AED

ALGORITMOS E ESTRUTURAS DE DADOS

2022/2023

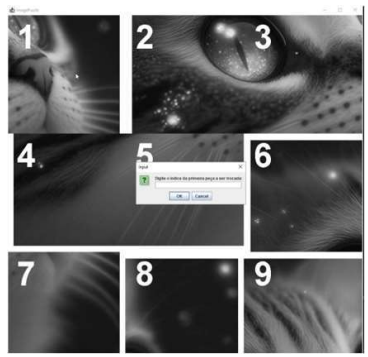
TRABALHO DE AVALIAÇÃO INTERMÉDIA

PARADIGMA DE PROGRAMAÇÃO ORIENTADA POR OBJETOS
MANIPULAÇÃO DE VETORES E MATRIZES

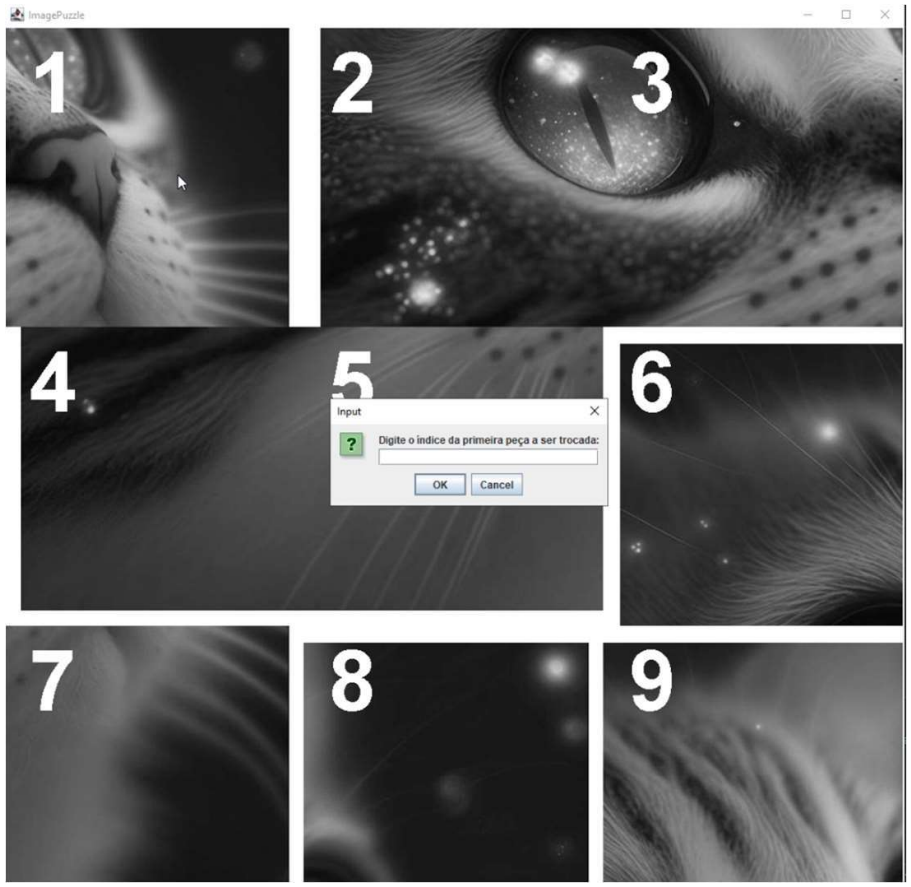
EXTENSÃO DE REQUISITOS DA CLASSE
ConversorImagemAED

ImagePuzzle

Douglas Bewiahn - 50038630



ImagePuzzle



ImagePuzzle

Funcionalidades principais:

