// Tic Tac Toe Java GUI

import java.awt.Dimension;

import java.awt.Color;

import java.awt.event.\*;

import java.awt.Graphics;

import java.awt.Image;

import java.awt.Font;

import java.awt.Toolkit;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.ImageIcon;

import java.io.\*;

import java.util.Scanner;

public class GamePanel extends JPanel implements ActionListener {

// logic variables

boolean playerX;

boolean gameDone = false;

int winner = -1;

int player1wins = 0, player2wins = 0;

int[][] board = new int[3][3];

// paint variables

int lineWidth = 5;

int lineLength = 270;

int x = 15, y = 100; // location of first line

int offset = 95; // square width

int a = 0;

int b = 5;

int selX = 0;

int selY = 0;

// COLORS

Color turtle = new Color(0x80bdab);

Color orange = new Color(0xfdcb9e);

Color offwhite = new Color(0xf7f7f7);

Color darkgray = new Color(0x3f3f44);

// COMPONENTS

JButton jButton;

// CONSTRUCTOR

public GamePanel() {

Dimension size = new Dimension(420, 300);

setPreferredSize(size);

setMaximumSize(size);

setMinimumSize(size);

addMouseListener(new XOListener());

jButton = new JButton("Play Again?");

jButton.addActionListener(this);

jButton.setBounds(315, 210, 100, 30);

add(jButton);

resetGame();

}

public void resetGame() {

playerX = true;

winner = -1;

gameDone = false;

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

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

board[i][j] = 0; // all spots are empty

}

}

getJButton().setVisible(false);

}

public void paintComponent(Graphics page) {

super.paintComponent(page);

drawBoard(page);

drawUI(page);

drawGame(page);

}

public void drawBoard(Graphics page) {

setBackground(turtle);

page.setColor(darkgray);

page.fillRoundRect(x, y, lineLength, lineWidth, 5, 30);

page.fillRoundRect(x, y + offset, lineLength, lineWidth, 5, 30);

page.fillRoundRect(y, x, lineWidth, lineLength, 30, 5);

page.fillRoundRect(y + offset, x, lineWidth, lineLength, 30, 5);

}

public void drawUI(Graphics page) {

// SET COLOR AND FONT

page.setColor(darkgray);

page.fillRect(300, 0, 120, 300);

Font font = new Font("Helvetica", Font.PLAIN, 20);

page.setFont(font);

// SET WIN COUNTER

page.setColor(offwhite);

page.drawString("Win Count", 310, 30);

page.drawString(": " + player1wins, 362, 70);

page.drawString(": " + player2wins, 362, 105);

// DRAW score X

ImageIcon xIcon = new ImageIcon("orangex.png");

Image xImg = xIcon.getImage();

Image newXImg = xImg.getScaledInstance(27, 27, java.awt.Image.SCALE\_SMOOTH);

ImageIcon newXIcon = new ImageIcon(newXImg);

page.drawImage(newXIcon.getImage(), 44 + offset \* 1 + 190, 47 + offset \* 0, null);

// DRAW score O

page.setColor(offwhite);

page.fillOval(43 + 190 + offset, 80, 30, 30);

page.setColor(darkgray);

page.fillOval(49 + 190 + offset, 85, 19, 19);

// DRAW WHOS TURN or WINNER

page.setColor(offwhite);

Font font1 = new Font("Serif", Font.ITALIC, 18);

page.setFont(font1);

if (gameDone) {

if (winner == 1) { // x

page.drawString("The winner is", 310, 150);

page.drawImage(xImg, 335, 160, null);

} else if (winner == 2) { // o

page.drawString("The winner is", 310, 150);

page.setColor(offwhite);

page.fillOval(332, 160, 50, 50);

page.setColor(darkgray);

page.fillOval(342, 170, 30, 30);

} else if (winner == 3) { // tie

page.drawString("It's a tie", 330, 178);

}

} else {

Font font2 = new Font("Serif", Font.ITALIC, 20);

page.setFont(font2);

page.drawString("It's", 350, 160);

if (playerX) {

page.drawString("X 's Turn", 325, 180);

} else {

page.drawString("O 's Turn", 325, 180);

}

}

// DRAW LOGO

Image cookie = Toolkit.getDefaultToolkit().getImage("kawaii.png");

page.drawImage(cookie, 345, 235, 30, 30, this);

Font c = new Font("Courier", Font.BOLD + Font.CENTER\_BASELINE, 13);

page.setFont(c);

page.drawString("Just One Byte", 310, 280);

}

public void drawGame(Graphics page) {

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

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

if (board[i][j] == 0) {

} else if (board[i][j] == 1) {

ImageIcon xIcon = new ImageIcon("orangex.png");

Image xImg = xIcon.getImage();

page.drawImage(xImg, 30 + offset \* i, 30 + offset \* j, null);

