Unit Tests

HarmonyGenerator.java Equivalence Tests:

A harmony can be generated for any color. Colors have three components: Hue, Saturation, and Value. The valid hue range of a color is 0 through 360, inclusive. Saturation and value can be 0.0 through 1.0, inclusive.

1. The analogous scheme only modifies the hue. The first test can be a color with a hue value of 180. Analogous colors should be generated both above and below the chosen color's hue.
2. The second test should be a color near the top of the hue range, such as 350, so that some of the generated analogous colors wrap around from 360 to 0.
3. The third test should be a color near the bottom of the hue range, such as 10, so that some of the generated analogous colors wrap around from 0 to 360.
4. Split-Complementary and the two Evenly Spaced schemes also only modify a color's hue, so the same tests have covered them.
5. The monochromatic scheme modifies only the value. Wrapping around wouldn't make sense, but we can still verify that the program handles large and small values well by displaying valid colors.
    - Repeat steps 1 to 3 but instead of hues and checking the analogous scheme, use values of 0.5, 0.9, and 0.1 and verify results in the monochromatic scheme.

ColorPickerFragment.java

- Possible inputs: non-transparent colors, partially transparent colors (how to test?), fully transparent colors

Equivalence Class	Description	Representative	Observed Output
$E_{Non-transparent}$	A non-transparent color is picked from the image	Clicking on the default dropper in the yellow/orange section	The color Freesia is picked and displayed with its HEX value and name

$E_{Fully\ transparent}$	A fully transparent “color” is picked from the image	Clicking on the checkerboard “transparency” around the dropper	The color Black is picked and displayed with its HEX value

EditColorActivity.java Equivalence Tests:

- updateSeekbarsHSV(int hue, int saturation, int value) and
    - Hue should be in the range (0,360)
    - Saturation and value should be in the range (0,100)
    - Equivalence classes and test case representatives:

Equivalence Class	Description	Representative	Observed Output

$E_{HSV\ \ in\ desired\ range}$	All of the inputs are greater than or equal to 0 and less than or equal to the max desired input	Hue = 100,
		Saturation = 50,
		Value = 50
			Seekbars update to the input

$E_{HSV\ <\ 0}$	One or more of the inputs are less than 0	Hue = -1,
		Saturation = 50,
		Value = 50
			Seekbars update to Hue = 0, saturation = 50, value = 50 (The value < 0 will be set as 0)

$E_{HSV\ >\ desired\ input}$	One or more of the inputs are greater than the max expected value	Hue = 100,
		Saturation = 200,
		Value = 50
			Seekbars update to Hue = 100,
			Saturation = 100,
			Value = 50 (The value > max will be set to max)

- updateSeekbarsRGB(int red, int green, int blue)
    - Red, green, and blue should be in the range (0, 255)

Equivalence Class	Description	Representative	Observed Output

$E_{RGB\ \ in\ desired\ range}$	All of the inputs are greater than or equal to 0 and less than or equal to 255	Red = 100, Green = 100, Blue = 100	Seekbars update to the input
$E_{RGB < 0}$	One or more of the inputs are less than 0	Red = -1. Green = 100, Blue = 100	Seekbars update to Red = 0, Green = 100, Blue = 100 (The value < 0 will be set as 0)
$E_{RGB\ \ > desired\ input}$	One or more of the inputs are greater than 255	Red = 300. Green = 100, Blue = 100	Seekbars update to Red = 255, Green = 100, Blue = 100 (The value > max will be set to max)

Color Info Equivalence Tests:

- Possible scenarios: Calling color info from color picker fragment, from a saved color, from a color in a platte, or from a color in a harmony

Equivalence Class	Description	Representative	Observed Output
$E_{color\ picker\ fragment}$	Color Info Activity was started from the color picker fragment	Sotek Green selected from the default image	Color info correctly gets the color, its name, and its values which are copyable
$E_{saved\ colors}$	Color Info Activity was started from the saved colors fragment	Figue selected from saved colors	Color info correctly gets the color, its name, and its values which are copyable
$E_{palette}$	Color Info Activity was started by clicking on a color in a palette	The dummy palette “Three Colors” was selected, then highlighter	Color info retrieves the wrong color (gets a color from saved colors). This is a bug
$E_{harmony}$	Color Info Activity was started by clicking on a color in	Color harmonies generated by	Color info retrieves the wrong color (gets a color from saved

	a harmony	Sotek Green. Analogous harmony selected, color 3 is then clicked	colors). This is a bug

ColorDatabase.java

- Possible scenarios: Called from color picker fragment when saving to saved color or palette depending on choice in the save dialog, from SavedColors where we get the info from the database to populate the list, from ColorInfoActivity where a color's name, HEX, RGB, HSV values are displayed, from EditColorActivity when a new color is saved from the new slider values.

Equivalence Class	Description	Representative	Observed Output
$E_{color\ picker\ fragment}$	ColorDatabase called when saving color from image	Vivid Imagination is picked, and saved to save color	ColorDatabase correctly saves the color Vivid Imagination into the SavedColor fragment
$E_{SavedColors}$	ColorDatabase called from saved colors to display saved colors	Vivid Imagination from color picker fragment was saved	ColorDatabase correctly displays Vivid Imagination from the color picker fragment and also new colors made from the EditColorActivity
$E_{palette}$	ColorDatabase called when saving to palette from the saved dialog that appears in color picker fragment	Dummy palettes no real call to database	Dummy palettes so no real call to database
$E_{ColorInfoActivity}$	ColorDatabase called when viewing a color, details are given like NAME, HEX, RGB,	Vivid Imagination from the SavedColor	ColorDatabase correctly displays name, HEX, RGB, HSV of Vivid

	HSV	fragment is clicked	Imagination
$E_{EditColorActivity}$	ColorDatabase called when saving the new color created from the sliders	Dead Flesh is created from the slider, original color VIvid Imagination, saved button is then clicked	ColorDatabase correctly saves and displays the new color Dead Flesh in SavedColor.