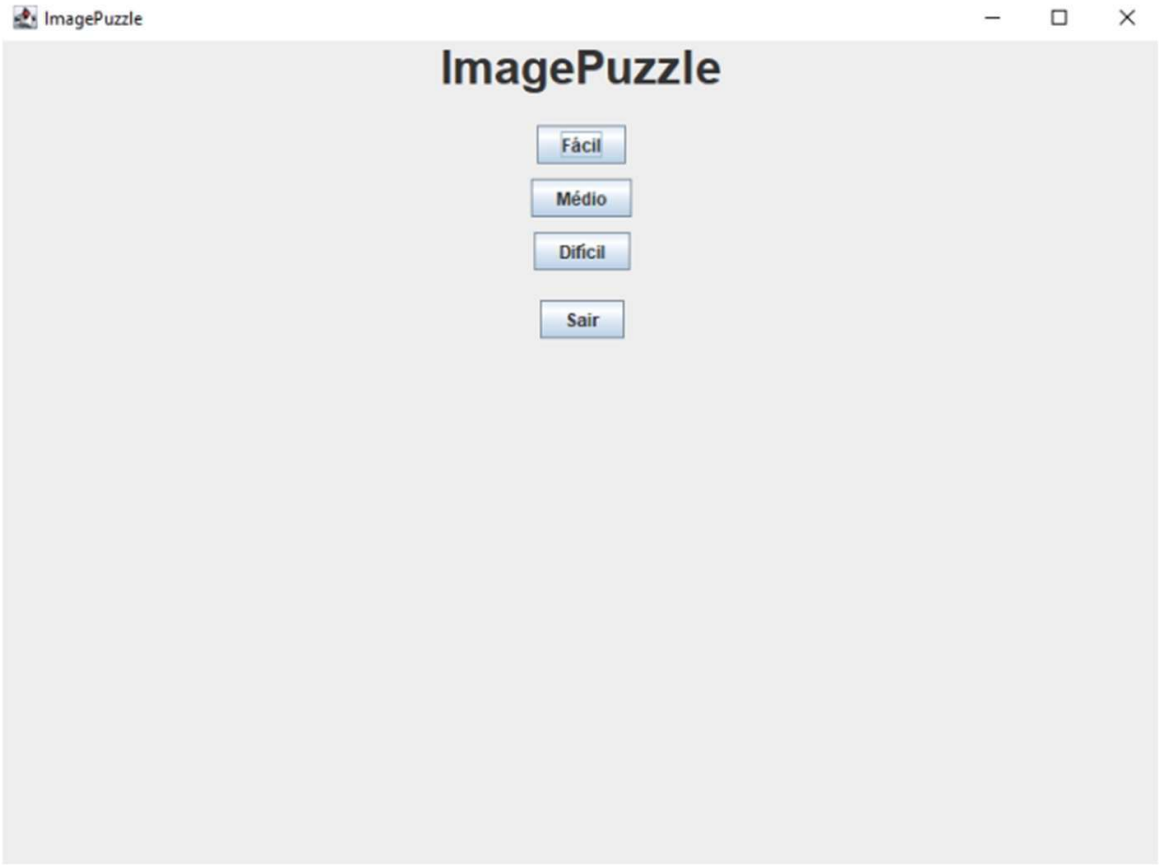
- Escolher dificuldade
- Converter imagem
- Dividir imagem
- Embaralhar peças
- Trocar peças
- Verificar vitória

