
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
-



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



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

Foreground Color : 

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

Red : Green : Blue :

The color should be between #000000 and #FFFFFF

Hexadecimal :

Minimum ratio :

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 RGAA 3.3 et 3.6

Background Color : 

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

Red : Green : Blue :

The color should be between #000000 and #FFFFFF

Hexadecimal :

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

```
199 private static final int CONSTANT_SL_COMPONENTS_HUNDRED = 100;
200 private static final int CONSTANT_S_COMPONENTS_TWO_HUNDRED = 200;
201 private static final int CONSTANT_SL_COMPONENTS_TWO = 2;
202 private static final int CONSTANT_S_COMPONENTS_FIFTY = 50;
203 /**
204  *
205  * @param color
206  * @return
207  */
208 public static String rgb2Hex(String color) {
209     Color.RGBtoRGB16f(color.getRed(), color.getGreen(), color.getBlue(), hsvTab);
210     float h = hsvTab[0] * 100;
211     float r = (CONSTANT_SL_COMPONENTS_TWO - (hsvTab[1] * 100)) / CONSTANT_SL_COMPONENTS_HUNDRED;
212     float g = (CONSTANT_SL_COMPONENTS_TWO - (hsvTab[2] * 100)) / CONSTANT_SL_COMPONENTS_HUNDRED;
213     float b = (CONSTANT_SL_COMPONENTS_TWO - (hsvTab[3] * 100)) / CONSTANT_SL_COMPONENTS_HUNDRED;
214     float s = (hsvTab[4] * 100) / (1 < CONSTANT_SL_COMPONENTS_FIFTY ? 1 * CONSTANT_SL_COMPONENTS_TWO : CONSTANT_S_COMPONENTS_TWO_HUNDRED - 1 * CONSTANT_SL_COMPONENTS_TWO);
215     //One of these lines needs to be commented out for the code to run correctly
216     //uncomment this return and comment the other to break the code
217     //return null;
218     //return "fail";
219     //uncomment this return statement and comment the above return to get correct output
220     return ("hsl(" + Float.valueOf(h).intValue()
221         + ", " + Float.valueOf(s).intValue() + "%"
222         + ", " + Float.valueOf(1).intValue() + "%" + ")");
223 }
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```


ContrastChecker.java

```
52  /**
53   *
54   *
55   */
56  public ContrastChecker() {
57  }
58
59  public static double distanceColor(final Color fgColor, final Color bgColor) {
60      int redFg = fgColor.getRed();
61      int redBg = bgColor.getRed();
62      int greenBg = bgColor.getGreen();
63      int greenFg = fgColor.getGreen();
64      int blueFg = fgColor.getBlue();
65      int blueBg = bgColor.getBlue();
66
67      //one of these lines needs to be commented out for the code to run correctly
68
69      // uncomment this return and comment the other to break the code
70      //return -1;
71
72      //uncomment this return and comment the other to get correct output
73      return (Math.sqrt(Math.pow(redFg - redBg, 2) + Math.pow(greenFg - greenBg, 2) + Math.pow(blueFg - blueBg, 2)));
74  }
75
76
77  /**
78   * This method computes the contrast ratio between 2 colors. It needs to
79   * determine which one is lighter first.
80   *
81   * @param fgColor
82   * @param bgColor
83   * @return the contrast ratio between the 2 colors
84   */
85  public static double getContrastRatio(final Color fgColor, final Color bgColor) {
86      double fgluminosity = getLuminosity(fgColor);
87      double bgLuminosity = getLuminosity(bgColor);
88
89      //one of these lines needs to be commented out for the code to run correctly
90
91      // uncomment this return and comment the if-else to break the code
92      //return -1;
93
94      //uncomment this if-else statement and comment the above return to get correct output
95      if (fgLuminosity > bgLuminosity) {
96          return computeContrast(fgLuminosity, bgLuminosity);
97      } else {
98          return computeContrast(bgLuminosity, fgLuminosity);
99      }
100  }
```

DistanceCalculator.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

```
1 import java.awt.Color;
2
3 public class rgb2HslDriver{
4
5     public static void main(String[] args){
6
7         ColorConverter checker = new ColorConverter();
8
9         String firstArg = "#" + args[0];
10
11         Color firstColor = Color.decode(firstArg);
12
13         System.out.println(checker.rgb2Hsl(firstColor));
14     }
15 }
```

Script

For each test case file

Read each line into a

Run the driver with the
input

Check it it matches output

```
17 for file in testCases/*.txt #loop though all test cases
18 do
19     i=0
20     while read line #fill an array with the data from the test cases
21     do
22         lines[i]="${line}";
23         i=$((i+1));
24         done < $file
25
26     #move values from array into variables
27     declare id=${lines[0]}
28     declare requirement=${lines[1]}
29     declare class=${lines[2]}
30     declare method=${lines[3]}
31     declare input=${lines[4]}
32     declare expectedOutput=${lines[5]}
33     declare output
34     declare passFail
35
36
37     cd testCaseExecutables #move to the location of the drivers
38
39
40     #Figure out what driver goes with the given test case
41     #Adds Driver to the given method to get driver name
42     declare temp="Driver"
43     declare method=$methodTemp
44
45     #Runs driver to get the output
46     output=$(java $method $input)
47
48
49     #Check to see if test passed or failed
50     #pass fail messages are formatted to be an element of a table in html
51     if [ "$output" == "$expectedOutput" ]
52     then
53         passFail="<td style=\"color:#228B22\">pass</td>"
54     else
55         passFail="<td style=\"color:#FF0000\">fail</td>"
56     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	2fa2c	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.pow(blueFg - blueBg, 2))
}
```

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_SL_COMPONENTS_TWO)
    * (hsvTab[BRIGHTNESS] * CONSTANT_SL_COMPONENTS_HUNDRED) / (hsvTab[BRIGHTNESS] * CONSTANT_SL_COMPONENTS_TWO +
    (l < CONSTANT_S_COMPONENTS_FIFTY ? l * CONSTANT_SL_COMPONENTS_TWO : CONSTANT_S_COMPONENTS_TWO_HUNDRED) * 441.67
    //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
