Tugas 8 PBO

Michael Fernandez, 212310060
TI-21-PA

1. Flow aplikasi

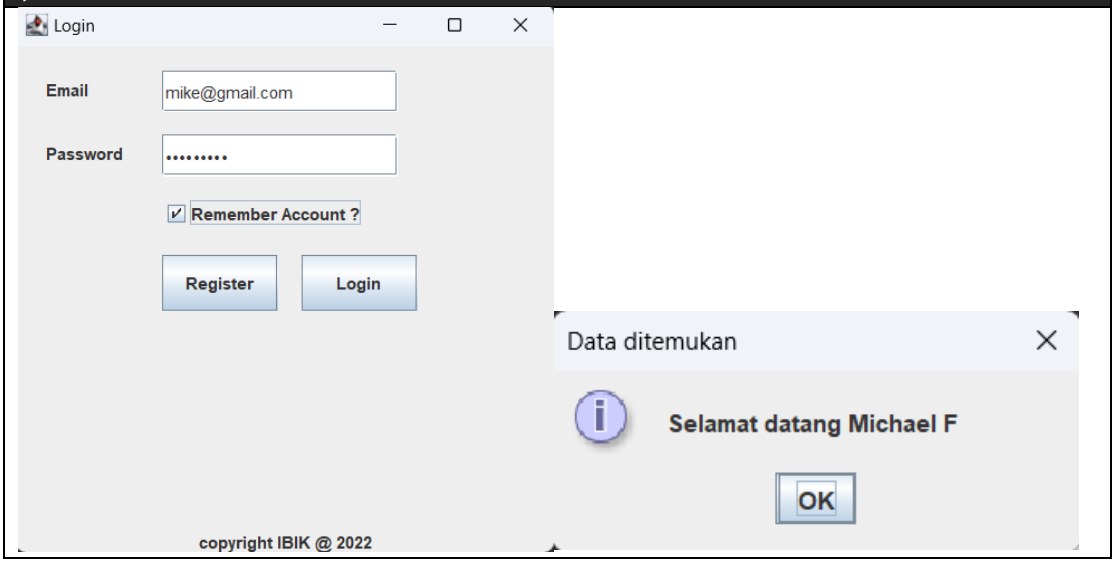
```
package com.ibik.pbo.praktikum;
import java.awt.*;
import javax.swing.*;
import javax.swing.border.Border;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ActionListenerLogin extends JFrame {
      private static final long serialVersionUID = 1L;

      ActionListenerLogin() {
        setVisible(true);
        pack();
        setLocationRelativeTo(null);
        setSize(400, 400);
        setTitle("Login");
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        GenerateUI(this);
    }

    public static void main(String[] args) {
        new ActionListenerLogin();
    }

    private void GenerateUI(ActionListenerLogin frame) {
        JPanel mainPanel = new JPanel();
        mainPanel.setLayout(new BorderLayout());
        frame.setContentPane(mainPanel);

        JPanel panel1 = new JPanel();
        panel1.setLayout(null);
        mainPanel.add(panel1, BorderLayout.NORTH);

        JLabel lblEmail = new JLabel("Email");
        lblEmail.setBounds(22, 23, 80, 16);
        panel1.add((lblEmail));

        final JTextField textEmail = new JTextField();
        textEmail.setBounds(105, 18, 169, 30);
        panel1.add(textEmail);

        JLabel lblPassword = new JLabel("Password");
        lblPassword.setBounds(22, 69, 61, 16);
        panel1.add(lblPassword);

        final JPasswordField textPassword = new JPasswordField();
        textPassword.setBounds(105, 64, 169, 30);
        panel1.add(textPassword);

        JCheckBox chkRemember = new JCheckBox("Remember Account
?");
        chkRemember.setBounds(105, 100, 169, 40);
        panel1.add(chkRemember);
```