Classe ImagePuzzle - Botões e Dificuldade

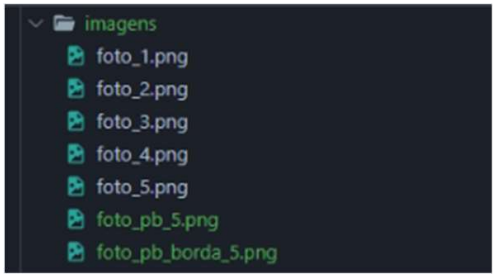
```java
public class ImagePuzzle extends JFrame implements ActionListener {

    private JButton btnFacil, btnMedio, btnDificil, btnSair;
    private JLabel lblTitulo;
    private JPanel pnlMenu;
    private int dificuldade;
    private TabuleiroQuebraCabeca tabuleiro;
    public ImagePuzzle() {
        setTitle(title:"ImagePuzzle");
        setSize(width:800, height:600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(c:null);

        // Inicializa componentes da interface gráfica
        pnlMenu = new JPanel();
        pnlMenu.setLayout(new BoxLayout(pnlMenu, BoxLayout.Y_AXIS));

        lblTitulo = new JLabel(text:"Quebra-Cabeça de Imagem");
        lblTitulo.setFont(new Font(name:"Arial", Font.BOLD, size:32));
        lblTitulo.setAlignmentX(Component.CENTER_ALIGNMENT);

        btnFacil = new JButton(text:"Fácil");
        btnFacil.setAlignmentX(Component.CENTER_ALIGNMENT);
        btnFacil.addActionListener(this);

        btnMedio = new JButton(text:"Médio");
        btnMedio.setAlignmentX(Component.CENTER_ALIGNMENT);
        btnMedio.addActionListener(this);

        btnDificil = new JButton(text:"Difícil");
        btnDificil.setAlignmentX(Component.CENTER_ALIGNMENT);
        btnDificil.addActionListener(this);

        btnSair = new JButton(text:"Sair");
        btnSair.setAlignmentX(Component.CENTER_ALIGNMENT);
        btnSair.addActionListener(this);

        pnlMenu.add(lblTitulo);
        pnlMenu.add(Box.createRigidArea(new Dimension(width:0, height:20)));
        pnlMenu.add(btnFacil);
        pnlMenu.add(Box.createRigidArea(new Dimension(width:0, height:10)));
        pnlMenu.add(btnMedio);
        pnlMenu.add(Box.createRigidArea(new Dimension(width:0, height:10)));
        pnlMenu.add(btnDificil);
        pnlMenu.add(Box.createRigidArea(new Dimension(width:0, height:20)));
        pnlMenu.add(btnSair);

        getContentPane().add(pnlMenu);
    }
```

```java
Run | Debug
public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        ImagePuzzle quebraCabeca = new ImagePuzzle();
        quebraCabeca.setVisible(b:true);
    });
}

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == btnFacil) {
        dificuldade = 6;
        iniciarJogo();
    } else if (e.getSource() == btnMedio) {
        dificuldade = 9;
        iniciarJogo();
    } else if (e.getSource() == btnDificil) {
        dificuldade = 12;
        iniciarJogo();
    } else if (e.getSource() == btnSair) {
        System.exit(status:0);
    }
}
```

