123	255.04	135.0	fail
13	Calculates Euclidian distance between two colors	DistanceCalculator.java	calculateDriver	a1b2c3 1a2b3c	233.83	194.7	fail
14	Calculates Euclidian distance between two colors	DistanceCalculator.java	calculateDriver	1e8c66 a34c6b	147.68	127.87	fail
15	Calculates Euclidian distance between two colors	DistanceCalculator.java	calculateDriver	10af7d fd65c4	258.24	236.72	fail
16	Calculates distance between two colors	ContrastChecker.java	distanceColorDriver	000000 ffffff	441.6729559300637	441.6729559300637	pass
17	Calculates distance between two colors	ContrastChecker.java	distanceColorDriver	123abc abc123	255.03529167548558	255.03529167548558	pass
18	Calculates distance between two colors	ContrastChecker.java	distanceColorDriver	a1b2c3 1a2b3c	233.82685902179844	233.82685902179844	pass
19	Calculates distance between two colors	ContrastChecker.java	distanceColorDriver	1e8c66 a34c6b	147.68209099278084	147.68209099278084	pass
20	Calculates distance between two colors	ContrastChecker.java	distanceColorDriver	10af7d 10af7d	0.0	0.0	pass
21	Finds the contrast ratio of two colors	ContrastChecker.java	getContrastRatioDriver	000000 ffffff	21.0	21.0	pass
22	Finds the contrast ratio of two colors	ContrastChecker.java	getContrastRatioDriver	123abc abc123	4.405327061494763	4.405327061494763	pass
23	Finds the contrast ratio of two colors	ContrastChecker.java	getContrastRatioDriver	a1b2c3 1a2b3c	6.648415996056606	6.648415996056606	pass
24	Finds the contrast ratio of two colors	ContrastChecker.java	getContrastRatioDriver	1e8c66 a34c6b	1.3141695081977176	1.3141695081977176	pass
25	Finds the contrast ratio of two colors	ContrastChecker.java	getContrastRatioDriver	10af7d fd65c4	1.0516702848607316	1.0516702848607316	pass

Error Insertion

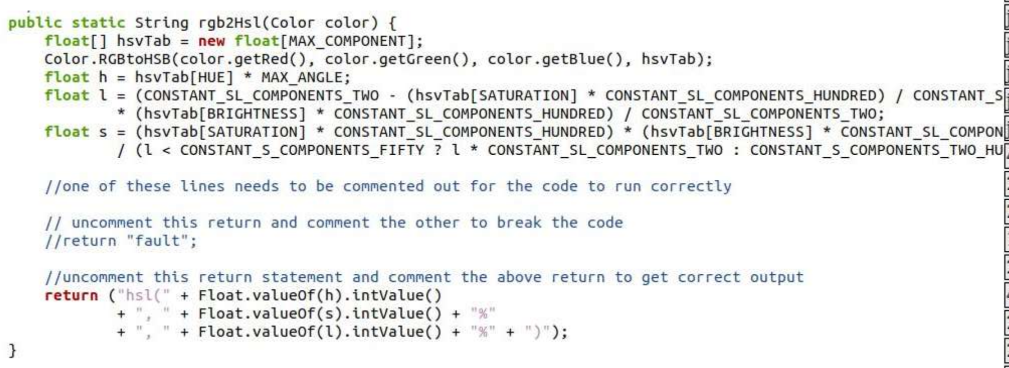


```
public static double distanceColor(final Color fgColor, final Color bgColor) {
    int redFg = fgColor.getRed();
    int redBg = bgColor.getRed();
    int greenBg = bgColor.getGreen();
    int greenFg = fgColor.getGreen();
    int blueFg = fgColor.getBlue();
    int blueBg = bgColor.getBlue();

    //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.po
}
```

