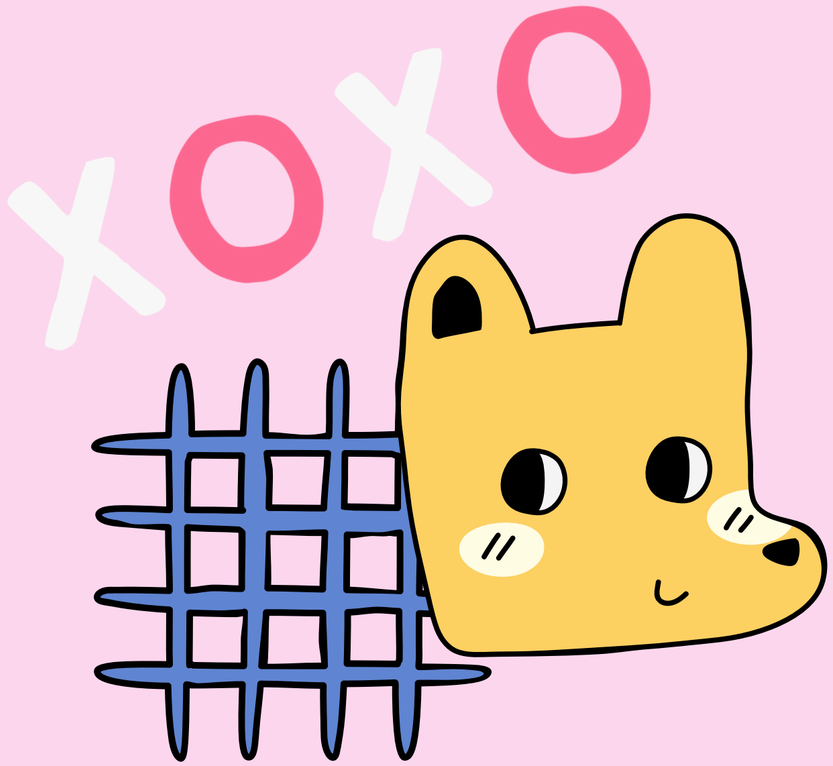
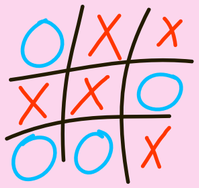


TIC_TAC_TOE



SHAHED ISLIM
202220441

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
```

1. import java.awt.*; : This package provides classes for creating and managing graphical user interface components such as windows , buttons, text fieleds,etc.
2. import java.awt.event.*; :This package contains event classes and listener event classes and listener interactiones for handling various user interactions such as mouse clicks,key presses,and other actons performed on GUI.
3. import javax..swing.*; : IT provides a setof ‘lightweight’ components that work the same on all platforms.(JFrame ,JPanel,JButton,JLabel,and more.)

```
public class TicTacToe {
    int boardWidth = 600;
    int boardHeight = 650;
    JFrame frame = new JFrame("Tic-
Tac-Toe");
    JLabel textLabel = new
JLabel();
    JPanel textPanel = new
JPanel();
    JPanel boardPanel = new
JPanel();
    JButton[][] board = new
JButton[3][3];
    String playerX = "X";
    String playerO = "O";
    String currentPlayer = playerX;
    boolean gameOver = false;
    int turns = 0;
}
```

Explaining the variables used to set up the game , the players, and the game state.

```
TicTacToe() {
    frame.setVisible(true);
    frame.setSize(boardWidth,
boardHeight);

frame.setLocationRelativeTo(null);
    frame.setResizable(false);

