MATCH THE COLOUR

Submitted in partial fulfillment of the requirements of the term work of

SE COMPUTER ENGINEERING

by

Sanket Nehe(Roll No:-27)

Pratik Patil (Roll No:- 43)

Gautam Pandey(Roll No:-33)

Guide

Prof. Merlin Priya Jacob

Department of Computer Engineering

A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

(2022-2023)

CERTIFICATE

This is to certify that the project entitled “movie ticket booking” is a bonafide work of by Sanket Nehe(Roll No:-27) Pratik Patil (Roll No:- 43)Gautam Pandey(Roll No:-33) submitted to the University of Mumbai in fulfilment of the requirement for the SBL Mini Project of Bachelor of Engineering in Computer Engineering

(PROF.MERLIN PRIYA)

Guide

(Name and Sign)	(Name and Sign)
Head of Department	Principal

ABSTRACT

This is a GUI based project which enables the user to register his/her account and also allows the user to match the colour. It is a user friendly application for children which enables user to Match the colour. The screen will display two colours on same screen .Once the colour are shown on the screen ,the user should try make the screen of same colour on both the side . The colours shown are the mixture of three colours. For matching the colour there are six button for negation and addition in colours.. The game also increases the way to see of colours of small children. This project aims to make the user experience better. This project has been designed as a substitution for the big applications which take up a lot of space and become slow after some time. After addition and negation in colour mixture ,The screen colour on both the side becomes same .Once the screen colour became same "YOU WON" interface appears . After such process if user wants to play the game once again, The Interface allows the user to "PLAY AGAIN" . Its the application works.

CONTENTS

Sr.no	Contents	Page no.
1.	Introduction	5
2.	Proposed Systems	6
3.	Specifications 3.1 Hardware Specifications 3.2 Software Specifications	7
4.	System Implementation	8

INTRODUCTION

The main purpose of our match the colour is to provide an alternate and convenient way for a customer to understand colours. It is an automatic system. After the data has been fed into the application .This project also provide knowledge about colour mixing . You just have to increase the number. This project uses 'swing' classes for designing the application and 'awt' classes for the event listener which enables the new panel to open after clicking a button. Given that you have two slots, and each slot will contain a random color like color red (R), yellow (Y), green (G), blue (B) respectively. For example, if you select YGGR (Slot-1 will be any random color, Slots-2 are green or red or any random color). The colors of slots are not known to you beforehand. You will make a guess about the colors. You might, for example, MATCH THE COLOR!!!.

When you MATCH THE COLOR the correct color for the correct slot, you get a "hit" If you match a color that exists but is in the wrong slot, you get a "the game will continue till the both the slot get matched:'

PROPOSED SYSTEM

Our project aims in making the understanding of colour mixture for user easily. For this purpose we have used the programming language Java, in which swing is used for better look of the application. We have chosen VS Code as an IDE for executing the program with JDK version 16. My biggest concern with this program is the repetitiveness of the JButtons and the functions which they run. There are 6 buttons, one that adds red, one that subtracts red, one that adds green, one that subtracts green, one that adds blue, and one that subtracts blue. The functionality is very similar because they all either increase or decrease the color by 15, so I was wondering if there was a way to condense them all into just one function

SPECIFICATION

HARDWARE SPECIFICATION:

RAM: Minimum of 4GB of ram.

Storage: 4GB of free hard disk space.

SOFTWARE SPECIFICATION:

JDK: The JDK is a development environment for building applications, applets, and components using the Java programming language. The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

SQL: MySql has been used as a database to store the information about the customer

JDBC: JDBC is the driver making the connection between the application and the database

VS Code: IDE to execute the machine vision program. Plenty of extensions, open-source, and offers cross-platform support.

Git & GitHub: Version Control System used for collaboration.

SYSTEM IMPLEMENTATION

We have made this program here (~240 lines) in which the user has to match their RGB Panel(right) with a randomized color on the left. It's a pretty fun program, and I suggest you try it out! It's fully runnable as is.

But a lot of the code seems repetitive and I was wondering how it could be condensed and made any more efficient if possible. Any other tips would be greatly appreciated. This is my first java program in a few years so I may have broken some unwritten rules.

My biggest concern with this program is the repetitiveness of the `JButtons` and the functions which they run. There are 6 buttons, one that adds red, one that subtracts red, one that adds green, one that subtracts green, one that adds blue, and one that subtracts blue. The functionality is very similar because they all either increase or decrease the color by 15, so I was wondering if there was a way to condense them all into just one function. Thanks!

I added some comments to help explain what's going on

Explanation:

`GameModel` is a plain Java class that holds the data for the game. In this simple game, we have two color fields, one for the random color and one for the user to adjust using the GUI buttons.

