**PLAYER.JAVA**

import javax.swing.\*;

public class Player {

private String name;

private ImageIcon profilePicture;

private int wins;

public Player(String name, ImageIcon profilePicture) {

this.name = name;

this.profilePicture = profilePicture;

this.wins = 0;

}

public String getName() {

return name;

}

public ImageIcon getProfilePicture() {

return profilePicture;

}

public int getWins() {

return wins;

}

public void incrementWins() {

wins++;

}

public void resetWins() {

wins = 0;

}

}

LOGIN SCREEN.JAVA

import javax.swing.\*;

import java.awt.\*;

import java.io.File;

public class LoginScreen extends JFrame {

private JTextField nameField;

private JLabel imageLabel;

private ImageIcon profilePic;

public LoginScreen() {

setTitle("Login");

setLayout(new BorderLayout());

setSize(400, 300);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

JPanel centerPanel = new JPanel(new GridLayout(3, 1));

nameField = new JTextField();

centerPanel.add(new JLabel("Enter your name:"));

centerPanel.add(nameField);

JButton uploadButton = new JButton("Upload Profile Picture");

uploadButton.addActionListener(e -> choosePicture());

centerPanel.add(uploadButton);

imageLabel = new JLabel();

imageLabel.setHorizontalAlignment(JLabel.CENTER);

add(centerPanel, BorderLayout.CENTER);

add(imageLabel, BorderLayout.SOUTH);

JButton startButton = new JButton("Start Game");

startButton.addActionListener(e -> startGame());

add(startButton, BorderLayout.NORTH);

setVisible(true);

}

private void choosePicture() {

JFileChooser chooser = new JFileChooser();

int result = chooser.showOpenDialog(this);

if (result == JFileChooser.APPROVE\_OPTION) {

File file = chooser.getSelectedFile();

profilePic = new ImageIcon(file.getAbsolutePath());

Image img = profilePic.getImage().getScaledInstance(80, 80, Image.SCALE\_SMOOTH);

profilePic = new ImageIcon(img);

imageLabel.setIcon(profilePic);

}

}

private void startGame() {

String name = nameField.getText().trim();

if (name.isEmpty() || profilePic == null) {

JOptionPane.showMessageDialog(this, "Please enter a name and upload a picture.");

return;

}

Player player = new Player(name, profilePic);

new GameBoard(player);

dispose();

}

}

GAMEBOARD.JAVA

import javax.swing.\*;

import java.awt.\*;

import java.io.\*;

import java.time.LocalDateTime;

import java.util.ArrayList;

import java.util.Random;

public class GameBoard extends JFrame {

private JButton[][] buttons = new JButton[3][3];

private Player player;

private JLabel statusLabel, scoreLabel, picLabel;

private int moves = 0;

private char currentMark = 'X';

private String mode;

private Random rand = new Random();

public GameBoard(Player player, String mode) {

this.player = player;

this.mode = mode;

setTitle("Tic Tac Toe");

setSize(500, 550);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

JPanel grid = new JPanel(new GridLayout(3, 3));

Font font = new Font("Arial", Font.BOLD, 50);

for (int i = 0; i < 3; i++)

for (int j = 0; j < 3; j++) {

buttons[i][j] = new JButton();

buttons[i][j].setFont(font);

final int x = i, y = j;

buttons[i][j].addActionListener(e -> handleClick(x, y));

grid.add(buttons[i][j]);

}

JPanel topPanel = new JPanel(new BorderLayout());

statusLabel = new JLabel("Your Turn (X)", JLabel.CENTER);

scoreLabel = new JLabel("Wins: 0", JLabel.CENTER);

picLabel = new JLabel(player.getProfilePicture(), JLabel.CENTER);

topPanel.add(statusLabel, BorderLayout.NORTH);

topPanel.add(scoreLabel, BorderLayout.CENTER);

topPanel.add(picLabel, BorderLayout.SOUTH);

JButton reset = new JButton("Reset");

reset.addActionListener(e -> resetBoard());

JButton save = new JButton("Save History");

save.addActionListener(e -> saveHistory());

JButton view = new JButton("View History");

view.addActionListener(e -> HistoryManager.viewHistory());

JButton clear = new JButton("Clear History");

clear.addActionListener(e -> HistoryManager.clearHistory());

JPanel bottom = new JPanel();

bottom.add(reset);

bottom.add(save);

bottom.add(view);

bottom.add(clear);

add(topPanel, BorderLayout.NORTH);

add(grid, BorderLayout.CENTER);

add(bottom, BorderLayout.SOUTH);

setVisible(true);

}

private void handleClick(int row, int col) {

if (!buttons[row][col].getText().isEmpty()) return;

buttons[row][col].setText(String.valueOf(currentMark));

buttons[row][col].setEnabled(false);

moves++;

if (checkWin()) {

JOptionPane.showMessageDialog(this, currentMark + " Wins!");

player.incrementWins();

scoreLabel.setText("Wins: " + player.getWins());

if (player.getWins() >= 5) {

JOptionPane.showMessageDialog(this, "🎉 Game Over. 5 Wins Achieved!");

