GamePanel.java

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

public class GamePanel extends JPanel {
    private GameFrame gameFrame;
    private JButton[][] buttons;
    private Player player1, player2;
    private JLabel player1Label, player2Label, statusLabel;
    private int turnCount;
    private boolean player1Turn;

    public GamePanel(GameFrame gameFrame) {
        this.gameFrame = gameFrame;
        setLayout(new BorderLayout());

        setBackground(new Color(224, 255, 255)); // Set background color to light cyan

        // Score panel to display player names and scores
        JPanel scorePanel = new JPanel();
        scorePanel.setBackground(new Color(224, 255, 255)); // Set background color to light cyan
        player1Label = new JLabel();
        player2Label = new JLabel();
        player1Label.setFont(new Font("Arial", Font.PLAIN, 18)); // Set font and size
        player2Label.setFont(new Font("Arial", Font.PLAIN, 18)); // Set font and size
        player1Label.setForeground(new Color(70, 130, 180)); // Set text color to steel blue
        player2Label.setForeground(new Color(70, 130, 180)); // Set text color to steel blue
        scorePanel.add(player1Label); // Add player 1 label to score panel
        scorePanel.add(player2Label); // Add player 2 label to score panel
        add(scorePanel, BorderLayout.NORTH); // Add score panel to the top of the panel

        // Game board panel with buttons
        JPanel gameBoard = new JPanel(new GridLayout(3, 3));
        buttons = new JButton[3][3];
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
```

```
                buttons[i][j] = new JButton("");
                buttons[i][j].setFont(new Font("Arial", Font.BOLD, 60)); // Set font and size
                buttons[i][j].setFocusPainted(false);
                buttons[i][j].setBackground(new Color(176, 224, 230)); // Set background color to powder blue
                buttons[i][j].setForeground(new Color(25, 25, 112)); // Set text color to midnight blue
                buttons[i][j].addActionListener(new ButtonClickListener(i, j)); // Add action listener to buttons
                gameBoard.add(buttons[i][j]); // Add button to game board panel
            }
        }
        add(gameBoard, BorderLayout.CENTER); // Add game board panel to the center of the panel

        // Control panel with status label and end game button
        JPanel controlPanel = new JPanel();
        controlPanel.setBackground(new Color(224, 255, 255)); // Set background color to light cyan
        statusLabel = new JLabel(" ");
        statusLabel.setFont(new Font("Arial", Font.PLAIN, 18)); // Set font and size
        statusLabel.setForeground(new Color(70, 130, 180)); // Set text color to steel blue
        controlPanel.add(statusLabel); // Add status label to control panel

        // End game button to end the current game
        JButton endGameButton = new JButton("End Game");
        endGameButton.setFont(new Font("Arial", Font.PLAIN, 18)); // Set font and size
        endGameButton.setBackground(new Color(176, 224, 230)); // Set background color to powder blue
        endGameButton.setForeground(new Color(25, 25, 112)); // Set text color to midnight blue
        endGameButton.addActionListener(e -> {
            // Display final scores when end game button is clicked
            String message = String.format("%s: %d\n%s: %d",
                    player1.getName(), player1.getScore(),
                    player2.getName(), player2.getScore());
            JOptionPane.showMessageDialog(this, message, "Final Scores", JOptionPane.INFORMATION_MESSAGE);
            gameFrame.showMainMenuPanel(); // Switch to main menu panel
        });

        controlPanel.add(endGameButton); // Add end game button to control panel
        add(controlPanel, BorderLayout.SOUTH); // Add control panel to the bottom of the panel
    }

    // Set player names for the game
    public void setPlayerNames(String player1Name, String player2Name) {
        player1 = new Player(player1Name);
        player2 = new Player(player2Name);
        player1Label.setText(player1.getName() + ": " + player1.getScore()); // Update player 1 label
```

