Desenvolvimento de um Aplicativo de Desenho em Java - Abordagem Prática com Swing

O desenvolvimento de aplicações gráficas sempre foi uma área fascinante na programação, permitindo que desenvolvedores criem interfaces interativas e visuais.

Neste artigo, apresentamos um projeto prático: a construção de um aplicativo de desenho utilizando Java Swing. A proposta é explorar conceitos fundamentais de manipulação gráfica, eventos e a arquitetura de aplicações orientadas a objetos, resultando em um software simples, mas funcional.

Por meio deste projeto, abordaremos a criação de um canvas onde os usuários podem desenhar livremente, selecionar cores e utilizar ferramentas básicas como pincéis e formas geométricas. Além disso, implementaremos funcionalidades de desfazer e refazer ações, oferecendo ao usuário uma experiência intuitiva e responsiva.

Este guia é ideal tanto para iniciantes que buscam aprimorar suas habilidades em Java, quanto para desenvolvedores mais experientes que desejam revisar conceitos essenciais de design de interface. Ao final, o leitor terá não apenas uma compreensão sólida dos componentes do Java Swing, mas também um projeto completo que poderá ser expandido e aprimorado.

Estrutura do Projeto

1. Configuração do Ambiente

1. **Instalar o JDK 8**:
   - Baixe e instale o JDK 8 a partir do site oficial da Oracle ou OpenJDK.
2. **Escolher uma IDE**:
   - Utilize uma IDE como **Eclipse**, **IntelliJ IDEA** ou **NetBeans** para facilitar o desenvolvimento.
3. **Criar um novo projeto**:
   - Abra a IDE escolhida e crie um novo projeto Java.

