**TextCollage Project Documentation**

**Overview**

The TextCollage project is a Java application that allows users to create and manipulate a collage of text items on a canvas. The application provides features for adding, modifying, and saving text items. It uses a graphical user interface (GUI) built with Java Swing components.

**1. Introduction**

The TextCollage project provides a simple and interactive way for users to create artistic collages using text items. Users can add multiple text strings to the canvas, customize their appearance, and save/load collages for later use.

**2. Project Structure**

The project is organized into three main classes:

* **DrawTextItem:** Represents an individual text item on the canvas, including properties such as text content, position, font, color, and background color.
* **DrawTextPanel:** The main panel containing the canvas where text items are drawn. It also handles user interactions and provides a menu bar for various actions.
* **SimpleFileChooser:** A utility class that simplifies file selection operations using Java's JFileChooser.

**3. Features**

**Adding Text**

Users can add text to the canvas by clicking on the drawing area. The text string, font, and color can be customized.

**Saving and Opening Collages**

The application supports saving the current collage to a file and opening existing collages. Collages are saved in a custom file format that includes information about the background color, each text item's content, position, font, and color.

**Editing Text Properties**

Users can customize the appearance of text items by changing their font and background color.

**Undo and Clear**

The application supports undoing the most recent action (removing the last added text item) and clearing the entire canvas.

**4. Usage**

To use the TextCollage application:

1. **Adding Text:**
   * Click on the drawing area to add a text item.
   * Enter the desired text content in the input box.
2. **Saving a Collage:**
   * Select "Save" from the "File" menu.
   * Choose a file location and name to save the collage.
3. **Opening a Collage:**
   * Select "Open" from the "File" menu.
   * Choose the saved file to open the collage.
4. **Customizing Text:**
   * Select a text item by clicking on it.
   * Use the "Set Text Color" and "Set Background Color" options from the "Options" menu.
   * Change the font using the "Set Font..." option.
5. **Undo and Clear:**
   * Use the "Remove Item" option from the "Edit" menu to undo the last action.
   * Use the "Clear" option from the "Edit" menu to remove all text items.

**5. Customization**

Users can customize the appearance of text items in the following ways:

* **Text Color:** Choose the color of the text.
* **Background Color:** Set the background color behind the text.
* **Font:** Select the font for the text.

**6. Error Handling**

The application provides error messages in case of the following issues:

* Attempting to open a file that does not follow the correct format.
* Selecting a file that does not exist.

**7. Conclusion**

The TextCollage project provides a user-friendly interface for creating and customizing text collages. Users can express their creativity by arranging and styling text items on the canvas. The application is designed to be intuitive, with features for both basic and advanced users.

**7. Program**

import java.awt.Dimension;

import java.awt.Toolkit;

import javax.swing.JFrame;

public class TextCollage {

    public static void main(String[] args) {

        JFrame frame = new JFrame("Text Collage");

        DrawTextPanel panel = new DrawTextPanel();

        frame.setContentPane( panel );

        frame.setJMenuBar(panel.getMenuBar());

        frame.setResizable(false);

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.pack();

        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

        frame.setLocation( (screenSize.width - frame.getWidth())/2,

                (screenSize.height - frame.getHeight())/2 );

        frame.setVisible(true);

    }

}

import java.awt.BasicStroke;

import java.awt.Color;

import java.awt.Font;

import java.awt.FontMetrics;

import java.awt.Graphics;

import java.awt.Graphics2D;

public class DrawTextItem {

    private final String string;

    private Font font = null;

    private int x = 0;

    private int y = 0;

    private Color textColor = Color.BLACK;

    private Color background = null;

    private boolean border = false;

    private double rotationAngle = 0;

    private double magnification = 1;

    private double textTransparency = 0;

    private double backgroundTransparency = 0;

    public DrawTextItem(String stringToDraw) {

        this(stringToDraw, 0, 0);

        this.font = new Font("Serif", Font.BOLD, 24);

        this.background = null;

    }

    public DrawTextItem(String stringToDraw, int x, int y) {

        if (stringToDraw == null)

            throw new NullPointerException("String can't be null.");

        string = stringToDraw;

        this.x = x;

        this.y = y;

    }

    public void draw(Graphics g) {

        Graphics2D g2 = (Graphics2D) g.create();

        if (font != null)

            g2.setFont(font);