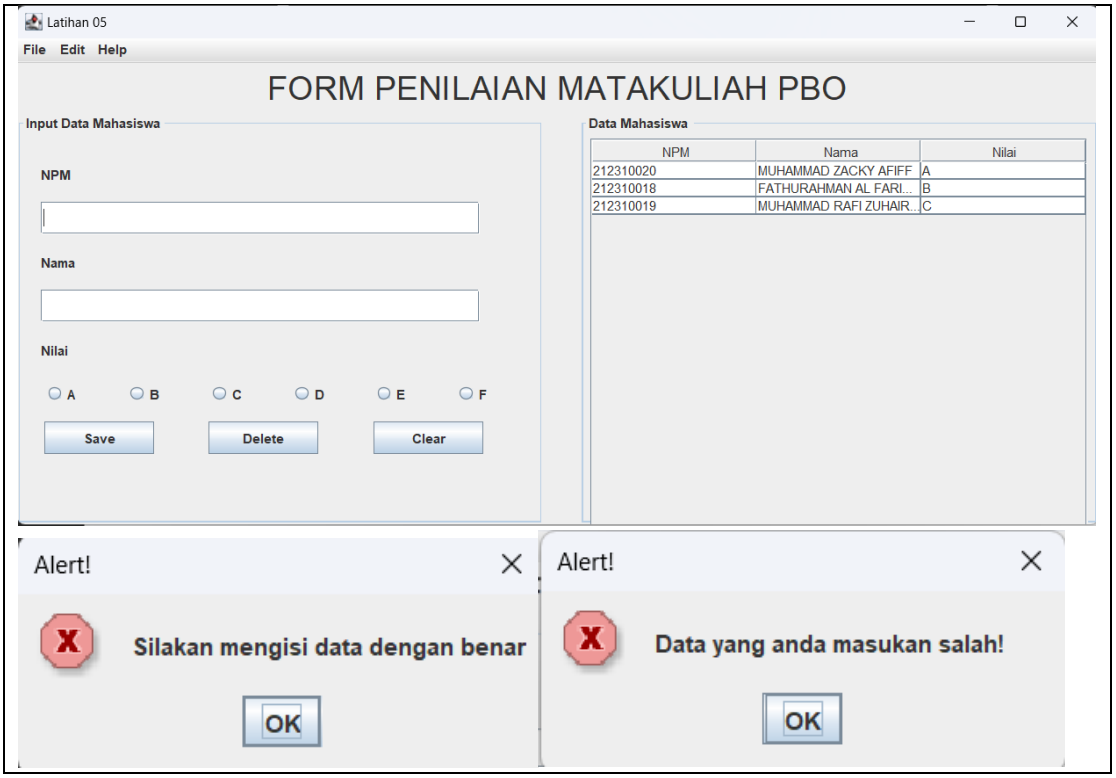
```
        JButton btnRegister = new JButton("Register");
        btnRegister.setBounds(105, 150, 83, 40);
        btnRegister.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                ActionListenerLogin regPage = new
ActionListenerLogin();
                regPage.setVisible(true);
                dispose();
            }
        });
        panel1.add(btnRegister);

        JButton btnLogin = new JButton("Login");
        btnLogin.setBounds(205, 150, 83, 40);
        btnLogin.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String data[] = {
                        "Mike@gmail.com",
                        "212310060",
                        "Michael F"
                };

                String inputedPass = new
String(textPassword.getPassword());

                if (textEmail.getText().isEmpty() ||
inputedPass.isEmpty()) {
                    JOptionPane.showMessageDialog(null, "Silakan
mengisi data dengan benar", "Alert!",
                            JOptionPane.ERROR_MESSAGE);
                } else {
                    if (!textEmail.getText().equals(data[0]) &&
!inputedPass.equals(data[1])) {
                        JOptionPane.showMessageDialog(null, "Data
yang anda masukan salah!", "Alert!",
                                JOptionPane.ERROR_MESSAGE);
                    } else {
                        JOptionPane.showMessageDialog(null,
"Selamat datang " + data[2], "Data ditemukan",
                                JOptionPane.INFORMATION_MESSAGE);

                        // Redirect to Form
                        Latihan01 formPage = new Latihan01();
                        formPage.setVisible(true);
                        dispose();
                    }
                }
            }
        });
        panel1.add(btnLogin);
        JLabel lblCopyright = new JLabel("copyright IBIK @ 2022",
SwingConstants.CENTER);
        lblCopyright.setBackground(Color.BLUE);
