CS102 – Algorithms and Programming II
Programming Assignment 5
Spring 2022

ATTENTION:

- Compress all of the Java program source files (.java) files into a single zip file.
- The name of the zip file should follow the below convention:

  CS102_Sec1_Asgn5_YourSurname_YourName.zip
- Replace the variables "Sec1", "YourSurname" and "YourName" with your actual section, surname and name.
- You may ask questions on Moodle and during the lab.
- Upload the above zip file to Moodle by the deadline (if not significant points will be taken off). You will get a chance to update and improve your solution by consulting to the TAs and tutors during the lab.

GRADING WARNING:

- Please read the grading criteria provided on Moodle. The work must be done individually. Code sharing is strictly forbidden. We are using sophisticated tools to check the code similarities. The Honor Code specifies what you can and cannot do. Breaking the rules will result in disciplinary action.

A Game of Colored Buttons

In this assignment, you are going to implement a Java program with a Graphical User Interface where you generate buttons following a recursive procedure. Clicking on buttons would change their color and the aim of the game is to make the buttons of the same group the same color.

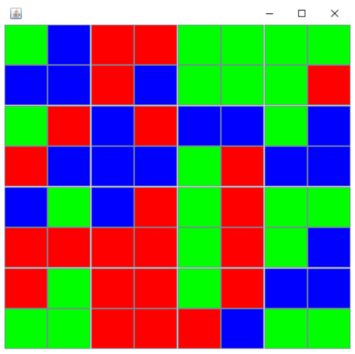
Implement a GamePanel class that extends JPanel. The constructor of this class should receive *int depth* as parameter. If depth is positive, this GamePanel should contain 4 new GamePanels in it. If depth is zero, this GamePanel should contain 4 new JButtons in it. In both cases, the GamePanel uses 2 by 2 GridLayout (*import java.awt.GridLayout to use GridLayout*).

In case the depth parameter of a GamePanel is positive, pass "depth - 1" as the input parameter of each new GamePanel that you create and add to this GamePanel. Basically, as depth is positive, we would like to recurse while decreasing depth. If depth is 0, we do not recurse but create 4 buttons in this GamePanel, using a 2 by 2 GridLayout.

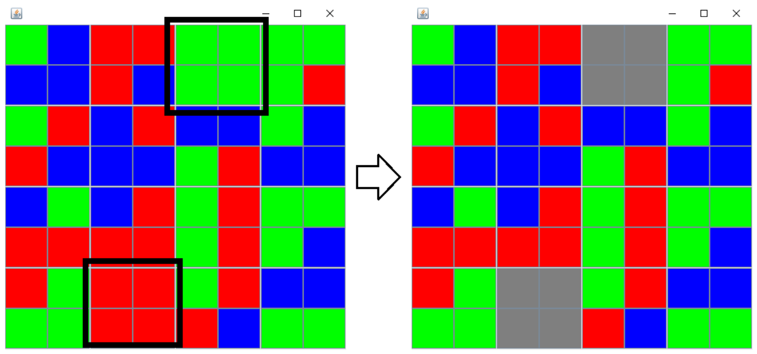
Each button you create should have a random background color that is either red or green or blue. No text will be displayed on the buttons; however, when we click on a button, we want to assign a new random color to it. Suppose the button that we click is red, then it can stay as red or switch to green or blue. You may associate integers to button colors (0: red, 1: green, 2: blue) and use a random integer to assign the color. You also need to keep the color of each button in an easy to access variable to be used in the game logic.

To test your initialization, in your main method, create a JFrame and add a new GamePanel of depth 2 to this frame. This would create 4 GamePanels of depth 1 in it, and each of those

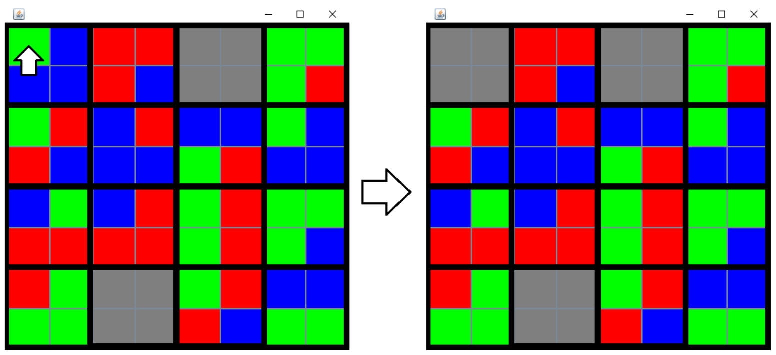
GamePanels would have 4 GamePanels of depth 0. The GamePanels with depth 0 would have 4 buttons in it, resulting in a 8 by 8 grid overall in your frame. In this configuration, each button should have a random color (red, green, or blue). A sample visual is given below.



GamePanels that include buttons in it would control the game logic. After setting up the colors of the 4 buttons in them, these GamePanels should check if all 4 buttons in it are the same color or not. If 4 buttons of a GamePanel are the same color, disable them, set their background colors to gray and increase the player's score by 10. Player's score starts from 10 at the beginning.



The aim of the player is to make all the buttons of each GamePanel the same color. Note that in this case we have 16 such GamePanels, starting with a different depth would result in a different number. During gameplay, each time we click on a button of a GamePanel, check if the buttons become the same color after a new color is assigned to the button we click. If the GamePanel's 4 buttons become the same color, disable them, set them to gray and increase the player's score by 10. For example, in the figure below, we click on the first green button which changes into blue and makes its group gray.



Each time we click on a button, the player’s score decreases by 1. This would make the player consider his/her moves, and try to start with the groups that are easy to convert into gray. Display player’s score in your frame’s title bar. This should update after each move. The game ends if player’s score becomes 0 or if all the buttons become gray. In this case, show a Confirm Dialog, asking the player if they want to play again or not.

Preliminary Submission: You will submit an early version of your solution before the final submission. This version should at least include the following:

- The functionality to recursively create the buttons with random colors should be complete.
- When we click on a button, it should switch to a random color.

You will have time to complete your solution after you submit your preliminary solution. You can consult the TAs and tutors during the lab. Do not forget to make your final submission at the end. Even if you finish the assignment in the preliminary submission, you should submit for the final submission on Moodle.

<u>Not completing the preliminary submission on time results in 50% reduction of this assignment’s final grade.</u>