        FontMetrics fm = g2.getFontMetrics();

        int width = fm.stringWidth(string);

        int height = fm.getAscent() + fm.getDescent();

        g2.translate(x, y);

        if (magnification != 1) {

            float pixelSize = 1 / (float) magnification;

            g2.setStroke(new BasicStroke(pixelSize));

            g2.scale(magnification, magnification);

        }

        if (rotationAngle > 0)

            g2.rotate(-Math.PI \* (rotationAngle / 180));

        Color colorToUseForText = textColor;

        if (colorToUseForText == null)

            colorToUseForText = g2.getColor();

        if (background != null) {

            if (backgroundTransparency == 0)

                g2.setColor(background);

            else

                g2.setColor(new Color(background.getRed(), background.getGreen(), background.getBlue(),

                        (int) (255 \* (1 - backgroundTransparency))));

            g2.fillRect(-width / 2 - 3, -height / 2 - 3, width + 6, height + 6);

        }

        if (textTransparency == 0)

            g2.setColor(colorToUseForText);

        else

            g2.setColor(new Color(colorToUseForText.getRed(),

                    colorToUseForText.getGreen(), colorToUseForText.getBlue(),

                    (int) (255 \* (1 - textTransparency))));

        if (border)

            g2.drawRect(-width / 2 - 3, -height / 2 - 3, width + 6, height + 6);

        g2.drawString(string, -width / 2, -height / 2 + fm.getAscent());

    }

    public String getString() {

        return string;

    }

    public void setBackground(Color background) {

        this.background = background;

    }

    public void setBackgroundTransparency(double backgroundTransparency) {

        if (backgroundTransparency < 0 || backgroundTransparency > 1)

            throw new IllegalArgumentException("Transparency must be in the range 0 to 1.");

        this.backgroundTransparency = backgroundTransparency;

    }

    public void setBorder(boolean border) {

        this.border = border;

    }

    public void setFont(Font font) {

        this.font = font;

    }

    public void setMagnification(double magnification) {

        if (magnification == 0)

            throw new IllegalArgumentException("Magnification cannot be 0.");

        this.magnification = magnification;

    }

    public void setRotationAngle(double rotationAngle) {

        this.rotationAngle = rotationAngle;

    }

    public void setTextColor(Color textColor) {

        this.textColor = textColor;

    }

    public void setTextTransparency(double textTransparency) {

        if (textTransparency < 0 || textTransparency > 1)

            throw new IllegalArgumentException("Transparency must be in the range 0 to 1.");

        this.textTransparency = textTransparency;

    }

    public void setX(int x) {

        this.x = x;

    }

    public void setY(int y) {

        this.y = y;

    }

    public Color getBackground() {

        return background;

    }

    public double getBackgroundTransparency() {

        return backgroundTransparency;

    }

    public boolean getBorder() {

        return border;

    }

    public Font getFont() {

        return font;

    }

    public double getMagnification() {

        return magnification;

    }

    public double getRotationAngle() {

        return rotationAngle;

    }

    public Color getTextColor() {

        return textColor;

    }

    public double getTextTransparency() {

        return textTransparency;

    }

    public int getX() {

        return x;

    }

    public int getY() {

        return y;

    }

}

// package textcollage;

import java.awt.Component;

import java.io.File;

import javax.swing.JFileChooser;

import javax.swing.JOptionPane;

public class SimpleFileChooser {

    private JFileChooser dialog;

    public void setDefaultDirectory() {

        if (dialog != null)

            dialog.setCurrentDirectory(null);

    }

    public void setDefaultDirectory(String directoryName) {

        if (dialog == null)

            dialog = new JFileChooser();

        dialog.setCurrentDirectory(new File(directoryName));

    }

    public void setDefaultDirectory(File directory) {

        if (dialog == null)

            dialog = new JFileChooser();

        dialog.setCurrentDirectory(directory);

    }

    public File getInputFile() {

        return getInputFile(null, null);

    }

    public File getInputFile(Component parent) {

        return getInputFile(parent, null);

