Picture Tic Tac Toe Game

Input :

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

// import java.net.URL;

public class PictureTicTacToe extends JFrame {

    private static final int SIZE = 3;

    private static final int CELL\_SIZE = 200;

    private ImageIcon xIcon;

    private ImageIcon oIcon;

    private ImageIcon emptyIcon;

    private JButton[][] cells;

    private char[][] board;

    private char currentPlayer;

    public PictureTicTacToe() {

        setTitle("Picture Tic Tac Toe");

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setResizable(false);

        // Resize images

        xIcon = resizeImage("x.jpeg");

        oIcon = resizeImage("0.jpeg");

        // emptyIcon = resizeImage("empty.png");

        cells = new JButton[SIZE][SIZE];

        board = new char[SIZE][SIZE];

        currentPlayer = 'X';

        initializeBoard();

        initializeGUI();

        pack();

        setLocationRelativeTo(null);

        setVisible(true);

    }

    private ImageIcon resizeImage(String imageName) {

        try {

            // Load original image

            BufferedImage originalImage = ImageIO.read(new File(imageName));

            // Resize image

            int width = CELL\_SIZE;

            int height = CELL\_SIZE;

            Image resizedImage = originalImage.getScaledInstance(width, height, Image.SCALE\_SMOOTH);

            return new ImageIcon(resizedImage);

        } catch (IOException e) {

            e.printStackTrace();

            return null;

        }

    }

    private void initializeBoard() {

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

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

                board[i][j] = '-';

            }

        }

    }

    private void initializeGUI() {

        JPanel panel = new JPanel();

        panel.setLayout(new GridLayout(SIZE, SIZE));

        ActionListener cellListener = new CellListener();

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

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

                cells[i][j] = new JButton(emptyIcon);

                cells[i][j].setPreferredSize(new Dimension(CELL\_SIZE, CELL\_SIZE));

                cells[i][j].addActionListener(cellListener);

                panel.add(cells[i][j]);

            }

        }

        add(panel, BorderLayout.CENTER);

    }

    private boolean hasWon(char symbol) {

        // Check rows and columns

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

            if (board[i][0] == symbol && board[i][1] == symbol && board[i][2] == symbol) {

                return true;

            }

            if (board[0][i] == symbol && board[1][i] == symbol && board[2][i] == symbol) {

                return true;

            }

        }

        // Check diagonals

        if (board[0][0] == symbol && board[1][1] == symbol && board[2][2] == symbol) {

            return true;

        }

        if (board[0][2] == symbol && board[1][1] == symbol && board[2][0] == symbol) {

            return true;

        }

        return false;

    }

    private boolean isBoardFull() {

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

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

                if (board[i][j] == '-') {

                    return false;

                }

            }

        }

        return true;

    }

    private void switchPlayer() {

        currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';

    }

    private class CellListener implements ActionListener {

        @Override

        public void actionPerformed(ActionEvent e) {

            JButton cell = (JButton) e.getSource();

            int row = -1, col = -1;

            boolean found = false;

            // Find the clicked cell's indices

            for (int i = 0; i < SIZE && !found; i++) {

                for (int j = 0; j < SIZE && !found; j++) {

                    if (cell == cells[i][j]) {

                        row = i;

                        col = j;

                        found = true;

                    }

                }

            }

            // If the cell is empty, set it to the current player's symbol

            if (board[row][col] == '-') {

                if (currentPlayer == 'X') {

                    cell.setIcon(xIcon);

                } else {

                    cell.setIcon(oIcon);

                }

                board[row][col] = currentPlayer;

                // Check for win or tie

                if (hasWon(currentPlayer)) {

                    JOptionPane.showMessageDialog(null, "Player " + currentPlayer + " wins!");

                    resetGame();

                } else if (isBoardFull()) {

                    JOptionPane.showMessageDialog(null, "It's a tie!");

                    resetGame();

                } else {

                    switchPlayer();

                }

            } else {

                JOptionPane.showMessageDialog(null, "This cell is already occupied. Choose another one.");

            }

        }

    }

    private void resetGame() {

        initializeBoard();

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

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

                cells[i][j].setIcon(emptyIcon);

            }

        }

        currentPlayer = 'X';

    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(() -> new PictureTicTacToe());

    }

}

Output :

