Notepad Application using JAVA Swings

To create a Notepad like app using the Swing and File handling in Java

From the JFrame

**Setting up the Menu Navigation bar**

Step 1: Create a Menubar using the JMenuBar()

Step 2: Create Menu using the JMenu() (eg: File, Edit, Help)

Step 3: Then add these menu to the menubar created before by using the add() function

Step 4: Now setJMenuBar() to fit the menus properly in the menubar

**Setting up the Menu item to the created menus by following the above steps done for menubar preparation**

Step 5: Create the JMenuItem eg(new, open, save, select all) similar functionalities and add it to the Menu created by using the add function

From the awt.;\* ;

**Setting up the TextField Area**

Step 6: JTextField and JScroLLPane() [for creating the container to the Text field]

Formatiing 🡪fonts,line wrapping words,border layout

From the awt.event.\*;

**Adding the actions or events to the menu items**

by using the filechooser and similar awt objects

**CODE:**

**Notepad.java**

/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

package notepad;

import java.awt.*\**;

import java.awt.BorderLayout;

import java.awt.event.*\**;

import java.awt.event.ActionEvent;

import java.awt.event.KeyEvent;

import java.awt.print.PrinterException;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import javax.swing.*\**;

import javax.swing.event.UndoableEditEvent;

import javax.swing.event.UndoableEditListener;

import javax.swing.undo.CannotRedoException;

import javax.swing.undo.CannotUndoException;

import javax.swing.undo.UndoManager;

/\*\*

 \*

 \* @author JSVS^

 \*/

public class Notepad extends *JFrame* implements *ActionListener* {

*JFrame* window;

*JTextArea* area;

*JScrollPane* pane;

*String* fileName;

*String* fileAddress;

*String* text;

*UndoManager* um = **new** UndoManager();

  Notepad() {

    setTitle("Notely");

    // setSize(1100,700);

    setBounds(200, 50, 1100, 700);

    setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

*JMenuBar* menubar = **new** JMenuBar();

*JMenu* file = **new** JMenu("File");

    //to add the menu items to the file menu

*JMenuItem* newdoc = **new** JMenuItem("New");

    newdoc.setAccelerator(

      KeyStroke.getKeyStroke(KeyEvent.VK\_N, ActionEvent.CTRL\_MASK)

    ); //to set he key shortcut

    newdoc.addActionListener(*this*);

*JMenuItem* open = **new** JMenuItem("Open");

    open.setAccelerator(

      KeyStroke.getKeyStroke(KeyEvent.VK\_O, ActionEvent.CTRL\_MASK)

    );

    open.addActionListener(*this*);

*JMenuItem* save = **new** JMenuItem("Save");

    save.setAccelerator(

      KeyStroke.getKeyStroke(KeyEvent.VK\_S, ActionEvent.CTRL\_MASK)

    );

    save.addActionListener(*this*);

*JMenuItem* saveas = **new** JMenuItem("Save As");

    saveas.setAccelerator(

      KeyStroke.getKeyStroke(

        KeyEvent.VK\_S,

        ActionEvent.CTRL\_MASK | ActionEvent.SHIFT\_MASK

      )

    );

    saveas.addActionListener(*this*);

*JMenuItem* print = **new** JMenuItem("Print");

    print.setAccelerator(

      KeyStroke.getKeyStroke(KeyEvent.VK\_P, ActionEvent.CTRL\_MASK)

    );

    print.addActionListener(*this*);

*JMenuItem* exit = **new** JMenuItem("Exit");

    exit.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK\_ESCAPE, 0));

    exit.addActionListener(*this*);

    //add these to the menu named "File"

    file.add(newdoc);

    file.add(open);

    file.add(save);

    file.add(saveas);

    file.add(print);

    file.add(exit);

*JMenu* edit = **new** JMenu("Edit");

*JMenuItem* undo = **new** JMenuItem("Undo");

    undo.setAccelerator(

      KeyStroke.getKeyStroke(KeyEvent.VK\_Z, ActionEvent.CTRL\_MASK)

    );

    undo.addActionListener(*this*);

*JMenuItem* redo = **new** JMenuItem("Redo");

    redo.setAccelerator(

      KeyStroke.getKeyStroke(KeyEvent.VK\_Y, ActionEvent.CTRL\_MASK)

