```
package lesson8;

import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.GridLayout;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.SwingConstants;
import javax.swing.SwingUtilities;

public class WordTypingGame extends JFrame {

	private JButton[][] buttons = new JButton[2][4];
	private String[][] values = new String[2][4];
	private boolean[][] matched = new boolean[2][4];

	private List<String> words = Arrays.asList("🍎", "🍌", "🍇", "🍈", "🍎", "🍌", "🍇", "🍈");

	private JButton firstButton = null;
	private int firstRow, firstCol;
	private boolean waiting = false;

	private int currentPlayer = 0;
	private int totalPlayers;
	private String[] playerNames;
	private int[] playerTimes;

	private JLabel statusLabel = new JLabel();
	private JLabel timerLabel = new JLabel("Time: 0 seconds");

	private javax.swing.Timer gameTimer;
	private int elapsedTime = 0;

	public WordTypingGame() {
		setTitle("🍎 Word Typing Match Game - Multiplayer");
		setSize(700, 400);
		setDefaultCloseOperation(EXIT_ON_CLOSE);
```

```
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());

        askPlayerInfo(); // Prompt number of players and names

        JPanel boardPanel = new JPanel(new GridLayout(2, 4, 10, 10));
        Font btnFont = new Font("Segoe UI Emoji", Font.BOLD, 48);

        for (int r = 0; r < 2; r++) {
            for (int c = 0; c < 4; c++) {
                JButton btn = new JButton("❓");
                btn.setFont(btnFont);
                final int row = r;
                final int col = c;
                btn.addActionListener(e -> handleClick(btn, row, col));
                buttons[r][c] = btn;
                boardPanel.add(btn);
            }
        }

        JPanel infoPanel = new JPanel(new GridLayout(1, 2));
        statusLabel.setHorizontalAlignment(SwingConstants.CENTER);
        timerLabel.setHorizontalAlignment(SwingConstants.CENTER);
        infoPanel.add(statusLabel);
        infoPanel.add(timerLabel);

        add(infoPanel, BorderLayout.NORTH);
        add(boardPanel, BorderLayout.CENTER);

        startNewRound();
        setVisible(true);
    }

    private void askPlayerInfo() {
        while (true) {
            String input = JOptionPane.showInputDialog(this, "Enter number of
players (2–4):");
            try {
                totalPlayers = Integer.parseInt(input);
                if (totalPlayers >= 2 && totalPlayers <= 4)
                    break;
            } catch (Exception ignored) {
            }
        }
```

```
                playerNames = new String[totalPlayers];
                playerTimes = new int[totalPlayers];

                for (int i = 0; i < totalPlayers; i++) {
                        playerNames[i] = JOptionPane.showInputDialog(this, "Enter name for
Player " + (i + 1) + ":");
                        if (playerNames[i] == null || playerNames[i].trim().isEmpty()) {
                                playerNames[i] = "Player " + (i + 1);
                        }
                }
        }

        private void setupGameTimer() {
                if (gameTimer != null) {
                        gameTimer.stop();
                }
                elapsedTime = 0;
                timerLabel.setText("Time: 0 seconds");
                gameTimer = new javax.swing.Timer(1000, e -> {
                        elapsedTime++;
                        timerLabel.setText("Time: " + elapsedTime + " seconds");
                });
                gameTimer.start();
        }

        private void startNewRound() {
                Collections.shuffle(words);
                fillBoard();
                resetButtons();
                firstButton = null;
                waiting = false;
                statusLabel.setText(playerNames[currentPlayer] + "'s turn");
                setupGameTimer();
        }

        private void fillBoard() {
                int idx = 0;
                for (int r = 0; r < 2; r++) {
                        for (int c = 0; c < 4; c++) {
                                values[r][c] = words.get(idx++);
                                matched[r][c] = false;
                        }
                }
```

```
	}

	private void resetButtons() {
		for (int r = 0; r < 2; r++) {
			for (int c = 0; c < 4; c++) {
				buttons[r][c].setText(" ❓ ");
				buttons[r][c].setEnabled(true);
			}
		}
	}

	private void handleClick(JButton btn, int row, int col) {
		if (matched[row][col] || waiting || btn == firstButton)
			return;

		btn.setEnabled(false);

		javax.swing.Timer flipTimer = new javax.swing.Timer(300, e -> {
			btn.setText(values[row][col]);
			btn.setEnabled(true);

			if (firstButton == null) {
				firstButton = btn;
				firstRow = row;
				firstCol = col;
			} else {
				waiting = true;
				if (values[firstRow][firstCol].equals(values[row][col])) {
					matched[firstRow][firstCol] = true;
					matched[row][col] = true;
					firstButton = null;
					waiting = false;

					if (checkGameOver()) {
						gameTimer.stop();
						playerTimes[currentPlayer] = elapsedTime;

						if (currentPlayer < totalPlayers - 1) {
							JOptionPane.showMessageDialog(this,
									playerNames[currentPlayer]
+ " finished in " + elapsedTime + " seconds!\n" +

playerNames[currentPlayer + 1] + "'s turn next.");
							currentPlayer++;
```

```
                                        startNewRound();
                                    } else {
                                        showFinalResult();
                                    }
                                }
                            } else {
                                javax.swing.Timer delay = new javax.swing.Timer(1000, ev
-> {
                                    firstButton.setText("❓");
                                    btn.setText("❓");
                                    firstButton = null;
                                    waiting = false;
                                });
                                delay.setRepeats(false);
                                delay.start();
                            }
                        }

                        ((javax.swing.Timer) e.getSource()).stop();
                    });

                    flipTimer.setRepeats(false);
                    flipTimer.start();
                }

                private boolean checkGameOver() {
                    for (boolean[] row : matched) {
                        for (boolean matchedCell : row) {
                            if (!matchedCell)
                                return false;
                        }
                    }
                    return true;
                }

                private void showFinalResult() {
                    int minTime = playerTimes[0];
                    int winnerIndex = 0;
                    boolean tie = false;

                    for (int i = 1; i < totalPlayers; i++) {
                        if (playerTimes[i] < minTime) {
                            minTime = playerTimes[i];
                            winnerIndex = i;
```

```
                        tie = false;
                    } else if (playerTimes[i] == minTime) {
                        tie = true;
                    }
                }

                StringBuilder result = new StringBuilder("🏁 Final Results:\n");
                for (int i = 0; i < totalPlayers; i++) {
                    result.append(playerNames[i]).append(":
").append(playerTimes[i]).append(" seconds\n");
                }

                result.append("\n");
                if (tie) {
                    result.append("It's a tie! 🤝");
                } else {
                    result.append(playerNames[winnerIndex]).append(" wins! 🎉");
                }

                JOptionPane.showMessageDialog(this, result.toString());
                System.exit(0);
            }

            public static void main(String[] args) {
                SwingUtilities.invokeLater(() -> new WordTypingGame());
            }
}
```