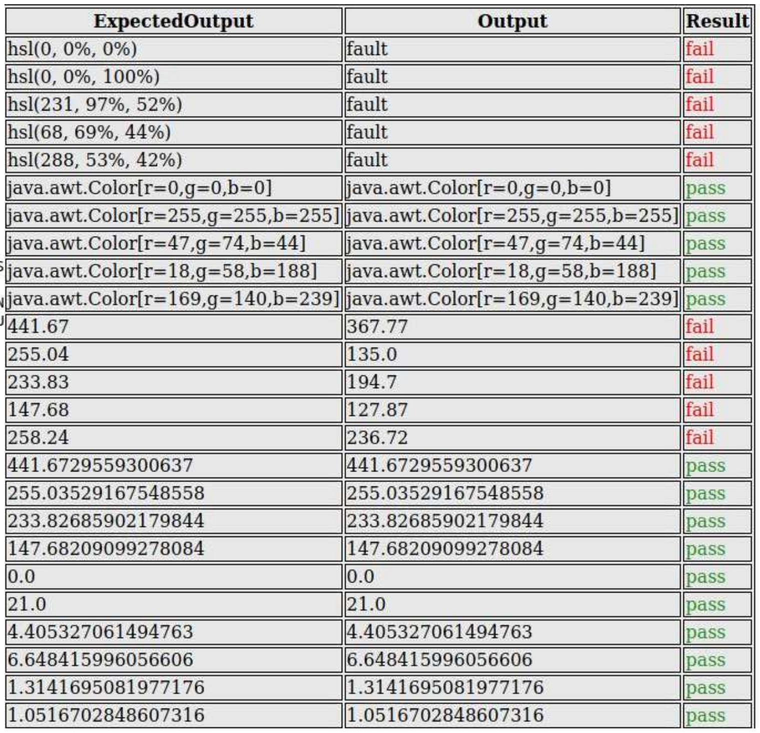


ExpectedOutput	Output	Result
hsl(0, 0%, 0%)	hsl(0, 0%, 0%)	pass
hsl(0, 0%, 100%)	hsl(0, 0%, 100%)	pass
hsl(231, 97%, 52%)	hsl(231, 97%, 52%)	pass
hsl(68, 69%, 44%)	hsl(68, 69%, 44%)	pass
hsl(288, 53%, 42%)	hsl(288, 53%, 42%)	pass
java.awt.Color[r=0,g=0,b=0]	java.awt.Color[r=0,g=0,b=0]	pass
java.awt.Color[r=255,g=255,b=255]	java.awt.Color[r=255,g=255,b=255]	pass
java.awt.Color[r=47,g=74,b=44]	java.awt.Color[r=47,g=74,b=44]	pass
java.awt.Color[r=18,g=58,b=188]	java.awt.Color[r=18,g=58,b=188]	pass
java.awt.Color[r=169,g=140,b=239]	java.awt.Color[r=169,g=140,b=239]	pass
441.67	367.77	fail
255.04	135.0	fail
233.83	194.7	fail
147.68	127.87	fail
258.24	236.72	fail
441.6729559300637	-1.0	fail
255.03529167548558	-1.0	fail
233.82685902179844	-1.0	fail
147.68209099278084	-1.0	fail
0.0	-1.0	fail
21.0	21.0	pass
4.405327061494763	4.405327061494763	pass
6.648415996056606	6.648415996056606	pass
1.3141695081977176	1.3141695081977176	pass
1.0516702848607316	1.0516702848607316	pass

Error Insertion continued



```
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_S
            * (hsvTab[BRIGHTNESS] * CONSTANT_SL_COMPONENTS_HUNDRED) / CONSTANT_SL_COMPONENTS_TWO;
    float s = (hsvTab[SATURATION] * CONSTANT_SL_COMPONENTS_HUNDRED) * (hsvTab[BRIGHTNESS] * CONSTANT_SL_COMPON
            / (l < CONSTANT_S_COMPONENTS_FIFTY ? l * CONSTANT_SL_COMPONENTS_TWO : CONSTANT_S_COMPONENTS_TWO_HU

    //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() + "%" + ")");
}
```



ExpectedOutput	Output	Result
hsl(0, 0%, 0%)	fault	fail
hsl(0, 0%, 100%)	fault	fail
hsl(231, 97%, 52%)	fault	fail
hsl(68, 69%, 44%)	fault	fail
hsl(288, 53%, 42%)	fault	fail
java.awt.Color[r=0,g=0,b=0]	java.awt.Color[r=0,g=0,b=0]	pass
java.awt.Color[r=255,g=255,b=255]	java.awt.Color[r=255,g=255,b=255]	pass
java.awt.Color[r=47,g=74,b=44]	java.awt.Color[r=47,g=74,b=44]	pass
java.awt.Color[r=18,g=58,b=188]	java.awt.Color[r=18,g=58,b=188]	pass
java.awt.Color[r=169,g=140,b=239]	java.awt.Color[r=169,g=140,b=239]	pass
441.67	367.77	fail
255.04	135.0	fail
233.83	194.7	fail
147.68	127.87	fail
258.24	236.72	fail
441.6729559300637	441.6729559300637	pass
255.03529167548558	255.03529167548558	pass
233.82685902179844	233.82685902179844	pass
147.68209099278084	147.68209099278084	pass
0.0	0.0	pass
21.0	21.0	pass
4.405327061494763	4.405327061494763	pass
6.648415996056606	6.648415996056606	pass
1.3141695081977176	1.3141695081977176	pass
1.0516702848607316	1.0516702848607316	pass

Final Thoughts

Learned valuable information

- Working with the terminal
- Scripting etc…

Practiced real world habits

- Meeting deadlines
- Providing updates on work
- Working as a collaborative team