Documentação

Documentação do Código: Sistema de Feedback

O código implementa um sistema de feedback utilizando Java Swing para a interface gráfica. A aplicação permite aos usuários fornecerem feedbacks em três categorias: Bom, Médio e Ruim, além de poderem adicionar comentários. O feedback é armazenado e pode ser exportado para um arquivo CSV.

Importações Necessárias:

```
import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.*;
import java.util.ArrayList;
import java.util.List;
```

Essas importações fornecem as classes necessárias para trabalhar com imagens, interface gráfica, eventos e manipulação de arquivos.

Classe Abstrata FeedbackButton:

```
abstract class FeedbackButton extends JButton implements ActionListener {
    // Atributos
    protected FeedbackGUI feedbackGUI;
    protected JLabel countLabel;
    protected String label;
    protected String emoji;
    protected Color color;

    // Construtor
    public FeedbackButton(FeedbackGUI feedbackGUI, JLabel countLabel, String label, String e
        this.feedbackGUI = feedbackGUI;
        this.countLabel = countLabel;
        this.label = label;
        this.emoji = emoji;
        this.color = color;
        setupButton();
    }

    // Configuração do botão
    private void setupButton() {
        setFont(new Font("Segoe UI Emoji", Font.BOLD, 30));
        setBackground(color);
        setForeground(Color.WHITE);
        setFocusPainted(false);
        setPreferredSize(new Dimension(400, 300));
        setOpaque(true);
        setBorderPainted(false);
        setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();

        // Adicionando o emoji
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.weighty = 0.6;
        gbc.anchor = GridBagConstraints.CENTER;
        JLabel labelEmoji = new JLabel(emoji, SwingConstants.CENTER);
        labelEmoji.setFont(new Font("Segoe UI Emoji", Font.PLAIN, 80));
        labelEmoji.setForeground(Color.WHITE);
        add(labelEmoji, gbc);
```

```
        // Adicionando o texto
        gbc.gridy = 1;
        gbc.weighty = 0.6;
        JLabel labelText = new JLabel(label, SwingConstants.CENTER);
        labelText.setFont(new Font("Segoe UI", Font.BOLD, 40));
        labelText.setForeground(Color.WHITE);
        add(labelText, gbc);

        addActionListener(this);
    }

    // Atualização da contagem de feedback
    protected void updateCount() {
        feedbackGUI.incrementCount(label);
        feedbackGUI.updateBackgroundColor();
    }

    // Ação ao clicar no botão
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            String feedbackComment = JOptionPane.showInputDialog(feedbackGUI, "Nos forneça u
            if (feedbackComment != null) {
                feedbackGUI.addFeedbackComment(feedbackComment);
                updateCount();
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

A classe FeedbackButton é uma abstração para os botões de feedback, configurando a aparência e comportamento dos botões de feedback.

Classes Específicas para Botões de Feedback:

```
class BomButton extends FeedbackButton {
    public BomButton(FeedbackGUI feedbackGUI, JLabel countLabel) {
        super(feedbackGUI, countLabel, "Bom", "😊", Color.decode("#4CAF50"));
    }
}

class MedioButton extends FeedbackButton {
    public MedioButton(FeedbackGUI feedbackGUI, JLabel countLabel) {
        super(feedbackGUI, countLabel, "Médio", "😐", Color.decode("#FFC107"));
    }
}

class RuimButton extends FeedbackButton {
    public RuimButton(FeedbackGUI feedbackGUI, JLabel countLabel) {
        super(feedbackGUI, countLabel, "Ruim", "😞", Color.decode("#F44336"));
    }
}
```

Essas classes específicas herdam FeedbackButton e configuram os botões "Bom", "Médio" e "Ruim" com cores e emojis apropriados.

Classe FeedbackCounts:

```
class FeedbackCounts implements Serializable {
    private static final long serialVersionUID = 1L;
    private int bomCount;
    private int medioCount;
    private int ruimCount;

    public FeedbackCounts() {
        this.bomCount = 0;
        this.medioCount = 0;
        this.ruimCount = 0;
    }

    public int getBomCount() {
        return bomCount;
    }

    public void setBomCount(int bomCount) {
        this.bomCount = bomCount;
    }

    public int getMedioCount() {
        return medioCount;
    }

    public void setMedioCount(int medioCount) {
        this.medioCount = medioCount;
    }

    public int getRuimCount() {
        return ruimCount;
    }

    public void setRuimCount(int ruimCount) {
        this.ruimCount = ruimCount;
    }
}
```

Esta classe armazena as contagens de feedbacks para cada categoria.

Classe FeedbackComments

```java
class FeedbackComments implements Serializable {
    private static final long serialVersionUID = 1L;
    private List<String> comments;

    public FeedbackComments() {
        this.comments = new ArrayList<>();
    }

    public List<String> getComments() {
        return comments;
    }

    public void addComment(String comment) {
        comments.add(comment);
    }
}
```

Esta classe armazena os comentários de feedbacks.

Classe Principal FeedbackGUI

Esta classe cria a interface gráfica principal, gerencia a lógica de feedback, carrega e salva dados de feedbacks e configura os elementos da GUI.