Project 5 Testing Report

storeButton1()/storeButton2():

Stores a button and its attributes into the static variable button1 and button2, respectively

adjacentButtons():

Takes the values from button1 and button2 and checks if the indices are adjacent to each other horizontally or vertically (not diagonally)

To test if two buttons are adjacent by rows, I stored a control button to button 2 at index (2,2). Then, I stored another button at index (4,2), (0,2), (3,2), and (1,2). The first two should return false because it is 2 spaces away from button2, while the last two should return true. (2,2) did not have to be tested because there is no point in the game where the same button will be compared to itself.

To test if the buttons were adjacent by columns, I did the same thing above, except the x and y were swapped (e.g. (4,2) → (2,4)).

Then, I tested the buttons diagonally with button1 at (3,3) and button2 at (2,2). It returned false.

Lastly, I tested the buttons where they were nowhere near each other. Button1 was at (2,2) and button2 was at (4,4). It returned false.

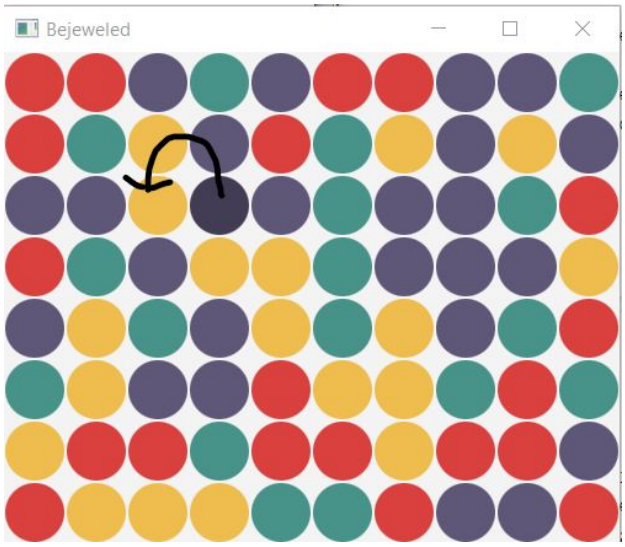
swapColors():

The method should swap the colors between two buttons. There is no junit testing for it because the method acts mostly visually.

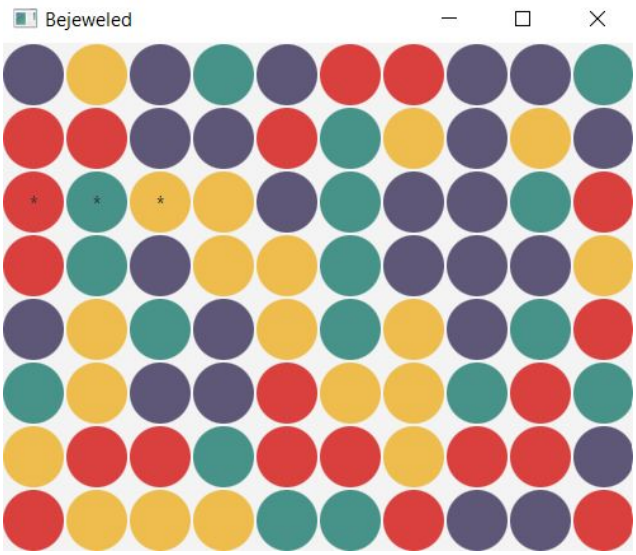
I also used the boolean value of adjacentButtons() to check if the two buttons were adjacent. If they were, then I would swap them. If they were not, the method will not execute.

tileMarkRow():

This method executes when there are 3+ consecutive colors in a horizontal row. It uses the ValueHolder values to find the initial and final index of the row of colors.

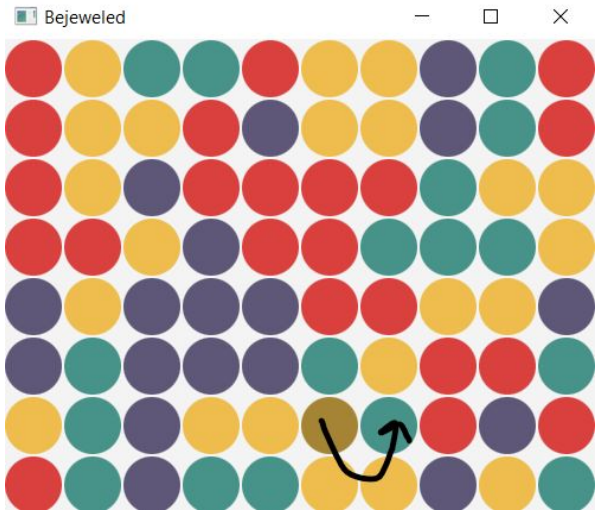


Using the indices, the method replaces the text of the button to "*" and sets the boolean[][] starMap to be true at the specific index. Then, it pushes the button colors downwards to replace the row that got taken out. In this instance, the three purples at x=0 to x=2 get replaced with red, green, and yellow. The ones above that get replaced with red, red, and purple. Finally, the top row will get a random set of colors.

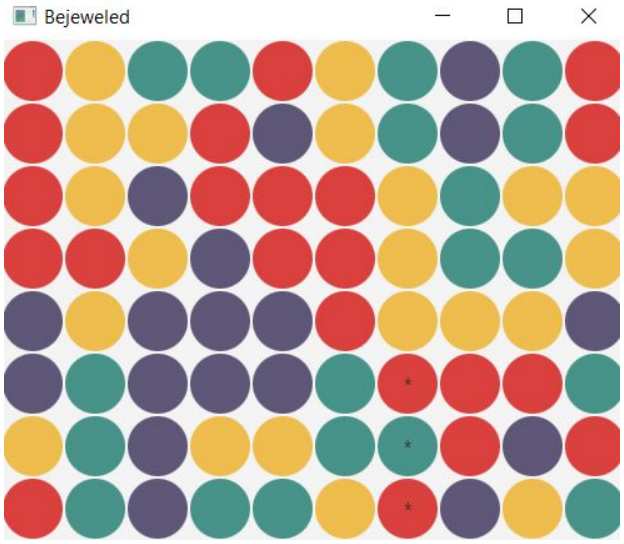


tileMarkCol():

The method checks for consecutive colors vertically. If there are 3 or more, then the method will execute in a similar fashion to tileMarkRow() in terms of setting the text to "*" and swapping colors. In this method, however, colors are not only swapped vertically, but the colors also must take into account the consecutive colors being removed.



The left yellow will swap with the green to the right of it to yield a 3 in a column of yellow. After that executes, the yellow gets overridden by the next non-yellow color, red, followed by green, red, and two yellows. Since the track of yellows did not get broken by trying to index past 0, the last button on the top also gets pushed down by three. The final 3 colors on top will be replaced by a random color. From the bottom, the board should yield red, green, red, yellow, yellow, followed by 3 randoms.



winCondition():

The method checks to see if the boolean[][] object was filled with “true.” If it does, the method returns true.

The first array I tested did not contain any true values and returned false.

The second array had all true values and returned true, accordingly.

The final array had some true values at random points and returned false.

```
You win!
Number of moves taken: 6
```

This text also got printed to the interactions panel.