Chapter 13 Lab
Advanced GUI Applications

Lab Objectives

- Be able to add a menu to the menu bar
- Be able to use nested menus
- Be able to add scroll bars, giving the user the option of when they will be seen
- Be able to change the look and feel, giving the user the option of which look and feel to use

Introduction

In this lab we will be creating a simple note taking interface. It is currently a working program, but we will be adding features to it. The current program displays a window which has one item on the menu bar, **Notes**, which allows the user 6 choices. These choices allow the user to store and retrieve up to 2 different notes. It also allows the user to clear the text area or exit the program.

We would like to add features to this program which allows the user to change how the user interface appears. We will be adding another choice on the menu bar called **Views**, giving the user choices about scroll bars and the look and feel of the GUI.

Task #1 Creating a Menu with Submenus

1. Copy the file *NoteTaker.java* (see Code Listing 13.1) from the Student CD or as directed by your instructor.
2. Compile and run the program. Observe the horizontal menu bar at the top which has only one menu choice, **Notes**. We will be adding an item to this menu bar called **Views** that has two submenus. One named **Look and Feel** and one named **Scroll Bars**. The submenu named **Look and Feel** lets the user change the look and feel to **Metal, Motif**, or **Windows**. The submenu named **Scroll Bars** offers the user three choices: **Never, Always**, and **As Needed**. When the user makes a choice, the scroll bars are displayed according to the choice.
3. We want to logically break down the problem and make our program easier to read and understand. We will write separate methods to create each of the vertical menus. The three methods that we will be writing are:
    a. **`createViews`**
    b. **`createScrollBars`**
    c. **`createLookAndFeel`**

    The method headings with empty bodies are provided.
4. Let's start with the **`createLookAndFeel`** method. This will create the first submenu, as shown in **Figure 1**. There are three items on this menu: **Metal, Motif**, and **Windows**. We will create this menu by doing the following:

a. Create a new `JMenu` with the name **Look and Feel**.
b. Create a new `JMenuItem` with the name **Metal**.
c. Add an action listener to the menu item (see the **`createNotes`** method to see how this is done).
d. Add the menu item to the menu.
e. Repeat steps b through d for each of the other two menu items.

5. Similarly, write the **`createScrollBars`** method to create a `JMenu` that has three menu items, **Never, Always**, and **As Needed.** See **Figure 2**.
6. Now that we have our submenus, these menus will become menu items for the **Views** menu. The **`createViews`** method will make the vertical menu shown cascading from the menu choice **Views** as shown in **Figure 3**. We will do this as follows:
   a. Create a new `JMenu` with the name **Views.**
   b. Call the **`createLookAndFeel`** method to create the **Look and Feel** submenu.
   c. Add an action listener to the **Look and Feel** menu.
   d. Add the **Look and Feel** menu to the **Views** menu.
   e. Repeat steps b through d for the **Scroll Bars** menu item, this time calling the **`createScrollBars`** method.
7. Finish creating your menu system by adding the **Views** menu to the menu bar in the constructor.

Task #2 Adding Scrolls Bars and Editing the Action Listener

1. Add scroll bars to the text area by completing the following steps in the constructor:
   a. Create a `JScrollPane` object called **`scrolledText`**, passing in **`theText`**.
   b. Change the line that adds to the `textPanel`, by passing in **`scrolledText`** (which now has **`theText`**).
2. Edit the action listener by adding 6 more branches to the `else-if` logic. Each branch will compare the `actionCommand` to the 6 submenu items: **Metal, Motif, Window, Never, Always,** and **As Needed**.
   a. Each **Look and Feel** submenu item will use a `try-catch` statement to set the look and feel to the appropriate one, displaying an error message if this was not accomplished.
   b. Each **Scroll Bars** submenu item will set the horizontal and vertical scroll bar policy to the appropriate values.
   c. Any components that have already been created need to be updated. This can be accomplished by calling the following method:

      ```
      SwingUtilities.updateComponentTreeUI
      ```

      and passing a reference to the component that you want to update as an argument. Specifically you will need to add the following line to each branch that you just added to the logic structure:

```
SwingUtilities.updateComponentTreeUIgetContentPane());
```