        lblCopyright.setSize(300, 50);
        mainPanel.add(lblCopyright, BorderLayout.SOUTH);

        frame.add(panel1);
    }
}
```

```java
package com.ibik.pbo.praktikum;
import java.awt.*;
import javax.swing.*;
import javax.swing.border.Border;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Latihan05 extends JFrame {
    Latihan05() {
        setVisible(true);
        pack();
        setLocationRelativeTo(null);
        setSize(1000, 480);
        setTitle("Latihan 05");
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        GenerateUI(this);
    }

    public static void main(String[] args) {
        new Latihan05();
    }

    private void GenerateUI(Latihan05 frame) {
        // Menu
        JMenuBar menuBar = new JMenuBar();
        JMenu fileMenu = new JMenu("File");
        JMenu editMenu = new JMenu("Edit");
        JMenu helpMenu = new JMenu("Help");

        menuBar.add(fileMenu);
        menuBar.add(editMenu);
        menuBar.add(helpMenu);

        JMenuItem item1 = new JMenuItem("New");
        JMenuItem item2 = new JMenuItem("Save");
        JMenuItem item3 = new JMenuItem("Exit");

        fileMenu.add(item1);
        fileMenu.add(item2);
        fileMenu.add(item3);

        frame.setJMenuBar(menuBar);

        // Main Panel
        JPanel mainPanel = new JPanel();
        mainPanel.setLayout(new BorderLayout());
        frame.setContentPane(mainPanel);

        // Header Title
        JPanel headerPanel = new JPanel();
        headerPanel.setLayout(new FlowLayout(FlowLayout.CENTER));
        mainPanel.add(headerPanel, BorderLayout.NORTH);
        JLabel headerTitle = new JLabel("FORM PENILAIAN MATAKULIAH
PBO");
        headerTitle.setFont(new Font("Arial", Font.PLAIN, 30));
        headerPanel.add(headerTitle);

        // Start of Form
        Border lftPanelBorder =
BorderFactory.createTitledBorder("Input Data Mahasiswa");
        JPanel leftPanel = new JPanel();
```

