PREPARED BY: RAMAN SHUKLA

Java PROJECT 2024

+91 630 774 1294
RAMANSHUKLA2005@GMAIL.COM

Submitted to:: Punit Sir

TABLE OF CONTENTS

Introduction to Project

The Notes Manager is a simple yet effective Java-based application designed using JFrame and Swing to streamline the process of taking and managing notes. This application provides an easy-to-use platform for individuals who need a straightforward tool to store and retrieve their notes efficiently. With a focus on simplicity and usability, the Notes Manager offers a seamless experience for users, making it an ideal tool for everyday note-taking needs.

Key Features

User-Friendly Interface: An intuitive and easy-to-navigate interface built with JFrame and Swing, allowing users to create, edit, and delete notes with minimal effort.

Organizational Tools: Options to categorize and tag notes, ensuring easy retrieval and organization.

Data Persistence: Notes are stored persistently, ensuring that users' data is saved and accessible even after the application is closed.This project leverages Java's robust programming capabilities alongside the Swing framework to deliver a reliable and efficient notes management solution. Whether for personal use or simple task management, the Notes Manager is designed to help users stay organized and productive.

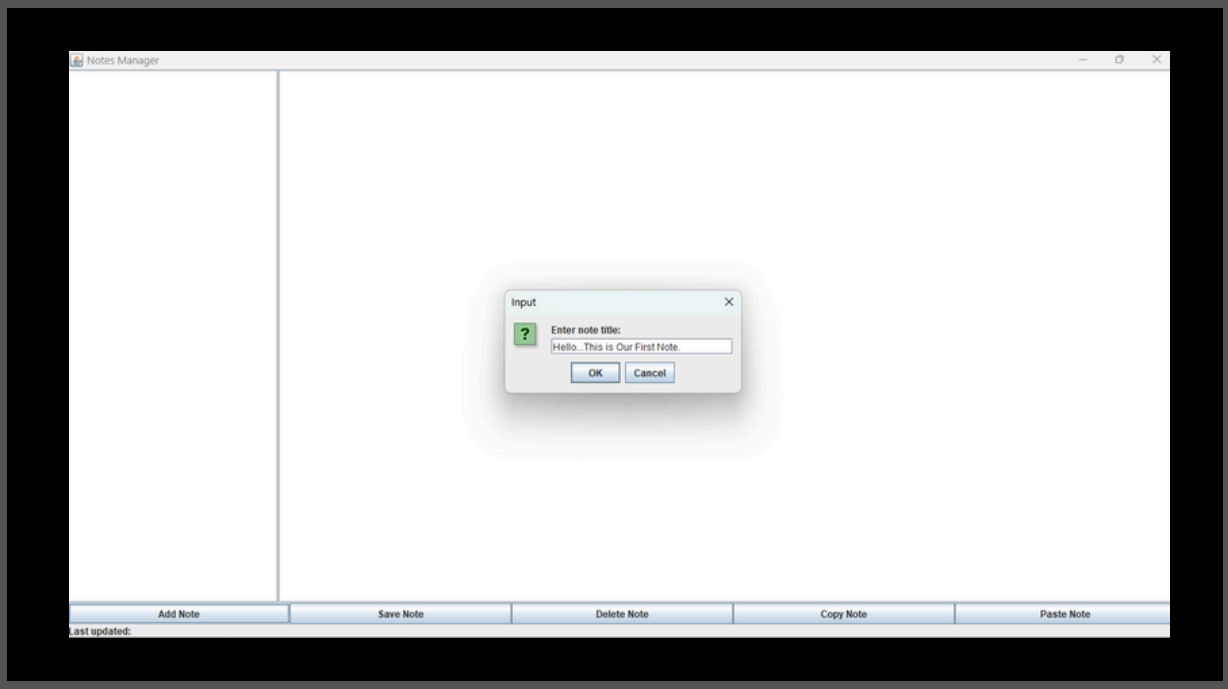
Here is the JAVA Code for Notes Manager

```
import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.GridLayout;
import java.awt.Toolkit;
import java.awt.datatransfer.Clipboard;
import java.awt.datatransfer.ClipboardOwner;
import java.awt.datatransfer.DataFlavor;
import java.awt.datatransfer.StringSelection;
import java.util.ArrayList;
import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.SwingUtilities;

public class NotesManager extends JFrame {
  private ArrayList<NotesManager$Note> notes = new ArrayList();
  private DefaultListModel<String> listModel = new DefaultListModel();
  private JList<String> notesList;
  private JTextArea noteArea;
  private JLabel lastUpdateLabel;
  private JButton addButton;
  private JButton saveButton;
  private JButton deleteButton;
  private JButton copyButton;
  private JButton pasteButton;

  public NotesManager() {
   this.setTitle("Notes Manager");
   this.setSize(600, 400);
   this.setDefaultCloseOperation(3);
   this.setLocationRelativeTo((Component)null);
   this.setLayout(new BorderLayout());
   this.notesList = new JList(this.listModel);
   this.notesList.setSelectionMode(0);
   this.notesList.addListSelectionListener((var1x) -> {
    this.showNote();
   });
   this.noteArea = new JTextArea();
   this.noteArea.setLineWrap(true);
   this.noteArea.setWrapStyleWord(true);
   this.lastUpdateLabel = new JLabel("Last updated: ");
   this.addButton = new JButton("Add Note");
   this.saveButton = new JButton("Save Note");
```

```
this.deleteButton = new JButton("Delete Note");
 this.copyButton = new JButton("Copy Note");
 this.pasteButton = new JButton("Paste Note");
 this.addButton.addActionListener((var1x) -> {
  this.addNote();
 });
 this.saveButton.addActionListener((var1x) -> {
  this.saveNote();
 });
 this.deleteButton.addActionListener((var1x) -> {
  this.deleteNote();
 });
 this.copyButton.addActionListener((var1x) -> {
  this.copyNote();
 });
 this.pasteButton.addActionListener((var1x) -> {
  this.pasteNote();
 });
 JPanel var1 = new JPanel();
 var1.setLayout(new GridLayout(1, 5));
 var1.add(this.addButton);
 var1.add(this.saveButton);
 var1.add(this.deleteButton);
 var1.add(this.copyButton);