2. **Criar as Classes**

Classe Canvas

Essa classe é responsável pela área de desenho. Ela gerencia os eventos de mouse e a lógica de desenho.

```
package com.educaciencia.paint.swing;

import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.Stack;

public class Canvas extends JPanel {

    private BufferedImage image; // Imagem que contém o desenho
    private Graphics2D g2d; // Graphics2D para desenhar na imagem
    private Stack<BufferedImage> undoStack = new Stack<>(); // Pilha para desfazer ações
    private Stack<BufferedImage> redoStack = new Stack<>(); // Pilha para refazer ações
    private Color currentColor = Color.BLACK; // Cor atual do desenho
    private int currentTool = 1; // Ferramenta atual (1: Pincel, 2: Retângulo)
    private int lastX, lastY; // Últimas coordenadas do mouse

    // Construtor que inicializa o painel de desenho
    public Canvas() {
        setPreferredSize(new Dimension(800, 600)); // Define o tamanho do painel
        image = new BufferedImage(800, 600, BufferedImage.TYPE_INT_ARGB); // Cria uma
nova imagem
        g2d = image.createGraphics(); // Cria o objeto Graphics2D
        g2d.setColor(Color.WHITE); // Define a cor de fundo
        g2d.fillRect(0, 0, 800, 600); // Preenche o fundo com branco
        g2d.setColor(currentColor); // Define a cor atual

        // Adiciona o listener para o mouse
        addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                lastX = e.getX(); // Armazena a posição X do mouse
                lastY = e.getY(); // Armazena a posição Y do mouse
                saveState(); // Salva o estado atual antes de desenhar
            }

            @Override
            public void mouseReleased(MouseEvent e) {
                lastX = e.getX(); // Atualiza a posição X ao soltar o mouse
                lastY = e.getY(); // Atualiza a posição Y ao soltar o mouse
            }
        });

        // Adiciona o listener para o movimento do mouse
        addMouseMotionListener(new MouseMotionAdapter() {
            @Override
            public void mouseDragged(MouseEvent e) {
                if (currentTool == 1) { // Se a ferramenta atual for Pincel
```

```
                g2d.setColor(currentColor); // Define a cor atual
                g2d.drawLine(lastX, lastY, e.getX(), e.getY()); // Desenha uma linha
                lastX = e.getX(); // Atualiza a posição X
                lastY = e.getY(); // Atualiza a posição Y
                repaint(); // Atualiza o painel para mostrar o desenho
            } else if (currentTool == 2) { // Se a ferramenta atual for Retângulo
                g2d.setColor(currentColor); // Define a cor atual
                int width = e.getX() - lastX; // Calcula a largura do retângulo
                int height = e.getY() - lastY; // Calcula a altura do retângulo
                g2d.drawRect(Math.min(lastX, e.getX()), Math.min(lastY, e.getY()),
Math.abs(width), Math.abs(height)); // Desenha o retângulo
                repaint(); // Atualiza o painel
            }
        }
    });
}

// Salva o estado atual da imagem na pilha de desfazer
private void saveState() {
    BufferedImage temp = new BufferedImage(image.getWidth(), image.getHeight(),
image.getType());
    Graphics2D g2dTemp = temp.createGraphics();
    g2dTemp.drawImage(image, 0, 0, null); // Copia a imagem atual
    undoStack.push(temp); // Adiciona à pilha de desfazer
    redoStack.clear(); // Limpa a pilha de refazer ao fazer uma nova ação
}

// Desfazer a última ação
public void undo() {
    if (!undoStack.isEmpty()) {
        BufferedImage temp = undoStack.pop(); // Remove a última imagem da pilha de
desfazer
        redoStack.push(copyImage(image)); // Armazena a imagem atual na pilha de refazer
        image = temp; // Restaura a imagem anterior
        repaint(); // Atualiza o painel
    }
}

// Refazer a última ação
public void redo() {
    if (!redoStack.isEmpty()) {
        BufferedImage temp = redoStack.pop(); // Remove a última imagem da pilha de refazer
        undoStack.push(copyImage(image)); // Armazena a imagem atual na pilha de desfazer
        image = temp; // Restaura a imagem
        repaint(); // Atualiza o painel
    }
}

// Método para copiar a imagem
private BufferedImage copyImage(BufferedImage img) {
    BufferedImage copy = new BufferedImage(img.getWidth(), img.getHeight(),
img.getType());
    Graphics2D g2dCopy = copy.createGraphics();
    g2dCopy.drawImage(img, 0, 0, null); // Copia a imagem
    return copy; // Retorna a cópia da imagem
}

// Carregar uma imagem
public void loadImage(BufferedImage img) {
    image = img; // Define a imagem atual
```

```
        repaint(); // Atualiza o painel
    }

    // Salvar a imagem em um arquivo
    public void saveImage(String filePath) {
        try {
            ImageIO.write(image, "png", new File(filePath)); // Salva a imagem como PNG
        } catch (IOException e) {
            e.printStackTrace(); // Exibe a pilha de erros
        }
    }

    // Método de pintura
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawImage(image, 0, 0, null); // Desenha a imagem no painel
    }

    // Definir a cor atual
    public void setCurrentColor(Color color) {
        this.currentColor = color; // Atualiza a cor atual
    }

    // Definir a ferramenta atual
    public void setCurrentTool(int tool) {
        this.currentTool = tool; // Atualiza a ferramenta atual
    }
}
```

Classe ColorPalette

Esta classe permite ao usuário selecionar cores para o desenho.

```
package com.educaciencia.paint.swing;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ColorPalette extends JPanel {

    // Construtor que cria a paleta de cores
    public ColorPalette(Canvas canvas) {
        setLayout(new FlowLayout()); // Define o layout como um fluxo horizontal

        // Array de cores disponíveis
        Color[] colors = {Color.BLACK, Color.RED, Color.GREEN, Color.BLUE, Color.YELLOW,
Color.CYAN, Color.MAGENTA};

        // Para cada cor, cria um botão
        for (Color color : colors) {
            JButton colorButton = new JButton(); // Cria um novo botão
            colorButton.setBackground(color); // Define a cor de fundo do botão
            colorButton.setPreferredSize(new Dimension(50, 50)); // Define o tamanho do botão
            // Adiciona um listener para o botão
            colorButton.addActionListener(new ActionListener() {
```

```
            @Override
            public void actionPerformed(ActionEvent e) {
                canvas.setCurrentColor(color); // Altera a cor atual do canvas
            }
        });
        add(colorButton); // Adiciona o botão ao painel
    }
  }
}
```