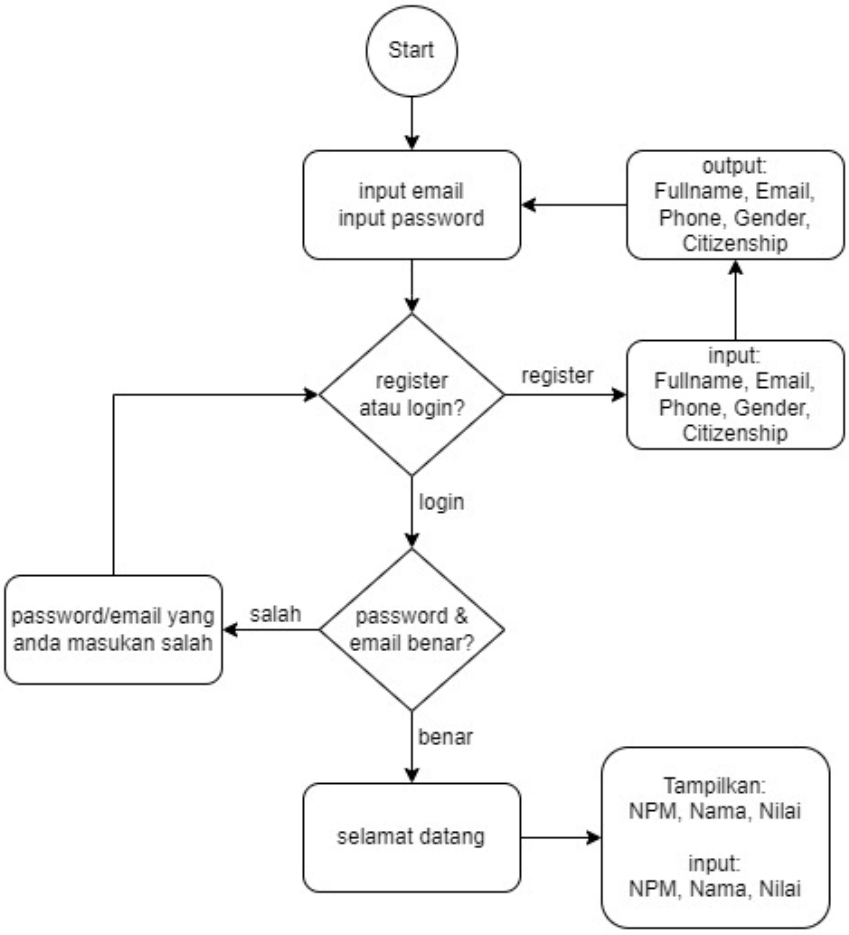
```
        leftPanel.setLayout(null);
        leftPanel.setPreferredSize(new Dimension(480, 480));
        mainPanel.add(leftPanel, BorderLayout.WEST);
        // Start of Panel Content

        // // Input NPM
        JLabel npmLabel = new JLabel("NPM");
        npmLabel.setBounds(22, 40, 80, 30);
        leftPanel.add(npmLabel);

        JTextField inputNPM = new JTextField();
        inputNPM.setBounds(22, 80, 400, 30);
        leftPanel.add(inputNPM);

        // // Input Nama
        JLabel namaLabel = new JLabel("Nama");
        namaLabel.setBounds(22, 120, 80, 30);
        leftPanel.add(namaLabel);

        JTextField inputNama = new JTextField();
        inputNama.setBounds(22, 160, 400, 30);
        leftPanel.add(inputNama);

        // // Input Nilai
        JLabel gradeLabel = new JLabel("Nilai");
        gradeLabel.setBounds(22, 200, 80, 30);
        leftPanel.add(gradeLabel);

        JRadioButton gradeA = new JRadioButton("A");
        gradeA.setBounds(25, 240, 50, 30);
        leftPanel.add(gradeA);

        JRadioButton gradeB = new JRadioButton("B");
        gradeB.setBounds(100, 240, 50, 30);
        leftPanel.add(gradeB);

        JRadioButton gradeC = new JRadioButton("C");
        gradeC.setBounds(175, 240, 50, 30);
        leftPanel.add(gradeC);

        JRadioButton gradeD = new JRadioButton("D");
        gradeD.setBounds(250, 240, 50, 30);
        leftPanel.add(gradeD);

        JRadioButton gradeE = new JRadioButton("E");
        gradeE.setBounds(325, 240, 50, 30);
        leftPanel.add(gradeE);

        JRadioButton gradeF = new JRadioButton("F");
        gradeF.setBounds(400, 240, 50, 30);
        leftPanel.add(gradeF);

        ButtonGroup bg = new ButtonGroup();
        bg.add(gradeA);
        bg.add(gradeB);
        bg.add(gradeC);
        bg.add(gradeD);
        bg.add(gradeE);
        bg.add(gradeF);

        // // Button CRUD Data
```

```
        JButton btnSave = new JButton("Save");
        btnSave.setBounds(25, 280, 100, 30);
        leftPanel.add(btnSave);

        JButton btnDelete = new JButton("Delete");
        btnDelete.setBounds(175, 280, 100, 30);
        leftPanel.add(btnDelete);

        JButton btnClear = new JButton("Clear");
        btnClear.setBounds(325, 280, 100, 30);
        leftPanel.add(btnClear);
        // End of Panel Content
        leftPanel.setBorder(lftPanelBorder);
        // End of Form

        // Table
        Border rPanelBorder =
BorderFactory.createTitledBorder("Data Mahasiswa");
        JPanel rightPanel = new JPanel();
        rightPanel.setLayout(new FlowLayout(FlowLayout.CENTER));
        mainPanel.add(rightPanel, BorderLayout.EAST);

        String data[][] = {
                  {"212310020", "MUHAMMAD ZACKY AFIFF", "A"},
                              {"212310018", "FATHURAHMAN AL
FARIDZI", "B"},
                    {"212310019", "MUHAMMAD RAFI ZUHAIR ATTA", "C"}
        };
        String column[] = {
                "NPM",
                "Nama",
                "Nilai"
        };

        JTable jt = new JTable(data, column);
        jt.setBounds(22, 40, 480, 450);
        JScrollPane sp = new JScrollPane(jt);

        rightPanel.add(sp);

        rightPanel.setBorder(rPanelBorder);
    }
}
```