frame.setDefaultCloseOperation(JFra
me.EXIT_ON_CLOSE);
    frame.setLayout(new
BorderLayout());
}
```

frame.setVisible(true) :Makes the frame visible
frame.setSize(boarwidth,boarHeight) : Set the frame size dimensions .
frame.setLocationRelativeTo(null): The position of the frame in the center of the screen.
frame.setResizable(false): The ability to change the frame size to unchangeable.

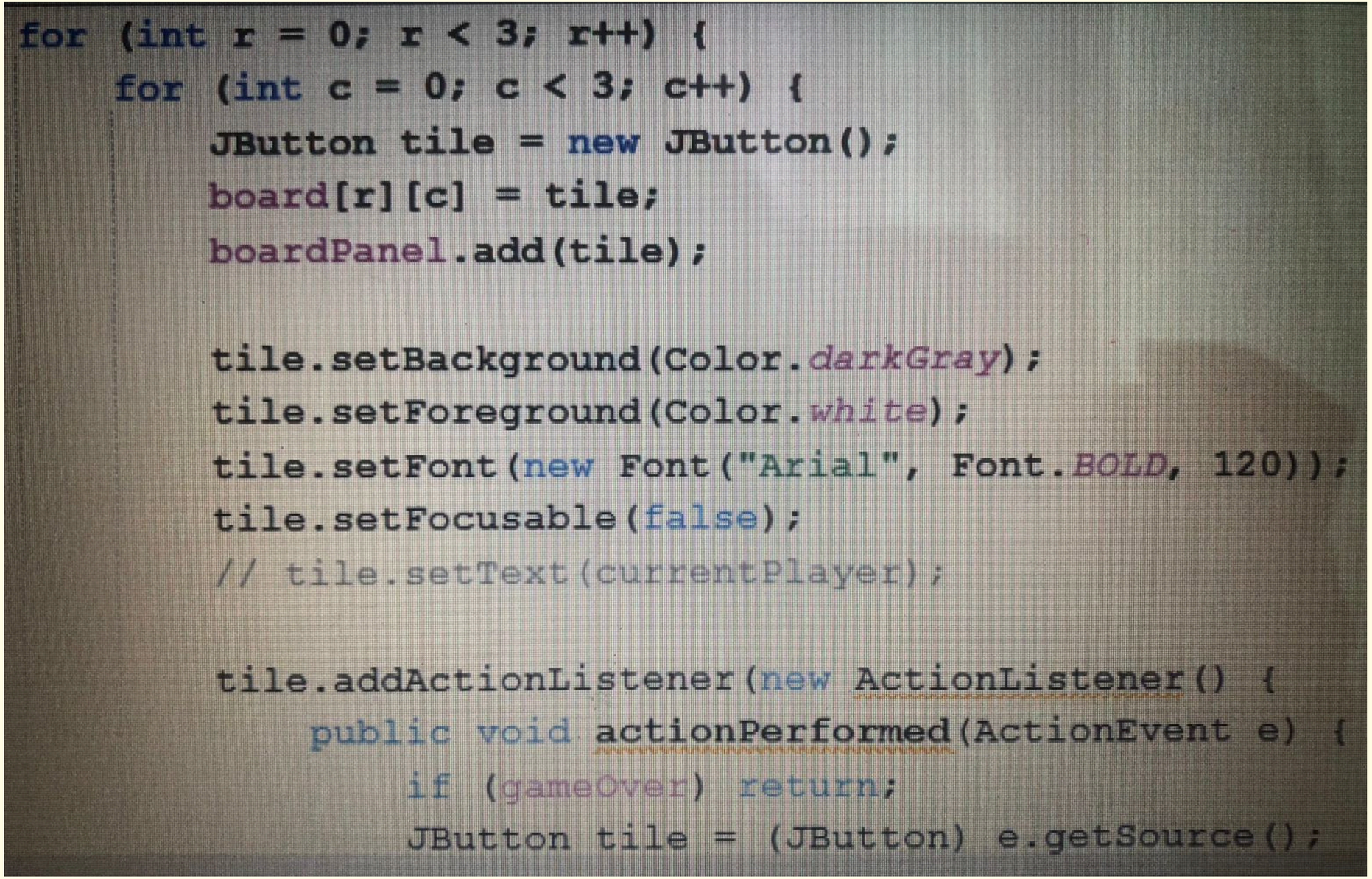
```
textLabel.setBackground(Color.darkG
ray);
textLabel.setForeground(Color.white
);
textLabel.setFont(new Font("Arial",
Font.BOLD, 50));
textLabel.setHorizontalAlignment(JL
abel.CENTER);
textLabel.setText("Tic-Tac-Toe");
textLabel.setOpaque(true);

textPanel.setLayout(new
BorderLayout());
textPanel.add(textLabel);
frame.add(textPanel,
BorderLayout.NORTH);
```

Sets the background color of the label to dark gray.
Sets the text color inside the label to white.
Sets the font type , size and style for the label text.
Makes the label opaque , meaning the background color will be visible (without this, the background would be transparent).