```
83          player2Label.setText(player2.getName() + ": " + player2.getScore()); // Update player 2 label
84          resetBoard(); // Reset the game board
85      }
86
87      // Reset the game board
88      private void resetBoard() {
89          turnCount = 0;
90          player1Turn = true;
91          statusLabel.setText(player1.getName() + "'s turn (X)"); // Display player 1's turn
92          for (int i = 0; i < 3; i++) {
93              for (int j = 0; j < 3; j++) {
94                  buttons[i][j].setText(""); // Clear button text
95                  buttons[i][j].setEnabled(true); // Enable buttons
96              }
97          }
98      }
99
100     // Update player scores
101     private void updateScores() {
102         player1Label.setText(player1.getName() + ": " + player1.getScore()); // Update player 1 label
103         player2Label.setText(player2.getName() + ": " + player2.getScore()); // Update player 2 label
104     }
105
106     // Check game status to determine winner or draw
107     private void checkGameStatus() {
108         String winner = getWinner();
109         if (winner != null) {
110             if (winner.equals("X")) {
111                 player1.incrementScore(); // Increment player 1's score
112
113                 JOptionPane.showMessageDialog(this, player1.getName() + " (X) wins!", "Game Over",
    JOptionPane.INFORMATION_MESSAGE);
114                 saveResultToFile(player1.getName() + " (X) wins!"); // Save game result to file
115             } else if (winner.equals("O")) {
116                 player2.incrementScore(); // Increment player 2's score
117                 JOptionPane.showMessageDialog(this, player2.getName() + " (O) wins!", "Game Over",
    JOptionPane.INFORMATION_MESSAGE);
118                 saveResultToFile(player2.getName() + " (O) wins!"); // Save game result to file
119             }
120             updateScores(); // Update player scores
121             resetBoard(); // Reset the game board
122         } else if (turnCount == 9) {
123             JOptionPane.showMessageDialog(this, "It's a draw!", "Game Over", JOptionPane.INFORMATION_MESSAGE);
```

```
124             saveResultToFile("It's a draw!"); // Save game result to file
125             resetBoard(); // Reset the game board
126         } else {
127             statusLabel.setText((player1Turn ? player1.getName() + "'s turn (X)" : player2.getName() + "'s turn (O)")); //
    Display player's turn
128         }
129     }
130 
131     // Determine the winner of the game
132     private String getWinner() {
133         for (int i = 0; i < 3; i++) {
134             if (!buttons[i][0].getText().isEmpty() &&
135                 buttons[i][0].getText().equals(buttons[i][1].getText()) &&
136                 buttons[i][1].getText().equals(buttons[i][2].getText())) {
137                 return buttons[i][0].getText(); // Return winning symbol
138             }
139         }
140 
141         for (int j = 0; j < 3; j++) {
142             if (!buttons[0][j].getText().isEmpty() &&
143                 buttons[0][j].getText().equals(buttons[1][j].getText()) &&
144                 buttons[1][j].getText().equals(buttons[2][j].getText())) {
145                 return buttons[0][j].getText(); // Return winning symbol
146             }
147         }
148 
149         if (!buttons[0][0].getText().isEmpty() &&
150             buttons[0][0].getText().equals(buttons[1][1].getText()) &&
151             buttons[1][1].getText().equals(buttons[2][2].getText())) {
152             return buttons[0][0].getText(); // Return winning symbol
153         }
154 
155         if (!buttons[0][2].getText().isEmpty() &&
156             buttons[0][2].getText().equals(buttons[1][1].getText()) &&
157             buttons[1][1].getText().equals(buttons[2][0].getText())) {
158             return buttons[0][2].getText(); // Return winning symbol
159         }
160 
161         return null; // Return null if there's no winner
162     }
163 
164     // Save game result to file
```

```
165     private void saveResultToFile(String result) {
166         try (BufferedWriter writer = new BufferedWriter(new FileWriter("scoreboard.txt", true))) {
167             writer.write(player1.getName() + " vs " + player2.getName() + ": " + result); // Write game result to file
168             writer.newLine(); // Write a new line
169         } catch (IOException e) {
170             e.printStackTrace(); // Print exception stack trace if an error occurs
171         }
172     }
173 
174     // Action listener for game buttons
175     private class ButtonClickListener implements ActionListener {
176         private int x, y;
177 
178         public ButtonClickListener(int x, int y) {
179             this.x = x;
180             this.y = y;
181         }
182 
183         @Override
184         public void actionPerformed(ActionEvent e) {
185             if (buttons[x][y].getText().isEmpty()) {
186                 buttons[x][y].setText(player1Turn ? "X" : "O"); // Set symbol based on player turn
187                 buttons[x][y].setEnabled(false); // Disable button
188                 turnCount++; // Increment turn count
189                 player1Turn = !player1Turn; // Toggle player turn
190                 checkGameStatus(); // Check game status
191             }
192         }
193     }
194 }
195 
```