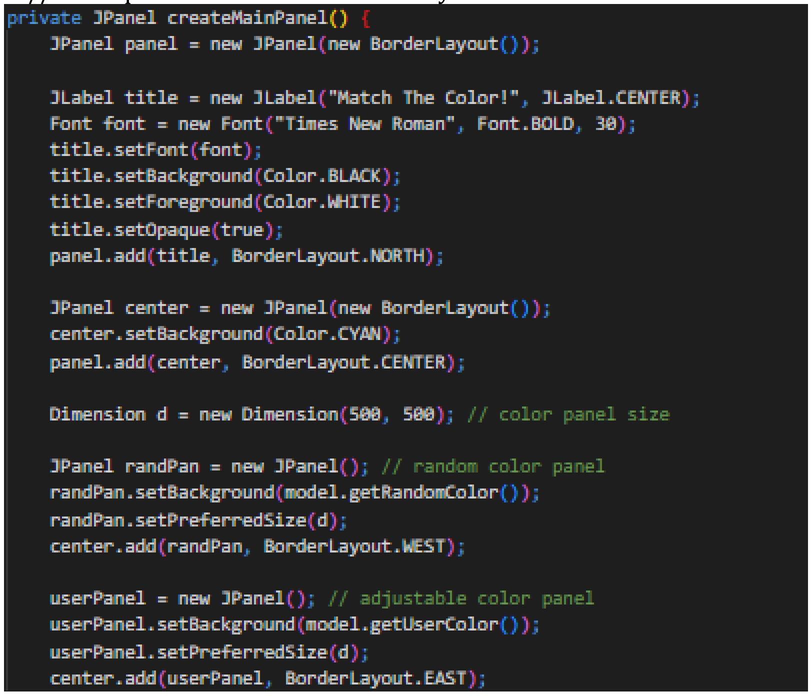
We moved the color initiation code to this class.

We have two `setRandomColor` methods, one to initialize the user color, and one to set the user color based on the GUI button actions.

Now that we've created a working game model, we can focus on the view. Here's the view code.

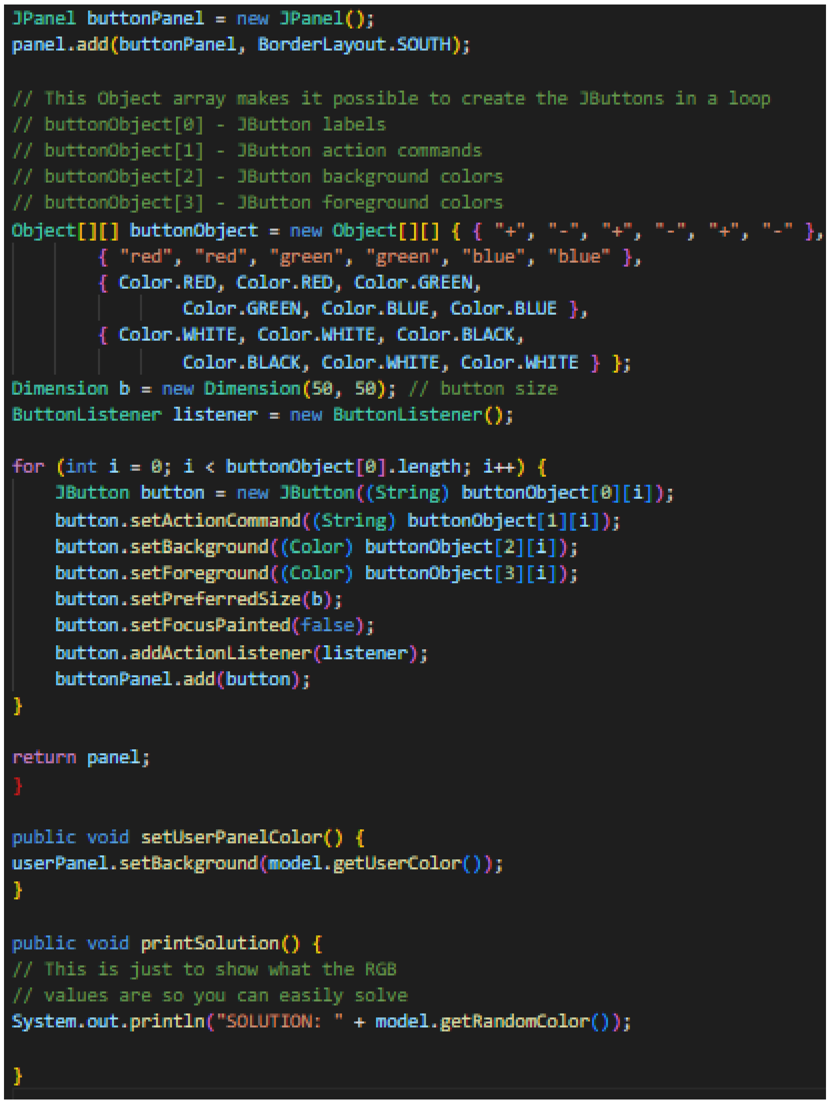
Code:

```
public class GuessColor {

    private GameModel model;

    private JFrame frame;

    private JPanel userPanel;

    public GuessColor() {
        this.model = new GameModel();
        this.model.createColors();

        frame = new JFrame("Match the color!");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.add(createMainPanel(), BorderLayout.CENTER);

        frame.pack();
        frame.setLocationRelativeTo(null);
        frame.setResizable(false);
        frame.setVisible(true);

        printSolution();
    }

    // sets up the frame and functionality of UI
```