Classe ImagePuzzle - Botões e Dificuldade



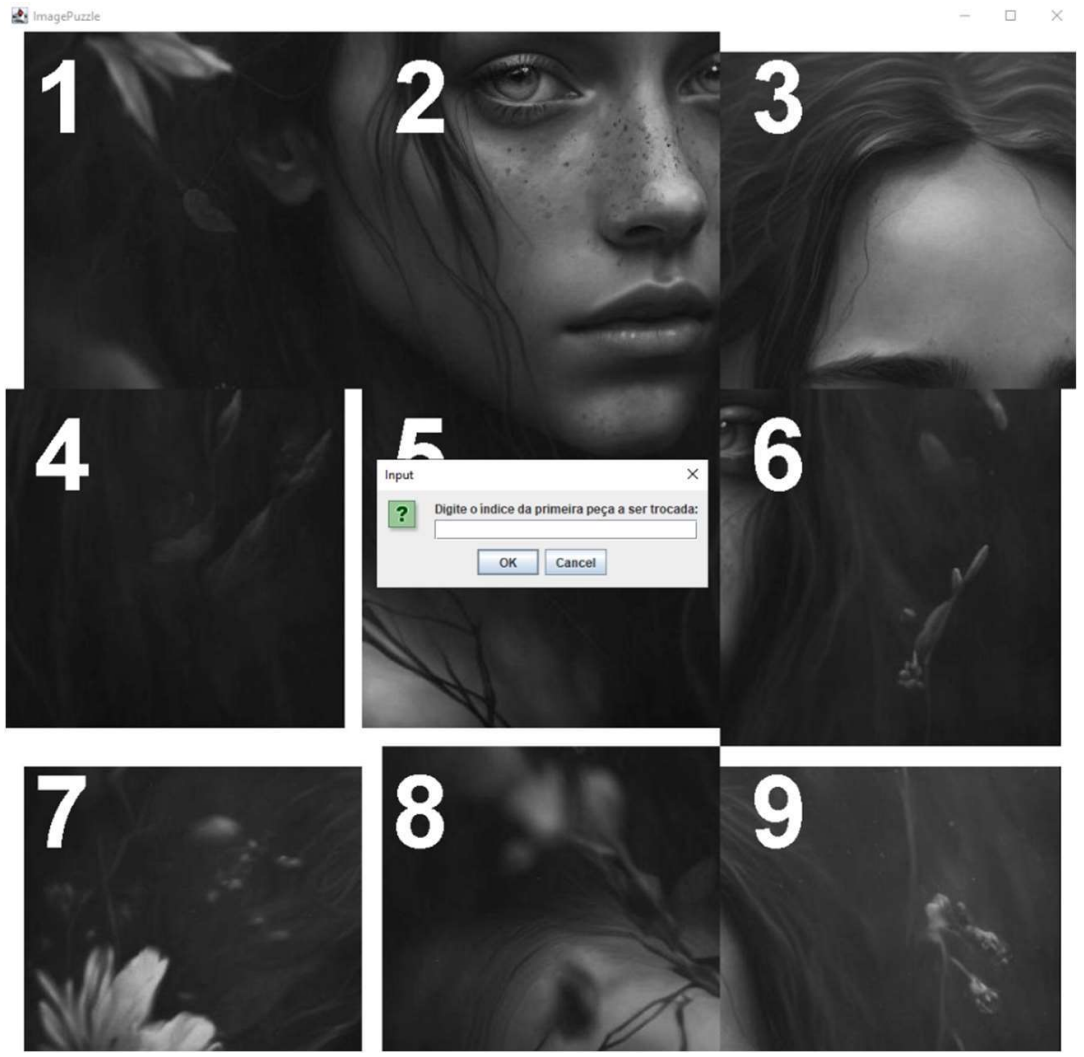
Classe ImagePuzzle - IniciarJogo()




```java
private void iniciarJogo() {
    // Carregar uma imagem aleatória
    Random random = new Random();
    int numeroImagem = random.nextInt(bound:5) + 1;
    String caminhoImagem = "src\\imagens\\foto_"+ numeroImagem + ".png";

    // Converter a imagem para preto e branco
    String caminhoImagemPB = "src\\imagens\\foto_pb_" + numeroImagem + ".png";
    ConversorImagemAED conversor = new ConversorImagemAED(caminhoImagem);
    conversor.converteBW(caminhoImagemPB);

    // Adicionar uma borda à imagem
    String caminhoImagemPBBorda = "src\\imagens\\foto_pb_borda_" + numeroImagem + ".png";
    int tamanhoBorda = 20;
    ConversorImagemAED conversorBorda = new ConversorImagemAED(caminhoImagemPB);
    conversorBorda.criarMargem(tamanhoBorda, caminhoImagemPBBorda);

    // Dividir a imagem de acordo com a dificuldade e embaralhar as peças
    BufferedImage imagemPBBorda = null;

    try {
        imagemPBBorda = ImageIO.read(new File(caminhoImagemPBBorda));
    } catch (IOException e) {
        e.printStackTrace();
        return;
    }

    int linhas;
    if (dificuldade == 6) {
        linhas = 2;
    } else if (dificuldade == 9) {
        linhas = 3;
    } else {
        linhas = 4;
    }

    int colunas = linhas;
    tabuleiro = new TabuleiroQuebraCabeca(imagemPBBorda, linhas, colunas);
    this.getContentPane().add(tabuleiro);
    this.pack();
    this.setLocationRelativeTo(c:null);
    this.setVisible(b:true);

    this.getContentPane().remove(pnlMenu);

    SwingUtilities.invokeLater(this::loopJogo);
}
```

Classe ImagePuzzle - loopJogo()

```
    private void loopJogo() {
        while (true) {
            // Obter entrada do usuário para os índices das peças a serem trocadas
            int indice1 = Integer.parseInt(JOptionPane.showInputDialog(message:"Digite o índice da primeira peça a ser trocada:"));
            int indice2 = Integer.parseInt(JOptionPane.showInputDialog(message:"Digite o índice da segunda peça a ser trocada:"));

            // Trocar as peças e atualizar o tabuleiro do jogo
            tabuleiro.trocarPecas(indice1-1, indice2-1);

            // Verificar se o jogador resolveu o quebra-cabeça
            if (tabuleiro.estaResolvido()) {
                JOptionPane.showMessageDialog(this, message:"Parabéns! Você resolveu o quebra-cabeça!", title:"Vitória",
                        JOptionPane.INFORMATION_MESSAGE);

                // Oferecer ao jogador opções para jogar novamente ou retornar ao menu principal
                Object[] opcoes = { "Jogar novamente", "Retornar ao menu" };
                int opcao = JOptionPane.showOptionDialog(this, message:"O que você gostaria de fazer em seguida?",
                        title:"Jogar novamente ou retornar ao menu",
                        JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE, icon:null, opcoes, opcoes[0]);

                if (opcao == 0) {
                    // Jogar novamente
                    this.getContentPane().removeAll();
                    iniciarJogo();
                    break;
                } else {
                    // Retornar ao menu principal
                    this.getContentPane().removeAll();
                    getContentPane().add(pnlMenu);
                    this.pack();
                    this.setLocationRelativeTo(c:null);
                    this.setVisible(b:true);
                    setSize(width:800, height:600);
                    break;
                }
            }
        }
    }
}
```

