import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.util.\*;

public class Main extends JFrame {

// List of possible colors in the game

private final String[] colors = {"Red", "Green", "Blue", "Yellow", "Orange", "Purple"};

private final String[] secretCode = new String[4];

private int attemptsLeft = 10;

// GUI components

private ArrayList<JComboBox<String>> guessComboBoxes = new ArrayList<>();

private JButton submitButton;

private JTextArea feedbackArea;

private JPanel feedbackPanel;

private JPanel colorDisplayPanel;

private JLabel attemptsLabel;

private JButton resetButton;

public Main() {

// Set up JFrame properties

setTitle("Mastermind Game");

setSize(600, 700);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new FlowLayout(FlowLayout.CENTER, 20, 20));

setLocationRelativeTo(null);

getContentPane().setBackground(new Color(240, 240, 240));

// Generate secret code

generateSecretCode();

// Create guess input fields (JComboBox)

JPanel guessPanel = new JPanel();

guessPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 20, 20));

guessPanel.setBackground(new Color(240, 240, 240));

for (int i = 0; i < 4; i++) {

JComboBox<String> comboBox = new JComboBox<>(colors);

comboBox.setPreferredSize(new Dimension(100, 40));

guessComboBoxes.add(comboBox);

guessPanel.add(comboBox);

}

add(guessPanel);

// Submit button

submitButton = new JButton("Submit Guess");

submitButton.setBackground(new Color(50, 150, 255));

submitButton.setForeground(Color.WHITE);

submitButton.setFocusPainted(false);

submitButton.setFont(new Font("Arial", Font.BOLD, 16));

submitButton.addActionListener(new SubmitButtonListener());

submitButton.setPreferredSize(new Dimension(150, 40));

add(submitButton);

// Feedback area

feedbackArea = new JTextArea();

feedbackArea.setEditable(false);

feedbackArea.setFont(new Font("Arial", Font.PLAIN, 14));

feedbackArea.setBackground(Color.LIGHT\_GRAY);

feedbackArea.setLineWrap(true);

feedbackArea.setWrapStyleWord(true);

feedbackArea.setPreferredSize(new Dimension(500, 100));

add(new JScrollPane(feedbackArea));

// Color display panel for showing selected colors and feedback

colorDisplayPanel = new JPanel();

colorDisplayPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 20, 20));

colorDisplayPanel.setBackground(new Color(240, 240, 240));

for (int i = 0; i < 4; i++) {

JLabel label = new JLabel();

label.setPreferredSize(new Dimension(50, 50));

label.setHorizontalAlignment(SwingConstants.CENTER);

label.setFont(new Font("Arial", Font.BOLD, 30));

label.setOpaque(true);

label.setBackground(Color.GRAY); // Default color (gray)

colorDisplayPanel.add(label);

}

add(colorDisplayPanel);

// Attempts label

attemptsLabel = new JLabel("Attempts Left: " + attemptsLeft);

attemptsLabel.setFont(new Font("Arial", Font.BOLD, 16));

attemptsLabel.setForeground(Color.RED);

add(attemptsLabel);

// Reset button

resetButton = new JButton("Reset Game");

resetButton.setBackground(new Color(255, 100, 100));

resetButton.setForeground(Color.WHITE);

resetButton.setFocusPainted(false);

resetButton.setFont(new Font("Arial", Font.BOLD, 16));

resetButton.addActionListener(new ResetButtonListener());

resetButton.setPreferredSize(new Dimension(150, 40));

add(resetButton);

setVisible(true);

}

// Method to generate a random secret code

private void generateSecretCode() {

Random rand = new Random();

for (int i = 0; i < 4; i++) {

secretCode[i] = colors[rand.nextInt(colors.length)];

}

System.out.println("Secret Code: " + Arrays.toString(secretCode)); // Debugging line

}

// Method to evaluate the guess

private String evaluateGuess() {

StringBuilder feedback = new StringBuilder();