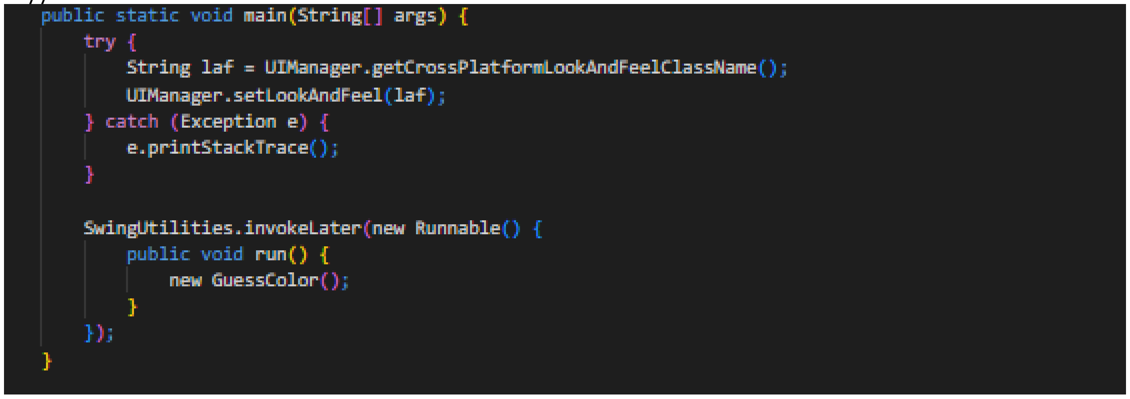
```
private JPanel createMainPanel() {
    JPanel panel = new JPanel(new BorderLayout());

    JLabel title = new JLabel("Match The Color!", JLabel.CENTER);
    Font font = new Font("Times New Roman", Font.BOLD, 30);
    title.setFont(font);
    title.setBackground(Color.BLACK);
    title.setForeground(Color.WHITE);
    title.setOpaque(true);
    panel.add(title, BorderLayout.NORTH);

    JPanel center = new JPanel(new BorderLayout());
    center.setBackground(Color.CYAN);
    panel.add(center, BorderLayout.CENTER);

    Dimension d = new Dimension(500, 500); // color panel size

    JPanel randPan = new JPanel(); // random color panel
    randPan.setBackground(model.getRandomColor());
    randPan.setPreferredSize(d);
    center.add(randPan, BorderLayout.WEST);

    userPanel = new JPanel(); // adjustable color panel
    userPanel.setBackground(model.getUserColor());
    userPanel.setPreferredSize(d);
    center.add(userPanel, BorderLayout.EAST);
```