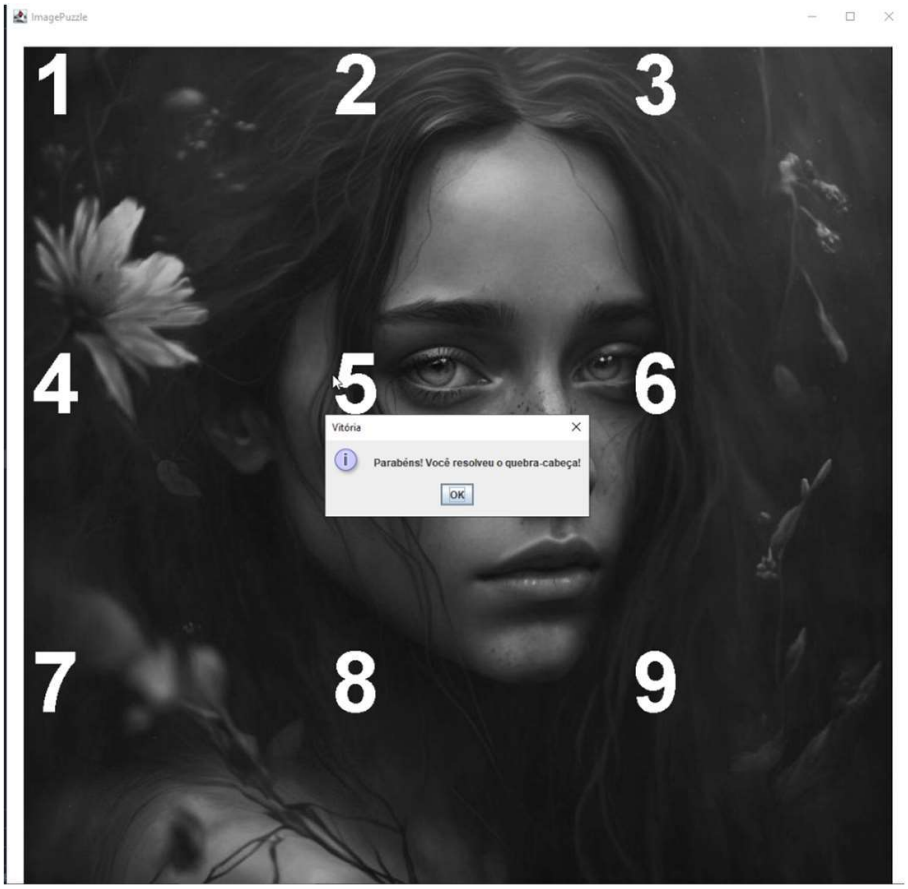

ImagePuzzle
1
2
3
4
5
6
7
8
9
Input
Digite o índice da primeira peça a ser trocada:
OK
Cancel