System.exit(0);

}

resetBoard();

} else if (moves == 9) {

JOptionPane.showMessageDialog(this, "Draw!");

resetBoard();

} else {

currentMark = currentMark == 'X' ? 'O' : 'X';

statusLabel.setText("Your Turn (" + currentMark + ")");

if (mode.contains("Single") && currentMark == 'O') {

makeAIMove();

}

}

}

private void makeAIMove() {

if (mode.contains("Easy")) {

makeRandomMove();

} else {

if (!makeWinningMove('O'))

if (!makeWinningMove('X'))

makeStrategicMove();

}

}

private void makeRandomMove() {

ArrayList<JButton> empty = new ArrayList<>();

for (JButton[] row : buttons)

for (JButton btn : row)

if (btn.getText().isEmpty()) empty.add(btn);

if (!empty.isEmpty()) {

JButton btn = empty.get(rand.nextInt(empty.size()));

btn.doClick();

}

}

private boolean makeWinningMove(char symbol) {

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

if (buttons[i][j].getText().isEmpty()) {

buttons[i][j].setText(String.valueOf(symbol));

if (checkWin()) {

if (symbol == 'O') {

buttons[i][j].setEnabled(false);

} else {

buttons[i][j].setText("O");

buttons[i][j].setEnabled(false);

}

return true;

}

buttons[i][j].setText("");

}

}

}

return false;

}

private void makeStrategicMove() {

if (buttons[1][1].getText().isEmpty()) {

buttons[1][1].doClick();

return;

}

int[][] corners = {{0, 0}, {0, 2}, {2, 0}, {2, 2}};

for (int[] pos : corners)

if (buttons[pos[0]][pos[1]].getText().isEmpty()) {

buttons[pos[0]][pos[1]].doClick();

return;

}

makeRandomMove();

}

private boolean checkWin() {

String mark = String.valueOf(currentMark);

for (int i = 0; i < 3; i++)

if (mark.equals(buttons[i][0].getText()) && mark.equals(buttons[i][1].getText()) && mark.equals(buttons[i][2].getText()))

return true;

for (int j = 0; j < 3; j++)

if (mark.equals(buttons[0][j].getText()) && mark.equals(buttons[1][j].getText()) && mark.equals(buttons[2][j].getText()))

return true;

return (mark.equals(buttons[0][0].getText()) && mark.equals(buttons[1][1].getText()) && mark.equals(buttons[2][2].getText())) ||

(mark.equals(buttons[0][2].getText()) && mark.equals(buttons[1][1].getText()) && mark.equals(buttons[2][0].getText()));

}

private void resetBoard() {

moves = 0;

currentMark = 'X';

statusLabel.setText("Your Turn (X)");

for (JButton[] row : buttons)

for (JButton btn : row) {

btn.setText("");

btn.setEnabled(true);

}

}

private void saveHistory() {

HistoryManager.saveHistory(player.getName(), player.getWins());

JOptionPane.showMessageDialog(this, "Game history saved!");

}

}

MAIN.JAVA

public class Main {

public static void main(String[] args) {

javax.swing.SwingUtilities.invokeLater(() -> new LoginScreen());

}

}

**HistoryManager.java**

import javax.swing.\*;

import java.io.\*;

import java.time.LocalDateTime;

public class HistoryManager {

private static final String FILE\_NAME = "game\_history.txt";

// Save game history for the player

public static void saveHistory(String playerName, int wins) {

try (BufferedWriter writer = new BufferedWriter(new FileWriter(FILE\_NAME, true))) {

writer.write(playerName + " - Wins: " + wins + " - " + LocalDateTime.now());

writer.newLine();

} catch (IOException e) {

e.printStackTrace();

}

}

// View game history in a scrollable dialog

public static void viewHistory() {

StringBuilder history = new StringBuilder();

try (BufferedReader reader = new BufferedReader(new FileReader(FILE\_NAME))) {

String line;

while ((line = reader.readLine()) != null) {

history.append(line).append("\n");

}

} catch (IOException e) {

history.append("No history found.");

}

JTextArea textArea = new JTextArea(history.toString());

textArea.setEditable(false);

JScrollPane scrollPane = new JScrollPane(textArea);

scrollPane.setPreferredSize(new java.awt.Dimension(400, 300));

JOptionPane.showMessageDialog(null, scrollPane, "Game History", JOptionPane.INFORMATION\_MESSAGE);

}

// Clear all history from the file

public static void clearHistory() {

try (PrintWriter writer = new PrintWriter(FILE\_NAME)) {

writer.print(""); // Overwrite file with empty content

JOptionPane.showMessageDialog(null, "History cleared successfully.");

} catch (IOException e) {

e.printStackTrace();

}

}

}