Figure 1

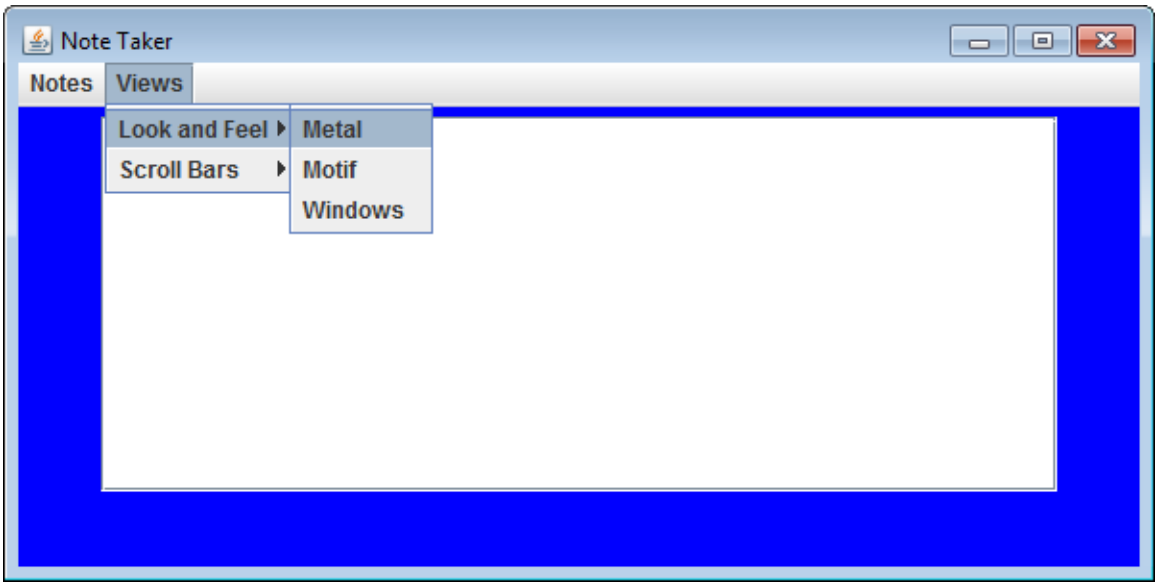


Figure 2

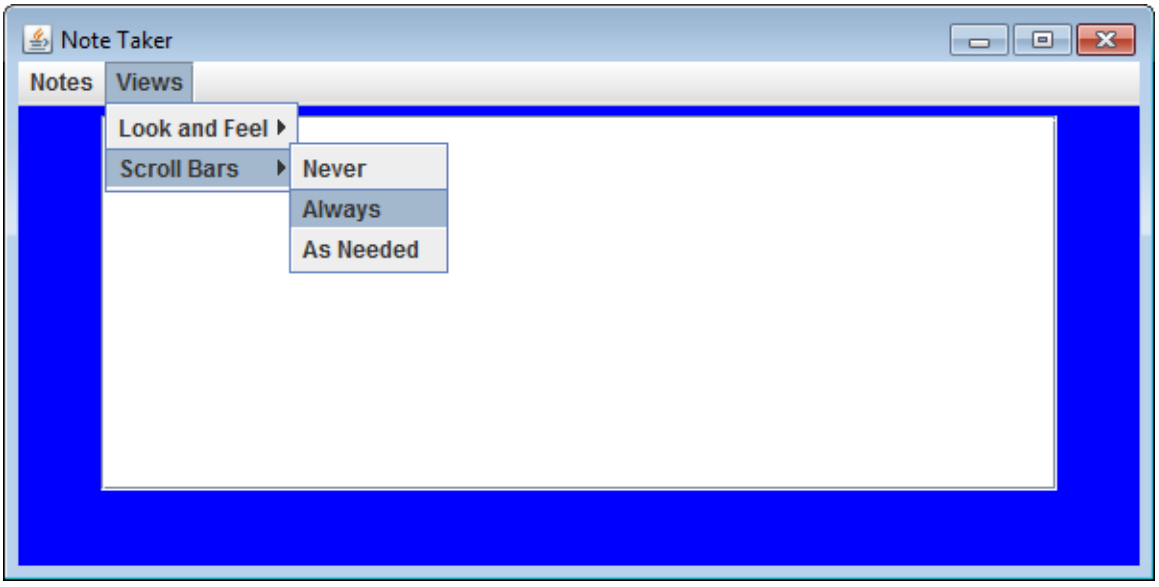
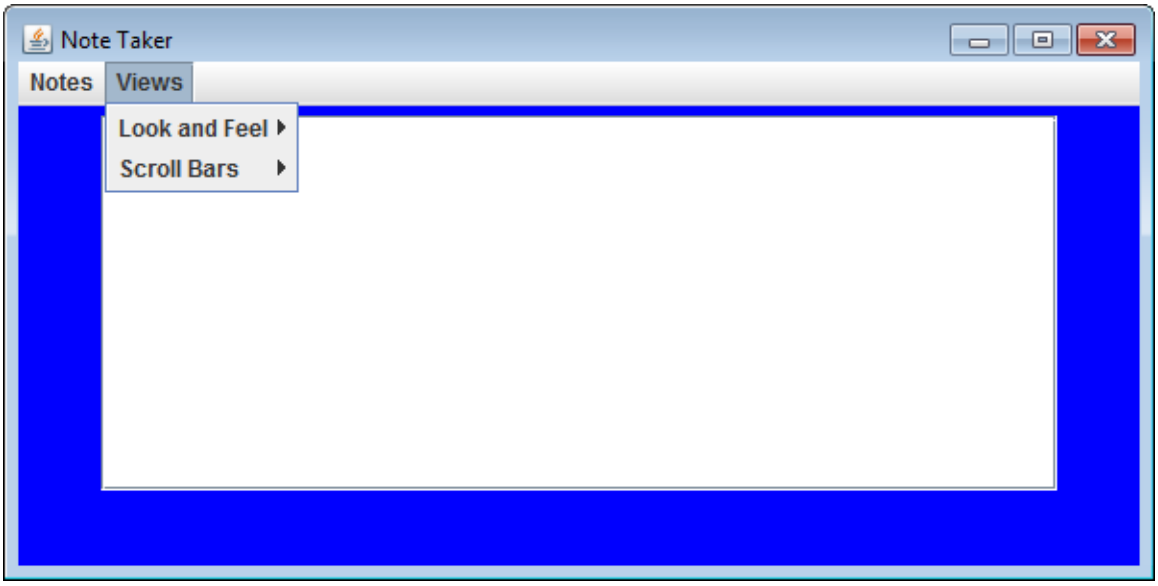