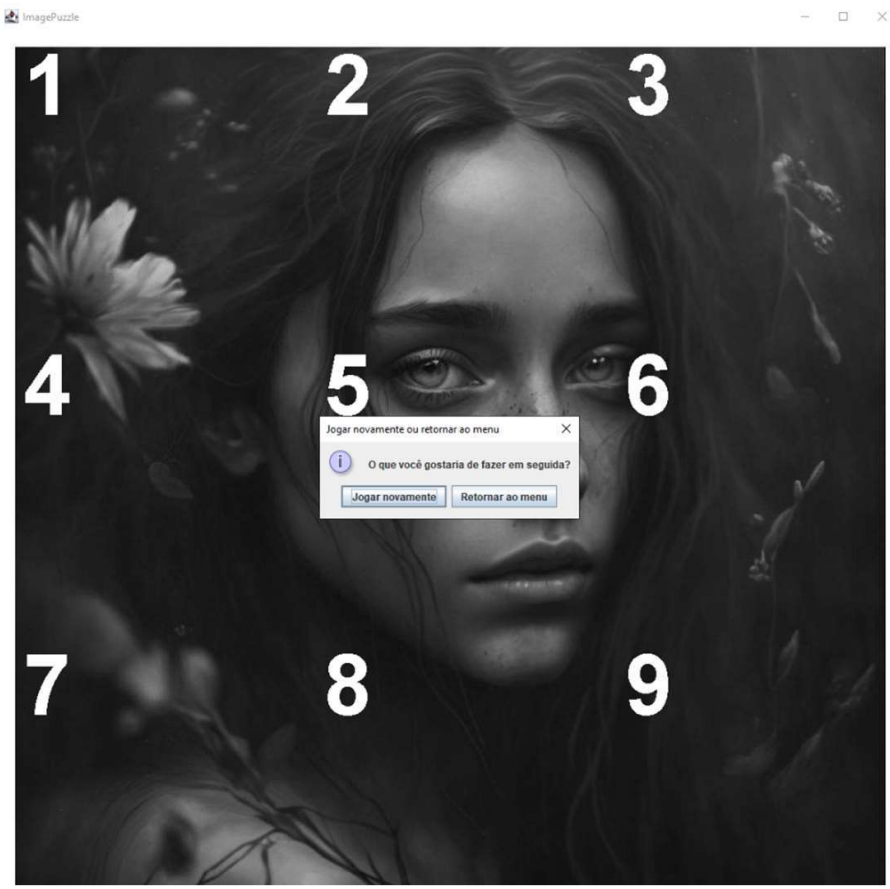
Classe TabuleiroQuebraCabeca

```java
package util;

import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.ArrayList;
import java.util.Collections;

public class TabuleiroQuebraCabeca extends JPanel {
    private ArrayList<BufferedImage> pecas;
    private ArrayList<BufferedImage> pecasOriginais;
    private int linhas;
    private int colunas;
    private int larguraPreferida;
    private int alturaPreferida;

    public TabuleiroQuebraCabeca(BufferedImage imagem, int linhas, int colunas) {
        this.linhas = linhas;
        this.colunas = colunas;
        pecas = new ArrayList<>();
        pecasOriginais = new ArrayList<>();
        larguraPreferida = imagem.getWidth();
        alturaPreferida = imagem.getHeight();

        // Divide a imagem em peças
        int larguraPeca = imagem.getWidth() / colunas;
        int alturaPeca = imagem.getHeight() / linhas;

        for (int i = 0; i < linhas; i++) {
            for (int j = 0; j < colunas; j++) {
                BufferedImage peca = imagem.getSubimage(j * larguraPeca, i * alturaPeca, larguraPeca, alturaPeca);
                pecas.add(peca);
                pecasOriginais.add(peca);
            }
        }

        // Embaralha as peças
        Collections.shuffle(pecas);
    }
```

Classe TabuleiroQuebraCabeca

```
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        int larguraPeca = getWidth() / colunas;
        int alturaPeca = getHeight() / linhas;
        g.setFont(new Font(name:"Arial", Font.BOLD, size:100));
        g.setColor(Color.WHITE);

        for (int i = 0; i < linhas; i++) {
            for (int j = 0; j < colunas; j++) {
                int indice = i * colunas + j;
                g.drawImage(pecas.get(indice), j * larguraPeca, i * alturaPeca, larguraPeca, alturaPeca, observer:null);
                g.drawString(Integer.toString(indice+1), j * larguraPeca + 30, i * alturaPeca + 100);
            }
        }
    }

    public void trocarPecas(int indice1, int indice2) {
        Collections.swap(pecas, indice1, indice2);
        repaint();
    }

    public boolean estaResolvido() {
        return pecas.equals(pecasOriginais);
    }
    @Override
    public Dimension getPreferredSize() {
        return new Dimension(larguraPreferida, alturaPreferida);
    }
}
```


ImagePuzzle
1
2
3
4
5
6
7
8
9
Vitória
Parabéns! Você resolveu o quebra-cabeça!
OK


ImagePuzzle
1
2
3
4
5
6
7
8
9
Jogar novamente ou retornar ao menu
O que você gostaria de fazer em seguida?
Jogar novamente
Retornar ao menu

Obrigado