2. Diagram activity dari flow aplikasi



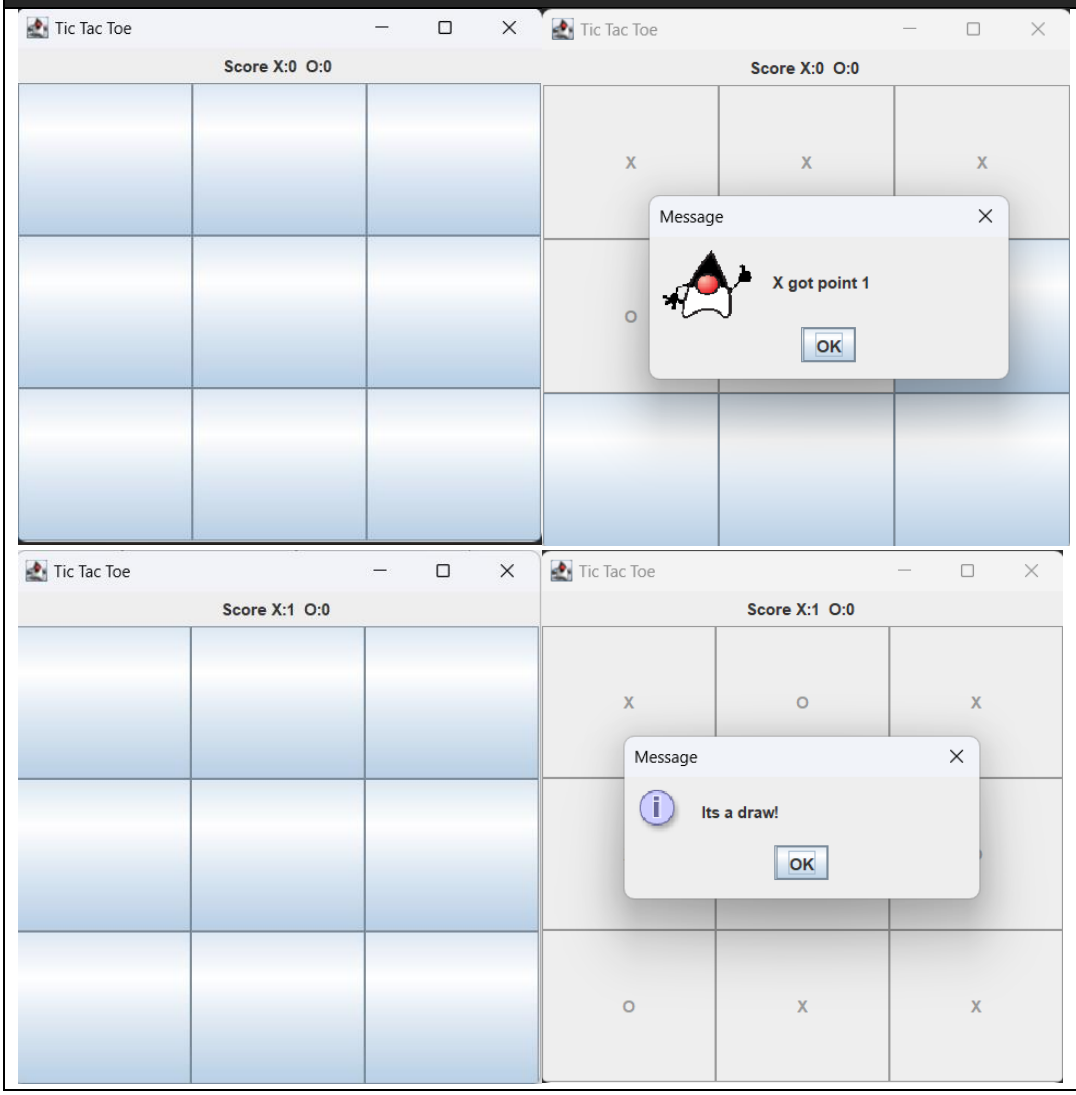
3. TicTacToe

```
package com.ibik.pbo.praktikum;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class TicTacToe extends JFrame {
    private static final long serialVersionUID = 1L;

    // Game board dimensions
    private static final int ROWS = 3;
    private static final int COLS = 3;

    // Game board buttons
    private JButton[][] buttons = new JButton[ROWS][COLS];

    // Scores
    private int playerXScore = 0;
    private int playerOScore = 0;

    // Score labels
    JLabel playerScoreLabel = new JLabel("Score X:" + playerXScore
+ "  O:" + playerOScore + " ");

    // Current player ("X" or "O")
    private String currentPlayer = "X";

    // Constructor
    public TicTacToe() {

      // Set up the game board
      setLayout(new BorderLayout());
        setSize(400, 400);
        setTitle("Tic Tac Toe");
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Add the score labels to the top of the window
        JPanel topPanel = new JPanel();
        topPanel.add(playerScoreLabel);

        add(topPanel, BorderLayout.NORTH);

        // Initialize game board buttons
        JPanel gameBoardPanel = new JPanel(new GridLayout(ROWS,
COLS));
        for (int i = 0; i < ROWS; i++) {
            for (int j = 0; j < COLS; j++) {
                buttons[i][j] = new JButton();
                buttons[i][j].addActionListener(new
ButtonListener());
                gameBoardPanel.add(buttons[i][j]);
            }
        }
        add(gameBoardPanel, BorderLayout.CENTER);

        // Show the game window
        setVisible(true);
    }

    // Main method
```

```
    public static void main(String[] args) {
      new TicTacToe();
    }

    // Button listener class
    private class ButtonListener implements ActionListener {
      @Override
        public void actionPerformed(ActionEvent e) {
            // Get the button that was clicked
            JButton button = (JButton) e.getSource();

            // Set the button text to the current player's symbol
            button.setText(currentPlayer);

            // Disable the button to prevent further clicks
            button.setEnabled(false);

            ImageIcon icon = new ImageIcon ("img/nice.png");
            Image image = icon.getImage();
            Image scaledImage = image.getScaledInstance(65, 50,
Image.SCALE_DEFAULT);
            Icon scaledIcon = new ImageIcon( scaledImage );

            // Check if the current player has won
            if (checkForWin()) {
                // Display a message indicating that the current
player has won
                JOptionPane.showMessageDialog(null, currentPlayer +
" got point 1", "Message", JOptionPane.PLAIN_MESSAGE, scaledIcon);
                if (currentPlayer == "X") {
                  playerXScore += 1;
                }else {
                  playerOScore += 1;