```
boardPanel.setLayout(new
GridLayout(3, 3));
boardPanel.setBackground(Color.dark
Gray);
frame.add(boardPanel);
```

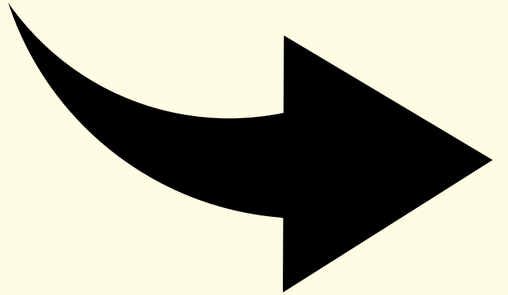
Sets the layout manager of the panel (JPalel)to a GridLayout with 3 rows and 3 columns.
Sets the background color of the panel to dark gray .
(The background color of the panel where the game buttons will be placed.



```
for (int r = 0; r < 3; r++) {
    for (int c = 0; c < 3; c++) {
        JButton tile = new JButton();
        board[r][c] = tile;
        boardPanel.add(tile);

        tile.setBackground(Color.darkGray);
        tile.setForeground(Color.white);
        tile.setFont(new Font("Arial", Font.BOLD, 120));
        tile.setFocusable(false);
        // tile.setText(currentPlayer);

        tile.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if (gameOver) return;
                JButton tile = (JButton) e.getSource();
```



- Starts an outer loop (three rows of the board).
- Starts an inner loop (three columns of each row).
- Sets the background color of the button to dark gray.
- Sets the text color inside the button to white.
- Sets the font type , size , and style for the text inside the button .
- Removes focus from the button,meaning it cannot be selected using the keyboard.

- Add an action lisener to the button. When the button is clicked, the code inside actionPerformed will execute.
- Defines the actionPerformed method which specifies what happens when the button is clicked.
- If the game is over(gameOver is true),the method returns and nothing happens.
- Gets the source of the event(the button that was clicked) and casts it to a JButton .
- Checks if the text inside the button is empty(the button has not been clicked yet).If it is empty, the rest of the code to handle clicks can execute.

```
void checkWinner() {
    // horizontal
    for (int r = 0; r < 3; r++) {
        if (board[r]
[0].getText().equals("")) continue;
        if (board[r]
[0].getText().equals(board[r]
[1].getText()) &&
            board[r]
[1].getText().equals(board[r]
[2].getText())) {
            for (int i = 0; i < 3;
i++) {
                setWinner(board[r]
[i]);
            }
            gameOver = true;
            return;
        }
    }
```

- Loop to check the three rows in the board.
- If the first cell in the current row is empty, skip this row and move to the next one .
- Checks if all cells in the current row contain the same player’s mark , either “X” or “O”.
- gameOver=true; : to indicate that the game has ended with a winner.

```
    if (board[0]
[2].getText().equals(board[1]
[1].getText()) &&
        board[1]
[1].getText().equals(board[2]
[0].getText()) &&
        !board[0]
[2].getText().equals("")) {
        setWinner(board[0][2]);
        setWinner(board[1][1]);
        setWinner(board[2][0]);
        gameOver = true;
        return;
    }

    if (turns == 9) {
        for (int r = 0; r < 3; r++)
{
            for (int c = 0; c < 3;
c++) {
                setTie(board[r]
[c]);
            }
        }
        gameOver = true;
    }
```

- *Checks if the text in the first cell of the anti-diagonal is equal to the text in the second cell (the middle one).*
- - Checks if the text in the second cell of the anti-diagonal (the middle one) is equal to the text in the third cell (the last one).
- - Checks if the first cell in the anti-diagonal is not empty.
- - Checks if the number of turns (clicks) has reached 9, indicating that the board is full and no more moves are possible.

```
void setWinner(JButton tile) {

tile.setForeground(Color.green);
    tile.setBackground(Color.gray);
    textLabel.setText(currentPlayer
+ " is the winner!");
}

void setTie(JButton tile) {

tile.setForeground(Color.orange);
    tile.setBackground(Color.gray);
    textLabel.setText("Tie!");
}
```

- Changes the text color of button to green to indicate it is parte of the winning combination.
- Changes the background color of the button to gray .
- Updates the textLabel to display current player is the winner .

- Changes the text color of button to orange to indicate it is parte of the tie.
- Changes the background color of the button to gray .
- Updates the textLabel to display a aessage indicating that the game ended in a tie.

```
public class App {
    public static void
main(String[] args) throws
Exception {
        TicTacToe ticTacToe = new
TicTacToe();
    }
```

ticTacToe: the name of new object.

new TicTacToe() : this is the statement that creates the actual object from TicTacToe class using the default constructor.

THE END