    );

    redo.addActionListener(*this*);

*JMenuItem* copy = **new** JMenuItem("Copy");

    copy.setAccelerator(

      KeyStroke.getKeyStroke(KeyEvent.VK\_C, ActionEvent.CTRL\_MASK)

    );

    copy.addActionListener(*this*);

*JMenuItem* paste = **new** JMenuItem("Paste");

    paste.setAccelerator(

      KeyStroke.getKeyStroke(KeyEvent.VK\_V, ActionEvent.CTRL\_MASK)

    );

    paste.addActionListener(*this*);

*JMenuItem* cut = **new** JMenuItem("Cut");

    cut.setAccelerator(

      KeyStroke.getKeyStroke(KeyEvent.VK\_X, ActionEvent.CTRL\_MASK)

    );

    cut.addActionListener(*this*);

*JMenuItem* selectall = **new** JMenuItem("Select All");

    selectall.setAccelerator(

      KeyStroke.getKeyStroke(KeyEvent.VK\_A, ActionEvent.CTRL\_MASK)

    );

    selectall.addActionListener(*this*);

    edit.add(undo);

    edit.add(redo);

    edit.add(copy);

    edit.add(paste);

    edit.add(cut);

    edit.add(selectall);

*JMenu* help = **new** JMenu("Help");

*JMenuItem* about = **new** JMenuItem("About Notely");

    about.addActionListener(*this*);

    help.add(about);

    //adding menu to the menubar

    menubar.add(file);

    menubar.add(edit);

    menubar.add(help);

    setJMenuBar(menubar);

    area = **new** JTextArea();

    //to set the font family and its size

    area.setFont(**new** Font("Open Sans", Font.PLAIN, 20));

    //for the line wrapping to avoid the single line infinity writings

    area.setLineWrap(true);

    area.setWrapStyleWord(true);

    area

      .getDocument()

      .addUndoableEditListener(

**new** UndoableEditListener() {

          public *void* undoableEditHappened(*UndoableEditEvent* e) {

            um.addEdit(e.getEdit());

          }

        }

      );

    pane = **new** JScrollPane(area);

    //to remove the border around the text field area

    pane.setBorder(BorderFactory.createEmptyBorder());

    add(pane, BorderLayout.CENTER);

  }

  @*Override*

  public *void* actionPerformed(*ActionEvent* e) {

    switch (e.getActionCommand()) {

      case "New":

        area.setText("");

        setTitle("New");

        fileName = null;

        fileAddress = null;

        break;

      case "Open":

*FileDialog* fd = **new** FileDialog(window, "Open", FileDialog.LOAD);

        fd.setVisible(true);

        if (fd.getFile() != null) {

          fileName = fd.getFile();

          fileAddress = fd.getDirectory();

          setTitle(fileName);

        }

        try {

*BufferedReader* br = **new** BufferedReader(

**new** FileReader(fileAddress + fileName)

          );

          area.setText("");

*String* line = null;

          while ((line = br.readLine()) != null) {

            area.append(line + "\n");

          }

          br.close();

        } catch (*IOException* ae) {

          System.out.println("FILE NOT OPENED!");

        }

        break;

      case "Save":

        if (fileName == null) {

*FileDialog* fd1 = **new** FileDialog(window, "Save", FileDialog.SAVE);

          fd1.setVisible(true);

          if (fd1.getFile() != null) {

            fileName = fd1.getFile();

            fileAddress = fd1.getDirectory();

            setTitle(fileName);

          }

          try {

*FileWriter* fw1 = **new** FileWriter(fileAddress + fileName);

            fw1.write(area.getText());

            setTitle(fileName);

            fw1.close();

          } catch (*IOException* ae) {

            System.out.println("SOMETHING WRONG!!");

          }

        } else {

          try {

*FileWriter* fw1 = **new** FileWriter(fileAddress + fileName);

            fw1.write(area.getText());

            fw1.close();

          } catch (*IOException* ae) {

            System.out.println("SOMETHING WRONG!!");

          }

        }

        break;

      case "Save As":

*FileDialog* fd2 = **new** FileDialog(window, "Save", FileDialog.SAVE);

        fd2.setVisible(true);

        if (fd2.getFile() != null) {

          fileName = fd2.getFile();

          fileAddress = fd2.getDirectory();

          setTitle(fileName);