                }

                // Reset the game board
                resetGame();
            } else if (checkForDraw() == true) {
                // Display a message indicating that the game is a
draw
                JOptionPane.showMessageDialog(null, "Its a draw!");

                // Reset the game board
                resetGame();
            } else {
                // Switch to the other player
                if (currentPlayer.equals("X")) {
                    currentPlayer = "O";
                } else {
                    currentPlayer = "X";
                }
            }
        }
    }

    // Check if the current player has won
    private boolean checkForWin() {
        // Check rows
        for (int i = 0; i < ROWS; i++) {
            if (checkRowCol(i, 0, 0, 1)) {
                return true;
```

```
            }
        }

        // Check columns
        for (int i = 0; i < COLS; i++) {
            if (checkRowCol(0, i, 1, 0)) {
                return true;
            }
        }

        // Check diagonal from top-left to bottom-right
        if (checkRowCol(0, 0, 1, 1)) {
            return true;
        }

        // Check diagonal from top-right to bottom-left
        if (checkRowCol(0, COLS - 1, 1, -1)) {
            return true;
        }

        // No win was found
        return false;
    }


    // Check for a win in the specified row or column
    private boolean checkRowCol(int startRow, int startCol, int
rowIncrement, int colIncrement) {
        String symbol = buttons[startRow][startCol].getText();

        // Check all elements in the row or column
        for (int i = 0; i < ROWS; i++) {
            int row = startRow + i * rowIncrement;
            int col = startCol + i * colIncrement;
            String buttonText = buttons[row][col].getText();
            if (buttonText.isEmpty() || !buttonText.equals(symbol))
{
                return false;
            }
        }

        // All elements match
        return true;
    }


    // Check if all buttons have been clicked, indicating a draw
    private boolean checkForDraw() {
      int draw = 0;
        for (int i = 0; i < ROWS; i++) {
            for (int j = 0; j < COLS; j++) {
                if (!buttons[i][j].getText().equals("")) {
                    draw += 1;
                }
            }
        }
        if (draw == 9) {
            return true;
        }else {
            return false;
        }
```

```
    }

    // Reset the game board
    private void resetGame() {
        currentPlayer = "X";
        for (int i = 0; i < ROWS; i++) {
            for (int j = 0; j < COLS; j++) {
                buttons[i][j].setText("");
                buttons[i][j].setEnabled(true);
            }
        }
     // Reset the scores
        playerScoreLabel.setText("Score X:" + playerXScore + "  O:"
+ playerOScore + " ");
    }
}
```