String[] guess = new String[4];

for (int i = 0; i < 4; i++) {

guess[i] = (String) guessComboBoxes.get(i).getSelectedItem();

}

int correctPosition = 0, correctColor = 0;

boolean[] secretUsed = new boolean[4]; // Track used positions in the secret code

boolean[] guessUsed = new boolean[4]; // Track used positions in the guess

// Check for correct positions

for (int i = 0; i < 4; i++) {

if (secretCode[i].equals(guess[i])) {

correctPosition++;

secretUsed[i] = true; // Mark secret code position as used

guessUsed[i] = true; // Mark guess position as used

}

}

// Check for correct color but wrong position

for (int i = 0; i < 4; i++) {

if (!guessUsed[i]) { // Skip already used positions in the guess

for (int j = 0; j < 4; j++) {

if (!secretUsed[j] && secretCode[j].equals(guess[i])) {

correctColor++;

secretUsed[j] = true; // Mark secret code position as used

break;

}

}

}

}

// Append feedback

feedback.append("Correct positions: ").append(correctPosition).append("\n");

feedback.append("Correct colors but wrong positions: ").append(correctColor).append("\n");

// Check for win

if (correctPosition == 4) {

feedback.append("Congratulations! You guessed the correct code!");

} else if (attemptsLeft == 0) {

feedback.append("Game Over! You've used all your attempts.\nThe secret code was: ");

feedback.append(Arrays.toString(secretCode));

revealSecretCode(); // Reveal the secret code after all attempts are used

}

// Update the visual feedback (color squares)

updateFeedbackPanel(correctPosition, correctColor);

return feedback.toString();

}

// Method to update the feedback panel with color squares

private void updateFeedbackPanel(int correctPosition, int correctColor) {

// Clear previous feedback (set all to gray)

Component[] labels = colorDisplayPanel.getComponents();

for (Component label : labels) {

JLabel jLabel = (JLabel) label;

jLabel.setBackground(Color.GRAY);

}

// Update feedback squares based on correct positions and colors

for (int i = 0; i < 4; i++) {

JLabel jLabel = (JLabel) labels[i];

if (correctPosition > 0) {

jLabel.setBackground(Color.BLACK); // Correct color in the correct position

correctPosition--;

} else if (correctColor > 0) {

jLabel.setBackground(Color.WHITE); // Correct color in the wrong position

correctColor--;

}

}

}

// Method to reveal the secret code after the game ends

private void revealSecretCode() {

Component[] labels = colorDisplayPanel.getComponents();

for (int i = 0; i < 4; i++) {

JLabel jLabel = (JLabel) labels[i];

String color = secretCode[i];

switch (color) {

case "Red":

jLabel.setBackground(Color.RED);

break;

case "Green":

jLabel.setBackground(Color.GREEN);

break;

case "Blue":

jLabel.setBackground(Color.BLUE);

break;

case "Yellow":

jLabel.setBackground(Color.YELLOW);

break;

case "Orange":

jLabel.setBackground(Color.ORANGE);

break;

case "Purple":

jLabel.setBackground(Color.MAGENTA);

break;

}

}

}

// Event listener for submit button

private class SubmitButtonListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

if (attemptsLeft > 0) {

String feedback = evaluateGuess();

feedbackArea.setText(feedback);

attemptsLeft--;

attemptsLabel.setText("Attempts Left: " + attemptsLeft);

}

if (attemptsLeft == 0) {

submitButton.setEnabled(false);

}

}

}

// Event listener for reset button

private class ResetButtonListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

attemptsLeft = 10;

generateSecretCode();

guessComboBoxes.forEach(comboBox -> comboBox.setSelectedIndex(0));

feedbackArea.setText("");

submitButton.setEnabled(true);

attemptsLabel.setText("Attempts Left: " + attemptsLeft);

// Reset the color display

Component[] labels = colorDisplayPanel.getComponents();

for (Component label : labels) {

JLabel jLabel = (JLabel) label;

jLabel.setBackground(Color.GRAY);

}

}

}

public static void main(String[] args) {

new Main();

}

}