    }

    public File getInputFile(Component parent, String dialogTitle) {

        if (dialog == null)

            dialog = new JFileChooser();

        if (dialogTitle != null)

            dialog.setDialogTitle(dialogTitle);

        else

            dialog.setDialogTitle("Select Input File");

        int option = dialog.showOpenDialog(parent);

        if (option != JFileChooser.APPROVE\_OPTION)

            return null;

        File selectedFile = dialog.getSelectedFile();

        return selectedFile;

    }

    public File getOutputFile() {

        return getOutputFile(null, null, null);

    }

    public File getOutputFile(Component parent) {

        return getOutputFile(parent, null, null);

    }

    public File getOutputFile(Component parent, String dialogTitle) {

        return getOutputFile(parent, dialogTitle, null);

    }

    public File getOutputFile(Component parent, String dialogTitle, String defaultFile) {

        if (dialog == null)

            dialog = new JFileChooser();

        if (dialogTitle != null)

            dialog.setDialogTitle(dialogTitle);

        else

            dialog.setDialogTitle("Select Output File");

        if (defaultFile == null)

            dialog.setSelectedFile(null);

        else

            dialog.setSelectedFile(new File(defaultFile));

        while (true) {

            int option = dialog.showSaveDialog(parent);

            if (option != JFileChooser.APPROVE\_OPTION)

                return null;

            File selectedFile = dialog.getSelectedFile();

            if (!selectedFile.exists())

                return selectedFile;

            else {

                int response = JOptionPane.showConfirmDialog(parent,

                        "The file \"" + selectedFile.getName()

                                + "\" already exists.\nDo you want to replace it?",

                        "Confirm Save",

                        JOptionPane.YES\_NO\_CANCEL\_OPTION,

                        JOptionPane.WARNING\_MESSAGE);

                if (response == JOptionPane.CANCEL\_OPTION)

                    return null;

                if (response == JOptionPane.YES\_OPTION)

                    return selectedFile;

            }

        }

    }

}

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.RenderingHints;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.Scanner;

import javax.imageio.ImageIO;

import javax.swing.BorderFactory;

import javax.swing.JColorChooser;

import javax.swing.JLabel;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JTextField;

import javax.swing.KeyStroke;

public class DrawTextPanel extends JPanel {

    private ArrayList<DrawTextItem> textItems;

    private Color currentTextColor = Color.BLACK;

    private Canvas canvas;

    private JTextField input;

    private SimpleFileChooser fileChooser;

    private JMenuBar menuBar;

    private MenuHandler menuHandler;

    private JMenuItem undoMenuItem;

    private class Canvas extends JPanel {

        Canvas() {

            setPreferredSize(new Dimension(800, 600));

            setBackground(Color.LIGHT\_GRAY);

            setFont(new Font("Serif", Font.BOLD, 24));

        }

        protected void paintComponent(Graphics g) {

            super.paintComponent(g);

            ((Graphics2D) g).setRenderingHint(RenderingHints.KEY\_ANTIALIASING,

                    RenderingHints.VALUE\_ANTIALIAS\_ON);

            for (DrawTextItem textItem : textItems) {

                textItem.draw(g);

            }

        }

    }

    private class MenuHandler implements ActionListener {

        public void actionPerformed(ActionEvent evt) {

            doMenuCommand(evt.getActionCommand());

        }

    }

    public DrawTextPanel() {

        fileChooser = new SimpleFileChooser();

        undoMenuItem = new JMenuItem("Remove Item");

        undoMenuItem.setEnabled(false);

        menuHandler = new MenuHandler();

        textItems = new ArrayList<>();

        setLayout(new BorderLayout(3, 3));

        setBackground(Color.BLACK);

        setBorder(BorderFactory.createLineBorder(Color.BLACK, 2));

        canvas = new Canvas();

        add(canvas, BorderLayout.CENTER);

        JPanel bottom = new JPanel();

        bottom.add(new JLabel("Text to add: "));

        input = new JTextField("Hello World!", 40);

        bottom.add(input);

        add(bottom, BorderLayout.SOUTH);

        canvas.addMouseListener(new MouseAdapter() {

            public void mousePressed(MouseEvent e) {

                doMousePress(e);

            }

        });

    }

    public void doMousePress(MouseEvent e) {

        String text = input.getText().trim();

        if (text.length() == 0) {

            input.setText("Hello World!");

            text = "Hello World!";

        }

        DrawTextItem s = new DrawTextItem(text, e.getX(), e.getY());

        s.setTextColor(currentTextColor);

        textItems.add(s);

        // SOME OTHER OPTIONS THAT CAN BE APPLIED TO TEXT ITEMS:

        // s.setFont( new Font( "Serif", Font.ITALIC + Font.BOLD, 12 )); // Default is

        // null, meaning font of canvas.