Classe ToolSelector

Esta classe fornece uma interface para selecionar ferramentas de desenho.

```
package com.educaciencia.paint.swing;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ToolSelector extends JPanel {

   // Construtor que cria o seletor de ferramentas
   public ToolSelector(Canvas canvas) {
      setLayout(new FlowLayout()); // Define o layout como um fluxo horizontal

      // Botão para selecionar a ferramenta Pincel
      JButton pencilButton = new JButton("Pincel");
      pencilButton.addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            canvas.setCurrentTool(1); // Define a ferramenta atual como Pincel
         }
      });
      add(pencilButton); // Adiciona o botão ao painel

      // Botão para selecionar a ferramenta Retângulo
      JButton rectangleButton = new JButton("Retângulo");
      rectangleButton.addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            canvas.setCurrentTool(2); // Define a ferramenta atual como Retângulo
         }
      });
      add(rectangleButton); // Adiciona o botão ao painel

      // Botão para desfazer a última ação
      JButton undoButton = new JButton("Desfazer");
      undoButton.addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            canvas.undo(); // Chama o método de desfazer do canvas
         }
      });
      add(undoButton); // Adiciona o botão ao painel

      // Botão para refazer a última ação
```

```
        JButton redoButton = new JButton("Refazer");
        redoButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                canvas.redo(); // Chama o método de refazer do canvas
            }
        });
        add(redoButton); // Adiciona o botão ao painel
    }
}
```

Classe Draw

Esta é a classe principal que inicializa a aplicação.

```
package com.educaciencia.paint.swing;

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

public class Draw extends JFrame {

    // Construtor da classe Draw
    public Draw() {
        setTitle("Aplicativo de Desenho"); // Define o título da janela
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Define a operação de
fechamento
        setLayout(new BorderLayout()); // Define o layout principal

        Canvas canvas = new Canvas(); // Cria uma nova instância do Canvas
        add(canvas, BorderLayout.CENTER); // Adiciona o Canvas ao centro da janela

        // Cria a paleta de cores e adiciona ao lado esquerdo
        ColorPalette colorPalette = new ColorPalette(canvas);
        add(colorPalette, BorderLayout.WEST);

        // Cria o seletor de ferramentas e adiciona ao lado esquerdo
        ToolSelector toolSelector = new ToolSelector(canvas);
        add(toolSelector, BorderLayout.SOUTH);

        pack(); // Ajusta o tamanho da janela
        setVisible(true); // Torna a janela visível
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new Draw(); // Cria uma nova instância do aplicativo de desenho
        });
    }
}
```

3. Compilação e Execução

1. **Compilar o Projeto**:
   - Certifique-se de que todas as classes estão salvas no diretório do projeto.
2. **Executar a Aplicação**:
   - Execute a classe Draw. Você verá a janela do aplicativo de desenho.

4. Testar Funcionalidades

1. **Testar Ferramentas**:
   - Use a ferramenta de pincel para desenhar livremente.
   - Selecione a ferramenta de retângulo e desenhe retângulos na área de desenho.
2. **Testar Seleção de Cores**:
   - Clique em diferentes cores para alterar a cor do desenho.
3. **Testar Undo/Redo**:
   - Desenhe algo, clique em "Desfazer" e "Refazer" para verificar se as ações funcionam corretamente.
4. **Salvar e Carregar**:
   - Adicione funcionalidades para salvar e carregar imagens, se desejar.

Se necessário, faça ajustes nas cores, tamanhos ou na lógica das ferramentas para melhorar a usabilidade.

Esse aplicativo de desenho básico em Java é um ótimo ponto de partida para entender a manipulação de gráficos e eventos em Java Swing.

Você pode expandi-lo adicionando mais ferramentas de desenho, como formas geométricas, preenchimentos e funcionalidades de edição mais avançadas.

Com este guia, você possui um projeto completo de um aplicativo de desenho em Java, incluindo a estrutura do código, a lógica e a explicação detalhada de cada componente.

Essa abordagem não apenas ensina conceitos importantes de programação, mas também oferece uma base sólida para projetos futuros.

EducaCiência FastCode para a comunidade.