} else if (board[i][j] == 2) {

page.setColor(offwhite);

page.fillOval(30 + offset \* i, 30 + offset \* j, 50, 50);

page.setColor(turtle);

page.fillOval(40 + offset \* i, 40 + offset \* j, 30, 30);

}

}

}

repaint();

}

public void checkWinner() {

if (gameDone == true) {

System.out.print("gameDone");

return;

}

// vertical

int temp = -1;

if ((board[0][0] == board[0][1])

&& (board[0][1] == board[0][2])

&& (board[0][0] != 0)) {

temp = board[0][0];

} else if ((board[1][0] == board[1][1])

&& (board[1][1] == board[1][2])

&& (board[1][0] != 0)) {

temp = board[1][1];

} else if ((board[2][0] == board[2][1])

&& (board[2][1] == board[2][2])

&& (board[2][0] != 0)) {

temp = board[2][1];

// horizontal

} else if ((board[0][0] == board[1][0])

&& (board[1][0] == board[2][0])

&& (board[0][0] != 0)) {

temp = board[0][0];

} else if ((board[0][1] == board[1][1])

&& (board[1][1] == board[2][1])

&& (board[0][1] != 0)) {

temp = board[0][1];

} else if ((board[0][2] == board[1][2])

&& (board[1][2] == board[2][2])

&& (board[0][2] != 0)) {

temp = board[0][2];

// diagonal

} else if ((board[0][0] == board[1][1])

&& (board[1][1] == board[2][2])

&& (board[0][0] != 0)) {

temp = board[0][0];

} else if ((board[0][2] == board[1][1])

&& (board[1][1] == board[2][0])

&& (board[0][2] != 0)) {

temp = board[0][2];

} else {

// CHECK FOR A TIE

boolean notDone = false;

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

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

if (board[i][j] == 0) {

notDone = true;

break;

}

}

}

if (notDone == false) {

temp = 3;

}

}

if (temp > 0) {

winner = temp;

if (winner == 1) {

player1wins++;

System.out.println("winner is X");

} else if (winner == 2) {

player2wins++;

System.out.println("winner is O");

} else if (winner == 3) {

System.out.println("It's a tie");

}

gameDone = true;

getJButton().setVisible(true);

}

}

public JButton getJButton() { return jButton; }

public void setPlayerXWins(int a) {

player1wins = a;

}

public void setPlayerOWins(int a) {

player2wins = a;

}

public static void main(String[] args) {

JFrame frame = new JFrame("Tic Tac Toe");

frame.getContentPane();

GamePanel gamePanel = new GamePanel();

frame.add(gamePanel);

frame.addWindowListener(new WindowAdapter() {

public void windowOpened(WindowEvent e) {

try {

File file = new File("score.txt");

Scanner sc = new Scanner(file);

gamePanel.setPlayerXWins(Integer.parseInt(sc.nextLine()));

gamePanel.setPlayerOWins(Integer.parseInt(sc.nextLine()));

sc.close();

} catch (IOException io) {

// file doesnt exist

File file = new File("score.txt");

}

}

public void windowClosing(WindowEvent e) {

try {

PrintWriter pw = new PrintWriter("score.txt");

pw.write("");

pw.write(gamePanel.player1wins + "\n");

pw.write(gamePanel.player2wins + "\n");

pw.close();

} catch (FileNotFoundException e1) { }

}

});

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setResizable(false);

frame.pack();

frame.setVisible(true);

}

private class XOListener implements MouseListener {

public void mouseClicked(MouseEvent event) {

selX = -1;

selY = -1;

if (gameDone == false) {

a = event.getX();

b = event.getY();

int selX = 0, selY = 0;

if (a > 12 && a < 99) {

selX = 0;

} else if (a > 103 && a < 195) {

selX = 1;

} else if (a > 200 && a < 287) {

selX = 2;

} else {

selX = -1;

}

if (b > 12 && b < 99) {

selY = 0;

} else if (b > 103 && b < 195) {

selY = 1;

} else if (b > 200 && b < 287) {

selY = 2;

} else {

selY = -1;

}

if (selX != -1 && selY != -1) {

if (board[selX][selY] == 0) {

if (playerX) {

board[selX][selY] = 1;

playerX = false;

} else {

board[selX][selY] = 2;

playerX = true;

}

checkWinner();

System.out.println(" CLICK= x:" + a + ",y: " + b + "; selX,selY: " + selX + "," + selY);

}

} else {

System.out.println("invalid click");

}

}

}

public void mouseReleased(MouseEvent event) {}

public void mouseEntered(MouseEvent event) {}

public void mouseExited(MouseEvent event) {}

public void mousePressed(MouseEvent event) {}

}

@Override

public void actionPerformed(ActionEvent e) { resetGame(); }

}