        // s.setMagnification(3); // Default is 1, meaning no magnification.

        // s.setBorder(true); // Default is false, meaning don't draw a border.

        // s.setRotationAngle(25); // Default is 0, meaning no rotation.

        // s.setTextTransparency(0.3); // Default is 0, meaning text is not at all

        // transparent.

        // s.setBackground(Color.BLUE); // Default is null, meaning don't draw a

        // background area.

        // s.setBackgroundTransparency(0.7); // Default is 0, meaning background is not

        // transparent.

        undoMenuItem.setEnabled(true);

        canvas.repaint();

    }

    public JMenuBar getMenuBar() {

        if (menuBar == null) {

            menuBar = new JMenuBar();

            String commandKey;

            if (System.getProperty("mrj.version") == null)

                commandKey = "control ";

            else

                commandKey = "meta ";

            JMenu fileMenu = new JMenu("File");

            menuBar.add(fileMenu);

            JMenuItem saveItem = new JMenuItem("Save...");

            saveItem.setAccelerator(KeyStroke.getKeyStroke(commandKey + "N"));

            saveItem.addActionListener(menuHandler);

            fileMenu.add(saveItem);

            JMenuItem openItem = new JMenuItem("Open...");

            openItem.setAccelerator(KeyStroke.getKeyStroke(commandKey + "O"));

            openItem.addActionListener(menuHandler);

            fileMenu.add(openItem);

            fileMenu.addSeparator();

            JMenuItem saveImageItem = new JMenuItem("Save Image...");

            saveImageItem.addActionListener(menuHandler);

            fileMenu.add(saveImageItem);

            JMenu editMenu = new JMenu("Edit");

            menuBar.add(editMenu);

            undoMenuItem.addActionListener(menuHandler);

            undoMenuItem.setAccelerator(KeyStroke.getKeyStroke(commandKey + "Z"));

            editMenu.add(undoMenuItem);

            editMenu.addSeparator();

            JMenuItem clearItem = new JMenuItem("Clear");

            clearItem.addActionListener(menuHandler);

            editMenu.add(clearItem);

            JMenu optionsMenu = new JMenu("Options");

            menuBar.add(optionsMenu);

            JMenuItem colorItem = new JMenuItem("Set Text Color...");

            colorItem.setAccelerator(KeyStroke.getKeyStroke(commandKey + "T"));

            colorItem.addActionListener(menuHandler);

            optionsMenu.add(colorItem);

            JMenuItem bgColorItem = new JMenuItem("Set Background Color...");

            bgColorItem.addActionListener(menuHandler);

            optionsMenu.add(bgColorItem);

            JMenuItem fontItem = new JMenuItem("Set Font...");

            fontItem.addActionListener(menuHandler);

            optionsMenu.add(fontItem);

        }

        return menuBar;

    }

    private void doMenuCommand(String command) {

        if (command.equals("Save...")) {

            File saveFile = fileChooser.getOutputFile(this, "Select Save File", "drawing.txt");

            if (saveFile != null) {

                try (PrintWriter writer = new PrintWriter(saveFile)) {

                    Color bgColor = canvas.getBackground();

                    writer.println(bgColor.getRed() + " " + bgColor.getGreen() + " " + bgColor.getBlue());

                    for (DrawTextItem item : textItems) {

                        writer.println(item.getString());

                        writer.println(item.getX() + " " + item.getY());

                        Color textColor = item.getTextColor();

                        writer.println(textColor.getRed() + " " + textColor.getGreen() + " " + textColor.getBlue());

                    }

                } catch (Exception e) {

                    JOptionPane.showMessageDialog(this, "Error saving file:\n" + e.getMessage());

                }

            }

        } else if (command.equals("Open...")) {

            File openFile = fileChooser.getInputFile(this, "Open Drawing");

            if (openFile != null) {

                try (Scanner scanner = new Scanner(openFile)) {

                    // Read background color

                    int bgRed = Integer.parseInt(scanner.nextLine());

                    int bgGreen = Integer.parseInt(scanner.nextLine());

                    int bgBlue = Integer.parseInt(scanner.nextLine());