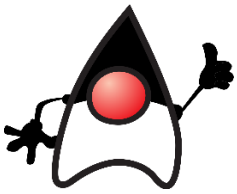
4. Program Rocket

```
package com.ibik.pbo.praktikum;
import java.awt.Color;
import java.awt.Image;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class RocketKeyListener extends JFrame implements
KeyListener {
    private JLabel rocket;
    private int x = 110;
    private int y = 150;
    private int speed = 10;

    public RocketKeyListener() {
        setTitle("Contoh Key Listener");
        setResizable(false);
        setSize(565, 438);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getContentPane().setBackground(Color.BLUE);
        getContentPane().setLayout(null);
        setLocationRelativeTo(null);


        // Load image as icon
        rocket = new JLabel("");
        Image icon = new ImageIcon
(this.getClass().getResource("/rocketright.png")).getImage();
        rocket.setIcon(new ImageIcon (icon));
        rocket.setBounds(x, y, 150, 110);
        getContentPane().add(rocket);
        addKeyListener(this);
    }

    @Override
    public void keyPressed(KeyEvent e) {
        int keyCode = e.getKeyCode();

        // Move left
        if (keyCode == KeyEvent.VK_A || keyCode ==
KeyEvent.VK_LEFT) {
            x -= speed;
        }
        // Move right
        else if (keyCode == KeyEvent.VK_D || keyCode ==
KeyEvent.VK_RIGHT) {
            x += speed;
        }
        // Move up
        else if (keyCode == KeyEvent.VK_W || keyCode ==
KeyEvent.VK_UP) {
            y -= speed;
        }
        // Move down
        else if (keyCode == KeyEvent.VK_S || keyCode ==
KeyEvent.VK_DOWN) {
            y += speed;
        }
```

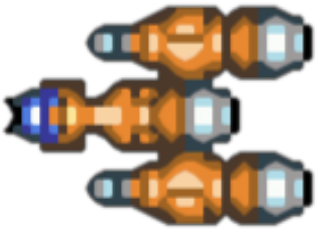
```
        // Check if object is at the edge of the frame
        if (x < 0) {
            // Change object to another image
            Image icon = new ImageIcon
(this.getClass().getResource("/rocketright.png")).getImage();
            rocket.setIcon(new ImageIcon (icon));
            x = 0;
        } else if (x > 405) {
            // Change object to another image
            Image icon = new ImageIcon
(this.getClass().getResource("/rocketleft.png")).getImage();
            rocket.setIcon(new ImageIcon (icon));
            x = 405;
        } else if (y < 0 || y > 290) {
            // Change object to another image
            y = rocket.getY();
        }


        // Update label position
        rocket.setBounds(x, y, rocket.getWidth(),
rocket.getHeight());
    }

    @Override
    public void keyReleased(KeyEvent e) {}

    @Override
    public void keyTyped(KeyEvent e) {}

    public static void main(String[] args) {
        RocketKeyListener frame = new RocketKeyListener();
        frame.setVisible(true);
    }
}
```