 var1.add(this.pasteButton);
 JPanel var2 = new JPanel();
 var2.setLayout(new BorderLayout());
 var2.add(var1, "North");
 var2.add(this.lastUpdateLabel, "South");
 this.add(new JScrollPane(this.notesList), "West");
 this.add(new JScrollPane(this.noteArea), "Center");
 this.add(var2, "South");
}

private void addNote() {
 String var1 = JOptionPane.showInputDialog("Enter note title:");
 if (var1 != null && !var1.trim().isEmpty()) {
  this.notes.add(new NotesManager$Note(this, var1, ""));
  this.listModel.addElement(var1);
 }

}

private void showNote() {
 int var1 = this.notesList.getSelectedIndex();
 if (var1 != -1) {
  NotesManager$Note var2 = (NotesManager$Note)this.notes.get(var1);
  this.noteArea.setText(var2.getContent());
  this.lastUpdateLabel.setText("Last updated: " + var2.getLastUpdated());
 }

}
```

```
  private void saveNote() {
   int var1 = this.notesList.getSelectedIndex();
   if (var1 != -1) {
    NotesManager$Note var2 = (NotesManager$Note)this.notes.get(var1);
    var2.setContent(this.noteArea.getText());
    var2.updateTimestamp();
    this.lastUpdateLabel.setText("Last updated: " + var2.getLastUpdated());
   }

  }

  private void deleteNote() {
   int var1 = this.notesList.getSelectedIndex();
   if (var1 != -1) {
    this.notes.remove(var1);
    this.listModel.remove(var1);
    this.noteArea.setText("");
    this.lastUpdateLabel.setText("Last updated: ");
   }

  }

  private void copyNote() {
   int var1 = this.notesList.getSelectedIndex();
   if (var1 != -1) {
    String var2 = ((NotesManager$Note)this.notes.get(var1)).getContent();
    StringSelection var3 = new StringSelection(var2);
    Clipboard var4 = Toolkit.getDefaultToolkit().getSystemClipboard();
    var4.setContents(var3, (ClipboardOwner)null);
    JOptionPane.showMessageDialog(this, "Note copied to clipboard!");
   }

  }

  private void pasteNote() {
   try {
    Clipboard var1 = Toolkit.getDefaultToolkit().getSystemClipboard();
    String var2 = (String)var1.getData(DataFlavor.stringFlavor);
    this.noteArea.append(var2);
   } catch (Exception var3) {
    JOptionPane.showMessageDialog(this, "Failed to paste text: " + var3.getMessage());
   }

  }

  public static void main(String[] var0) {
   SwingUtilities.invokeLater(() -> {
    (new NotesManager()).setVisible(true);
   });
  }
}
```

Output of the code



What We have learnt??

Creating a Notes Manager app using Java with the Swing library involves several key concepts and components. Here are the main points covered in the process:

1. Swing Framework:
  -JFrame: The main window for the application.
  -JPanel, JButton, JTextArea, JTextField, JLabel: Basic components used for building the user interface.

2. Event Handling:
  - ActionListener: An interface for handling action events like button clicks.
  - Action Events: Methods to respond to user interactions, such as saving or deleting notes.

3. File Handling:
  - Reading and Writing Files: Using `FileReader` and `FileWriter` or `BufferedReader` and `BufferedWriter` to handle text files for saving and loading notes.

My Special Thanks to...

I would like to extend my heartfelt thanks to Punit Vishwakarma Sir, my Java trainer and lab incharge. Your guidance, support, and expertise have been invaluable throughout my learning journey. Your dedication to teaching and your willingness to help have greatly contributed to my understanding of Java. Thank you for being an inspiring mentor and for all the effort you put into ensuring our success.

Thanks for reading,

Raman Shukla