                    Color bgColor = new Color(bgRed, bgGreen, bgBlue);

                    canvas.setBackground(bgColor);

                    // Read DrawTextItems

                    textItems.clear(); // Clear existing items

                    while (scanner.hasNextLine()) {

                        String text = scanner.nextLine();

                        // Read x and y values

                        String[] xyValues = scanner.nextLine().split(" ");

                        int x = Integer.parseInt(xyValues[0]);

                        int y = Integer.parseInt(xyValues[1]);

                        // Read textColor values

                        String[] textColorValues = scanner.nextLine().split(" ");

                        int textRed = Integer.parseInt(textColorValues[0]);

                        int textGreen = Integer.parseInt(textColorValues[1]);

                        int textBlue = Integer.parseInt(textColorValues[2]);

                        Color textColor = new Color(textRed, textGreen, textBlue);

                        DrawTextItem newItem = new DrawTextItem(text, x, y);

                        newItem.setTextColor(textColor);

                        textItems.add(newItem);

                    }

                } catch (Exception e) {

                    JOptionPane.showMessageDialog(this,

                            "Error opening file: " + e.getMessage(),

                            "Error", JOptionPane.ERROR\_MESSAGE);

                }

                canvas.repaint();

            }

        } else if (command.equals("Clear")) { // remove all strings

            textItems.clear();

            undoMenuItem.setEnabled(false);

            canvas.repaint();

        } else if (command.equals("Remove Item")) {

            textItems.remove(textItems.size() - 1);

            undoMenuItem.setEnabled(!textItems.isEmpty());

            canvas.repaint();

        } else if (command.equals("Set Text Color...")) {

            Color c = JColorChooser.showDialog(this, "Select Text Color", currentTextColor);

            if (c != null)

                currentTextColor = c;

        } else if (command.equals("Set Background Color...")) {

            Color c = JColorChooser.showDialog(this, "Select Background Color", canvas.getBackground());

            if (c != null) {

canvas.setBackground(c);

                canvas.repaint();

            }

        } else if (command.equals("Save Image...")) { // save a PNG image of the drawing area

            File imageFile = fileChooser.getOutputFile(this, "Select Image File Name", "textimage.png");

            if (imageFile == null)

                try {

                    BufferedImage image = new BufferedImage(canvas.getWidth(), canvas.getHeight(),

                            BufferedImage.TYPE\_INT\_RGB);

                    Graphics g = image.getGraphics();

                    g.setFont(canvas.getFont());

                    canvas.paintComponent(g); // draws the canvas onto the BufferedImage, not the screen!

                    boolean ok = ImageIO.write(image, "PNG", imageFile); // write to the file

                    if (ok == false)

                        throw new Exception("PNG format not supported (this shouldn't happen!).");

                } catch (Exception e) {

                    JOptionPane.showMessageDialog(this,

                            "Sorry, an error occurred while trying to save the image:\n" + e);

                }

        } else if (command.equals("Set Font...")) {

            setFont();

        }

    }

    private void setFont() {

        Font selectedFont = JFontChooser.showDialog(DrawTextPanel.this, "Select Font");

        if (selectedFont != null) {

            for (DrawTextItem textItem : textItems) {

                textItem.setFont(selectedFont);

            }

            canvas.repaint();

        }

    }

}

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Component;

import java.awt.Container;

import java.awt.Dimension;

import java.awt.FlowLayout;

import java.awt.Font;

import java.awt.GraphicsEnvironment;

import java.awt.Window;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.ComponentAdapter;

import java.awt.event.ComponentEvent;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import javax.swing.Box;

import javax.swing.JButton;

import javax.swing.JComponent;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JList;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextField;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

public class JFontChooser extends JComponent {

  public static void main(String[] args) {

    Font font = JFontChooser.showDialog((Component) null, "Font");

    System.out.println("Font: " + font);

  }

  public static Font showDialog(Component parent, String title) {

    final JFontChooser pane = new JFontChooser();

    FontTracker ok = new FontTracker(pane);

