Ini merupaka kodingan dari progress projek game puzzle saya:

```
package gamepuzzle;



import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Graphics2D;
import java.awt.GridLayout;
import java.awt.Image;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.awt.image.CropImageFilter;
import java.awt.image.FilteredImageSource;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.AbstractAction;
import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
```

```
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

class MyButton extends JButton {

    private boolean isLastButton;

    public MyButton() {

        super();

        initUI();
    }

    public MyButton(Image image) {

        super(new ImageIcon(image));

        initUI();
    }

    private void initUI() {

        isLastButton = false;
        BorderFactory.createLineBorder(Color.gray);
```

```
        addMouseListener(new MouseAdapter() {

            @Override
            public void mouseEntered(MouseEvent e) {
                setBorder(BorderFactory.createLineBorder(Color.yellow));
            }

            @Override
            public void mouseExited(MouseEvent e) {
                setBorder(BorderFactory.createLineBorder(Color.gray));
            }
        });
    }

    public void setLastButton() {

        isLastButton = true;
    }

    public boolean isLastButton() {

        return isLastButton;
    }
}

public class PuzzleEx extends JFrame {

    private JPanel panel;
    private BufferedImage source;
```

```
private BufferedImage resized;
private Image image;
private MyButton lastButton;
private int width, height;

private List<MyButton> buttons;
private List<Point> solution;

private final int NUMBER_OF_BUTTONS = 12;
private final int DESIRED_WIDTH = 300;

public PuzzleEx() {

    initUI();
}

private void initUI() {

    solution = new ArrayList<>();

    solution.add(new Point(0, 0));
    solution.add(new Point(0, 1));
    solution.add(new Point(0, 2));
    solution.add(new Point(1, 0));
    solution.add(new Point(1, 1));
    solution.add(new Point(1, 2));
    solution.add(new Point(2, 0));
    solution.add(new Point(2, 1));
    solution.add(new Point(2, 2));
```

```
solution.add(new Point(3, 0));
solution.add(new Point(3, 1));
solution.add(new Point(3, 2));

buttons = new ArrayList<>();

panel = new JPanel();
panel.setBorder(BorderFactory.createLineBorder(Color.gray));
panel.setLayout(new GridLayout(4, 3, 0, 0));

try {
   source = loadImage();
   int h = getNewHeight(source.getWidth(), source.getHeight());
   resized = resizeImage(source, DESIRED_WIDTH, h,
         BufferedImage.TYPE_INT_ARGB);

} catch (IOException ex) {
   Logger.getLogger(PuzzleEx.class.getName()).log(
         Level.SEVERE, null, ex);
}

width = resized.getWidth(null);
height = resized.getHeight(null);

add(panel, BorderLayout.CENTER);

for (int i = 0; i < 4; i++) {

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

```
        image = createImage(new FilteredImageSource(resized.getSource(),
                new CropImageFilter(j * width / 3, i * height / 4,
                        (width / 3), height / 4)));

        MyButton button = new MyButton(image);
        button.putClientProperty("position", new Point(i, j));

        if (i == 3 && j == 2) {
            lastButton = new MyButton();
            lastButton.setBorderPainted(false);
            lastButton.setContentAreaFilled(false);
            lastButton.setLastButton();
            lastButton.putClientProperty("position", new Point(i, j));
        } else {
            buttons.add(button);
        }
    }
}

Collections.shuffle(buttons);
buttons.add(lastButton);

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

    MyButton btn = buttons.get(i);
    panel.add(btn);
    btn.setBorder(BorderFactory.createLineBorder(Color.gray));
    btn.addActionListener(new ClickAction());
```

```
    }

    pack();
    setTitle("Puzzle");
    setResizable(false);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

private int getNewHeight(int w, int h) {

    double ratio = DESIRED_WIDTH / (double) w;
    int newHeight = (int) (h * ratio);
    return newHeight;
}

private BufferedImage loadImage() throws IOException {

    BufferedImage bimg = ImageIO.read(new File("src/gambar/oranghutan.jpg"));

    return bimg;
}

private BufferedImage resizeImage(BufferedImage originalImage, int width,
        int height, int type) throws IOException {

    BufferedImage resizedImage = new BufferedImage(width, height, type);
    Graphics2D g = resizedImage.createGraphics();
    g.drawImage(originalImage, 0, 0, width, height, null);
```

```
    g.dispose();

    return resizedImage;
}

private class ClickAction extends AbstractAction {

    @Override
    public void actionPerformed(ActionEvent e) {

        checkButton(e);
        checkSolution();
    }

    private void checkButton(ActionEvent e) {

        int lidx = 0;

        for (MyButton button : buttons) {
            if (button.isLastButton()) {
                lidx = buttons.indexOf(button);
            }
        }

        JButton button = (JButton) e.getSource();
        int bidx = buttons.indexOf(button);

        if ((bidx - 1 == lidx) || (bidx + 1 == lidx)
                || (bidx - 3 == lidx) || (bidx + 3 == lidx)) {
```

```
            Collections.swap(buttons, bidx, lidx);
            updateButtons();
        }
    }

    private void updateButtons() {

        panel.removeAll();

        for (JComponent btn : buttons) {

            panel.add(btn);
        }

        panel.validate();
    }
}

private void checkSolution() {

    List<Point> current = new ArrayList<>();

    for (JComponent btn : buttons) {
        current.add((Point) btn.getClientProperty("position"));
    }

    if (compareList(solution, current)) {
        JOptionPane.showMessageDialog(panel, "Finished",
                "Congratulation", JOptionPane.INFORMATION_MESSAGE);
```

```
        }
    }

    public static boolean compareList(List ls1, List ls2) {

        return ls1.toString().contentEquals(ls2.toString());
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {

            @Override
            public void run() {
                PuzzleEx puzzle = new PuzzleEx();
                puzzle.setVisible(true);
            }
        });
    }
}
```