Source file untuk gambar rocket:

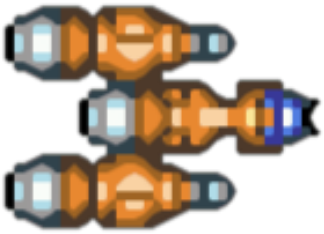
- nice.png

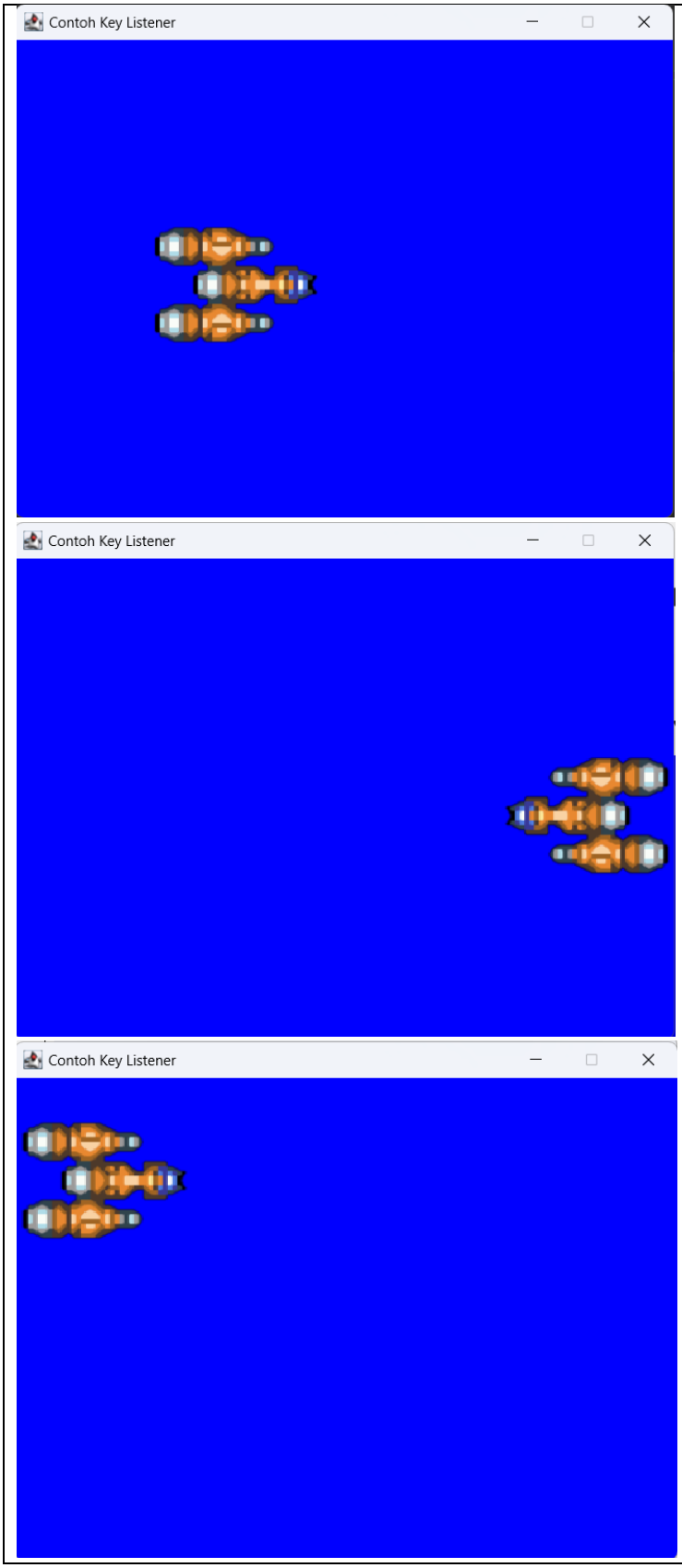


- rocketleft.png



- rocketright.png




Contoh Key Listener
Contoh Key Listener
Contoh Key Listener