    JDialog dialog = createDialog(parent, title, true, pane, ok, null);

    dialog.addWindowListener(new JFontChooserDialog.Closer());

    dialog.addComponentListener(new JFontChooserDialog.DisposeOnClose());

    dialog.setVisible(true);

    return ok.getSelectedFont();

  }

  public static JDialog createDialog(

      Component parent, String title,

      boolean modal,

      JFontChooser chooserPane,

      ActionListener okListener,

      ActionListener cancelListener) {

    return new JFontChooserDialog(parent, title, modal, chooserPane, okListener, cancelListener);

  }

  public JFontChooser() {

    setLayout(new BorderLayout());

    PreviewPanel previewPane = new PreviewPanel();

    m\_inputPane = new InputPanel(previewPane);

    add(m\_inputPane, BorderLayout.CENTER);

    add(previewPane, BorderLayout.SOUTH);

  }

  public Font getSelectedFont() {

    return m\_inputPane.getSelectedFont();

  }

  private InputPanel m\_inputPane;

  class InputPanel extends JPanel {

    public InputPanel(ListSelectionListener listener) {

      setLayout(new BorderLayout());

      Box nameBox = Box.createVerticalBox();

      nameBox.add(Box.createVerticalStrut(10));

      JLabel fontNameLabel = new JLabel("Font Name:");

      nameBox.add(fontNameLabel);

      if (listener != null) {

        m\_fontNameList.addListSelectionListener(listener);

      }

      JScrollPane namePane = new JScrollPane(m\_fontNameList);

      nameBox.add(namePane);

      nameBox.add(Box.createVerticalStrut(10));

      Box styleBox = Box.createVerticalBox();

      styleBox.add(Box.createVerticalStrut(10));

      JLabel fontStyleLabel = new JLabel("Font Style:");

      styleBox.add(fontStyleLabel);

      if (listener != null) {

        m\_fontStyleList.addListSelectionListener(listener);

      }

      JScrollPane stylePane = new JScrollPane(m\_fontStyleList);

      styleBox.add(stylePane);

      styleBox.add(Box.createVerticalStrut(10));

      Box sizeBox = Box.createVerticalBox();

      sizeBox.add(Box.createVerticalStrut(10));

      JLabel fontSizeLabel = new JLabel("Size:");

      sizeBox.add(fontSizeLabel);

      if (listener != null) {

        m\_fontSizeList.addListSelectionListener(listener);

      }

      JScrollPane sizePane = new JScrollPane(m\_fontSizeList);

      sizeBox.add(sizePane);

      sizeBox.add(Box.createVerticalStrut(10));

      Box mainBox = Box.createHorizontalBox();

      mainBox.add(Box.createHorizontalStrut(10));

      mainBox.add(nameBox);

      mainBox.add(Box.createHorizontalStrut(10));

      mainBox.add(styleBox);

      mainBox.add(Box.createHorizontalStrut(10));

      mainBox.add(sizeBox);

      mainBox.add(Box.createHorizontalStrut(10));

      add(mainBox, BorderLayout.CENTER);

    }

    public Font getSelectedFont() {

      return new Font(m\_fontNameList.getFontName(), m\_fontStyleList.getFontStyle(), m\_fontSizeList.getFontSize());

    }

    private FontNameList m\_fontNameList = new FontNameList();

    private FontStyleList m\_fontStyleList = new FontStyleList();

    private FontSizeList m\_fontSizeList = new FontSizeList();

  }

  class PreviewPanel extends JPanel

      implements ListSelectionListener {

    public PreviewPanel() {

      setLayout(new FlowLayout());

      Box box = Box.createVerticalBox();

      JLabel previewLabel = new JLabel("Preview:");

      box.add(previewLabel);

      m\_text.setEditable(false);

      m\_text.setBackground(Color.white);

      m\_text.setForeground(Color.black);

      JScrollPane pane = new JScrollPane(m\_text);

      pane.setPreferredSize(new Dimension(300, 80));

      box.add(pane);

      add(box);

    }

    public void valueChanged(ListSelectionEvent ev) {

      m\_text.setFont(JFontChooser.this.getSelectedFont());

    }

    private JTextField m\_text = new JTextField("The quick brown fox jumps over the lazy dog");

  }

}

class FontNameList extends JList {

  FontNameList() {

    super(m\_fontNames);

    setSelectedIndex(0);

    setVisibleRowCount(5);