        }

        try {

*FileWriter* fw2 = **new** FileWriter(fileAddress + fileName);

          fw2.write(area.getText());

          fw2.close();

        } catch (*IOException* ae) {

          System.out.println("SOMETHING WRONG!!");

        }

        break;

      case "Print":

        try {

          area.print();

        } catch (*PrinterException* ae) {

          System.out.println("SOMETHING WRONG!!");

        }

        break;

      case "Exit":

        System.exit(0);

        break;

      //Edit Menu

      case "Undo":

        try {

          um.undo();

        } catch (*CannotUndoException* ae) {

          System.out.println("SOMETHING WRONG!!");

        }

        break;

      case "Redo":

        try {

          um.redo();

        } catch (*CannotRedoException* ae) {

          System.out.println("SOMETHING WRONG!!");

        }

        break;

      case "Copy":

        text = area.getSelectedText();

        break;

      case "Paste":

        area.insert(text, area.getCaretPosition());

        break;

      case "Cut":

        area.replaceRange("", area.getSelectionStart(), area.getSelectionEnd());

        break;

      case "Select All":

        area.selectAll();

        break;

      case "About Notely":

**new** About().setVisible(true);

        break;

    }

  }

  public static *void* main(*String*[] args) {

**new** Notepad().setVisible(true);

  }

  // TODO code application logic here

}

**About.java**

/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

package notepad;

import java.awt.*\**;

import java.awt.event.*\**;

import javax.swing.*\**;

/\*\*

 \*

 \* @author JSVS

 \*/

public class About extends *JFrame* implements *ActionListener* {

*JButton* btn;

  About() {

    setTitle("About Notely");

    setBounds(500, 200, 690, 550);

    setLayout(null);

*ImageIcon* img1 = **new** ImageIcon(

      ClassLoader.getSystemResource("notepad/icons/windows11.png")

    );

*Image* img2 = img1

      .getImage()

      .getScaledInstance(380, 70, Image.SCALE\_DEFAULT);

*ImageIcon* img3 = **new** ImageIcon(img2); //storing the scaled image to img3

*JLabel* l1 = **new** JLabel(img3);

    l1.setBounds(161, 40, 380, 70);

    add(l1);

*ImageIcon* img4 = **new** ImageIcon(

      ClassLoader.getSystemResource("notepad/icons/notely.png")

    );

*Image* img5 = img4

      .getImage()

      .getScaledInstance(120, 100, Image.SCALE\_DEFAULT);

*ImageIcon* img6 = **new** ImageIcon(img5); //storing the scaled image to img3

*JLabel* l2 = **new** JLabel(img6);

    l2.setBounds(40, 150, 110, 90);

    add(l2);

*JLabel* l3 = **new** JLabel(

      "<html>Notely ft. Brainiac Space<br> Version Stable 1.00.00 <br> Vidhya Varshany, All Rights Reserved<br><br><b>Notely</b> is a Simple Minimal Text Editor Program with smooth user-experience and interface which allows to create files in any format instantly<br><br>🌟Why Notely??<br>1. Easy to use<br>2. Portable to Create Documents<br>3. Compact in size<br><br>Made with 💖Java Swings</html>"

    );

    l3.setBounds(180, 140, 520, 330);

    l3.setFont(**new** Font("Monospace", Font.PLAIN, 17));

    add(l3);

    btn = **new** JButton("OK");

    btn.setBounds(520, 460, 80, 20);

    btn.addActionListener(*this*);

    add(btn);

  }

  public static *void* main(*String*[] args) {

**new** About().setVisible(true);

  }

  @*Override*

  public *void* actionPerformed(*ActionEvent* e) {

*this*.setVisible(false);

  }

}

**OUTPUT:**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA picture containing text, screenshot, monitor, indoor

Description automatically generated

**A screenshot of a computer

Description automatically generated with low confidence**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

**Converting to jar**

**Text

Description automatically generated**

**Text

Description automatically generated**

Text

Description automatically generatedGraphical user interface, text, chat or text message

Description automatically generatedText

Description automatically generated

Graphical user interface, application

Description automatically generated