/** BUTTONS **/



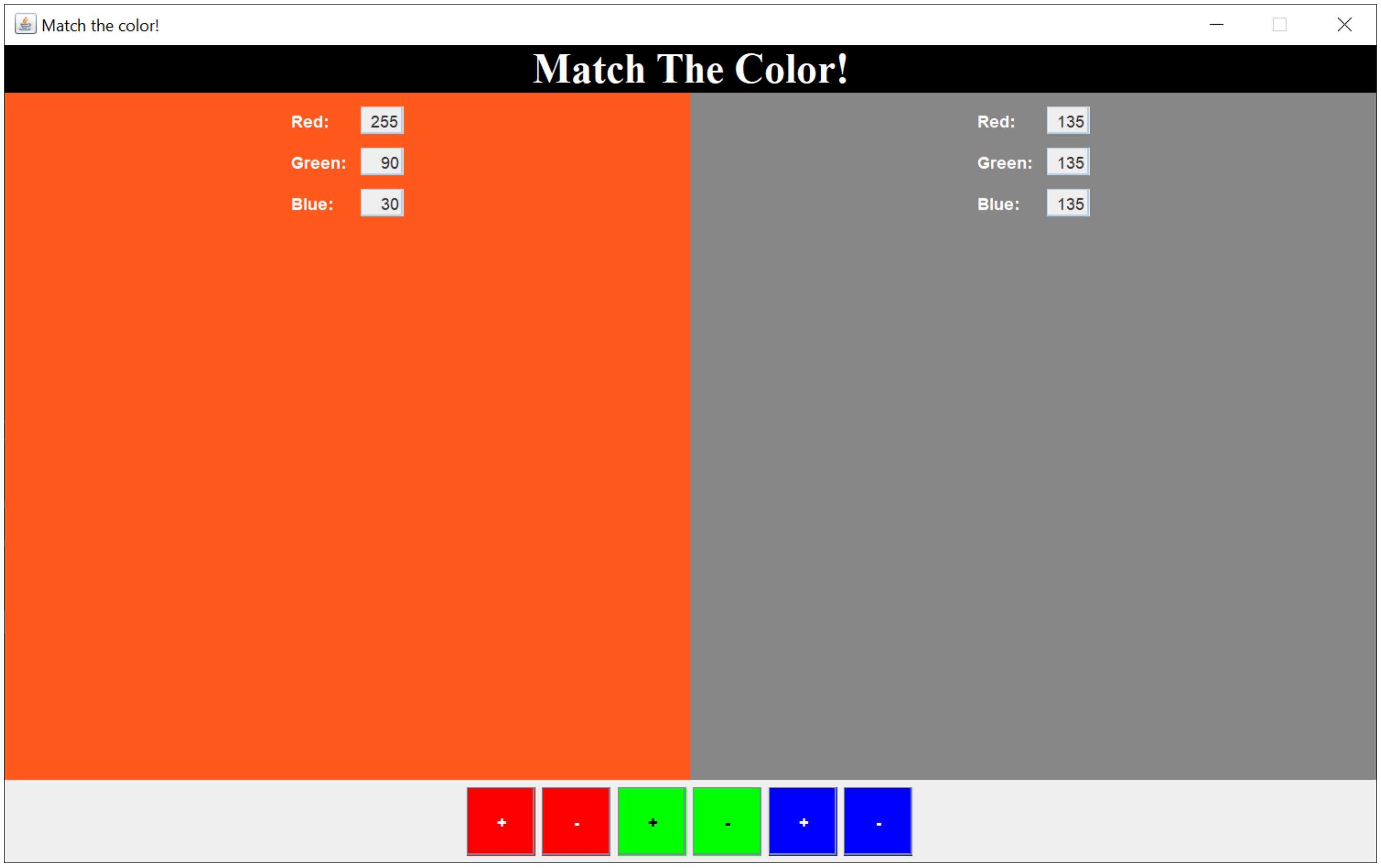
```java
JPanel buttonPanel = new JPanel();
panel.add(buttonPanel, BorderLayout.SOUTH);

// This Object array makes it possible to create the JButtons in a loop
// buttonObject[0] - JButton labels
// buttonObject[1] - JButton action commands
// buttonObject[2] - JButton background colors
// buttonObject[3] - JButton foreground colors
Object[][] buttonObject = new Object[][] { { "+", "-", "+", "-", "+", "-" },
        { "red", "red", "green", "green", "blue", "blue" },
        { Color.RED, Color.RED, Color.GREEN,
                Color.GREEN, Color.BLUE, Color.BLUE },
        { Color.WHITE, Color.WHITE, Color.BLACK,
                Color.BLACK, Color.WHITE, Color.WHITE } };
Dimension b = new Dimension(50, 50); // button size
ButtonListener listener = new ButtonListener();

for (int i = 0; i < buttonObject[0].length; i++) {
    JButton button = new JButton((String) buttonObject[0][i]);
    button.setActionCommand((String) buttonObject[1][i]);
    button.setBackground((Color) buttonObject[2][i]);
    button.setForeground((Color) buttonObject[3][i]);
    button.setPreferredSize(b);
    button.setFocusPainted(false);
    button.addActionListener(listener);
    buttonPanel.add(button);
}

return panel;
}

public void setUserPanelColor() {
userPanel.setBackground(model.getUserColor());
}

public void printSolution() {
// This is just to show what the RGB
// values are so you can easily solve
System.out.println("SOLUTION: " + model.getRandomColor());

}
```

// main method



```java
public static void main(String[] args) {
    try {
        String laf = UIManager.getCrossPlatformLookAndFeelClassName();
        UIManager.setLookAndFeel(laf);
    } catch (Exception e) {
        e.printStackTrace();
    }

    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new GuessColor();
        }
    });
}
```

Final Output of Game:



CONCLUSION:

Hence we have successfully implemented and executed the MATCH THE COLOR!!!!!