  }

  String getFontName() {

    String name = (String) getSelectedValue();

    return name;

  }

  private static final String[] m\_fontNames = GraphicsEnvironment.getLocalGraphicsEnvironment()

      .getAvailableFontFamilyNames();

}

class FontStyleList extends JList {

  FontStyleList() {

    super(m\_fontStyles);

    setSelectedIndex(0);

    setVisibleRowCount(5);

  }

  int getFontStyle() {

    int style = 0;

    String name = (String) getSelectedValue();

    if (name.equals("Regular")) {

      style = Font.PLAIN;

    } else if (name.equals("Italic")) {

      style = Font.ITALIC;

    } else if (name.equals("Bold")) {

      style = Font.BOLD;

    } else {

      style = Font.BOLD + Font.ITALIC;

    }

    return style;

  }

  private static final String[] m\_fontStyles = { "Regular", "Italic", "Bold", "Bold Italic" };

}

class FontSizeList extends JList {

  FontSizeList() {

    super(m\_fontSizes);

    setSelectedIndex(4);

    setVisibleRowCount(5);

  }

  int getFontSize() {

    int size = Integer.parseInt((String) getSelectedValue());

    return size;

  }

  private static final String[] m\_fontSizes = {

      "6", "8", "10", "12", "14", "16", "18",

      "20", "22", "24", "36", "72"

  };

}

class JFontChooserDialog extends JDialog {

  public JFontChooserDialog(

      Component component,

      String title,

      boolean modal,

      JFontChooser chooserPane,

      ActionListener okListener,

      ActionListener cancelListener) {

    super(JOptionPane.getFrameForComponent(component), title, modal);

    m\_chooserPane = chooserPane;

    Container contentPane = getContentPane();

    contentPane.setLayout(new BorderLayout());

    contentPane.add(m\_chooserPane, BorderLayout.CENTER);

    JPanel buttonPane = new JPanel();

    buttonPane.setLayout(new FlowLayout(FlowLayout.CENTER));

    JButton okButton = new JButton("OK");

    getRootPane().setDefaultButton(okButton);

    okButton.setActionCommand("OK");

    if (okListener != null) {

      okButton.addActionListener(okListener);

    }

    okButton.addActionListener(new ActionListener() {

      public void actionPerformed(ActionEvent e) {

        setVisible(false);

      }

    });

    buttonPane.add(okButton);

    JButton cancelButton = new JButton("Cancel");

    cancelButton.setActionCommand("cancel");

    if (cancelListener != null) {

      cancelButton.addActionListener(cancelListener);

    }

    cancelButton.addActionListener(new ActionListener() {

      public void actionPerformed(ActionEvent e) {

        setVisible(false);

      }

    });

    buttonPane.add(cancelButton);

    contentPane.add(buttonPane, BorderLayout.SOUTH);

    pack();

    setLocationRelativeTo(component);

  }

  static class Closer extends WindowAdapter {

    public void windowClosing(WindowEvent e) {

      Window w = e.getWindow();

      w.setVisible(false);

    }

  }

  static class DisposeOnClose extends ComponentAdapter {

    public void componentHidden(ComponentEvent e) {

      Window w = (Window) e.getComponent();

      w.dispose();

    }

  }

  private JFontChooser m\_chooserPane;

}

class FontTracker implements ActionListener {

  public FontTracker(JFontChooser chooser) {

    m\_chooser = chooser;

  }

  public void actionPerformed(ActionEvent e) {

    m\_font = m\_chooser.getSelectedFont();

  }

  public Font getSelectedFont() {

    return m\_font;

  }

  private JFontChooser m\_chooser;

  private Font m\_font;

}

import java.io.File;

import java.util.Scanner;

public class DirectoryList {

   public static void main(String[] args) {

      String directoryName;

      File directory;

      String[] files;

      Scanner scanner;

      scanner = new Scanner(System.in);

      System.out.print("Enter a directory name: ");

      directoryName = scanner.nextLine().trim();

      directory = new File(directoryName);

      if (directory.isDirectory() == false) {

         if (directory.exists() == false)

            System.out.println("There is no such directory!");

         else

            System.out.println("That file is not a directory.");

      } else {

         files = directory.list();

         System.out.println("Files in directory \"" + directory + "\":");

         for (int i = 0; i < files.length; i++)

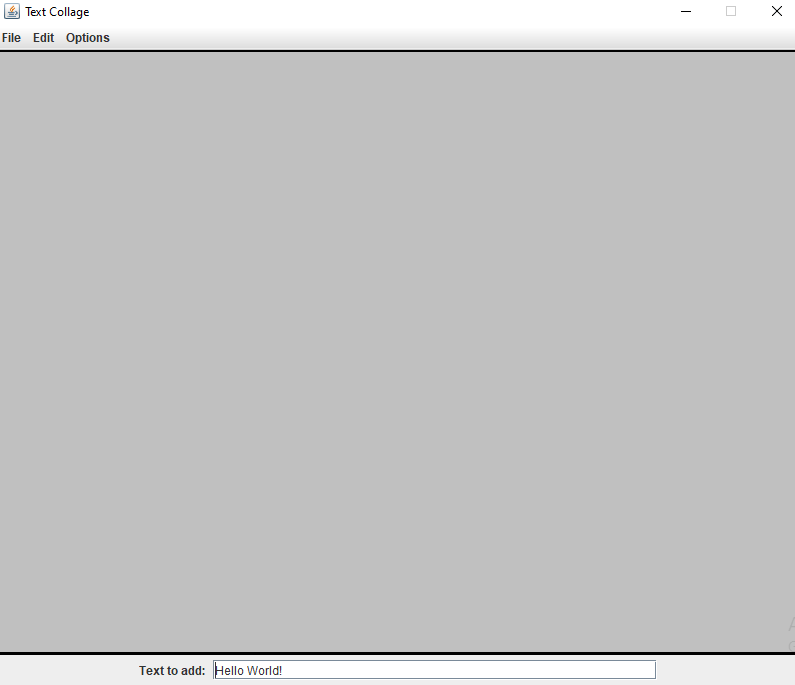
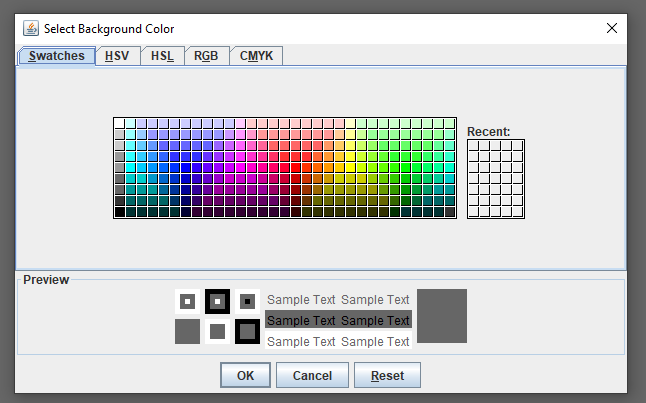
            System.out.println("   " + files[i]);

      }

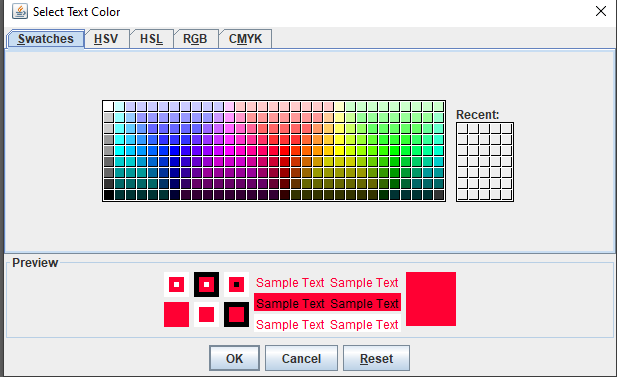
      scanner.close();

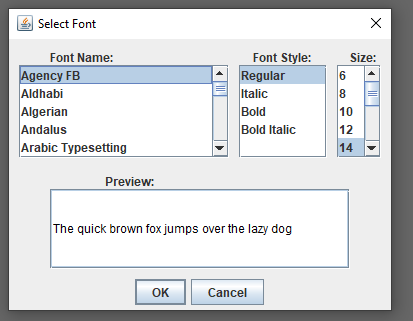
   }

}

**7. Output**  








**The End**