Figure 3



Code Listing 13.1 (`NoteTaker.java`)

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

public class NoteTaker extends JFrame
{
   // Constants for set up of the note taking area
   public static final int WIDTH = 600;
   public static final int HEIGHT = 300;
   public static final int LINES = 13;
   public static final int CHAR_PER_LINE = 45;

   // Objects in GUI
   private JTextArea theText; // Text area to take notes
   private JMenuBar mBar;     // Horizontal menu bar
   private JPanel textPanel;  // Scrolling text area panel
   private JMenu notesMenu;   // Vertical menu for notes

   // THESE ITEMS ARE NOT YET USED
   // YOU WILL BE CREATING THEM IN THIS LAB
   private JMenu viewMenu; // Vertical menu for views
   private JMenu lafMenu;  // Vertical menu look and feel
   private JMenu sbMenu;   // Vertical menu for scroll bar
   private JScrollPane scrolledText;   // Scroll bars
```

```
// Default notes
private String note1 = "No Note 1.";
private String note2 = "No Note 2.";

/**
   Constructor
*/

public NoteTaker()
{
   // Create a closeable JFrame with a specific size
   super("Note Taker");
   setSize(WIDTH, HEIGHT);
   setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);

   // Get contentPane and set layout of the window
   Container contentPane = getContentPane();
   contentPane.setLayout(new BorderLayout());

   // Creates the vertical menus
   createNotes();
   createViews();

   // Creates horizontal menu bar and
   // adds vertical menus to it
   mBar = new JMenuBar();
   mBar.add(notesMenu);

   // ADD THE viewMenu TO THE MENU BAR HERE
   setJMenuBar(mBar);

   // Creates a panel to take notes on
   textPanel = new JPanel();
   textPanel.setBackground(Color.blue);
   theText = new JTextArea(LINES, CHAR_PER_LINE);
   theText.setBackground(Color.white);

   // CREATE A JScrollPane OBJECT HERE CALLED
   // scrolledText AND PASS IN theText, THEN
   // CHANGE THE LINE BELOW BY PASSING IN scrolledText

   textPanel.add(theText);
   contentPane.add(textPanel, BorderLayout.CENTER);
 }
```

```
 /**
   Creates vertical menu associated with Notes
   menu item on menu bar.
*/

 public void createNotes()
 {
   notesMenu = new JMenu("Notes");
   JMenuItem item;

   item = new JMenuItem("Save Note 1");
   item.addActionListener(new MenuListener());
   notesMenu.add(item);

   item = new JMenuItem("Save Note 2");
   item.addActionListener(new MenuListener());
   notesMenu.add(item);

   item = new JMenuItem("Open Note 1");
   item.addActionListener(new MenuListener());
   notesMenu.add(item);

   item = new JMenuItem("Open Note 2");
   item.addActionListener(new MenuListener());
   notesMenu.add(item);

   item = new JMenuItem("Clear");
   item.addActionListener(new MenuListener());
   notesMenu.add(item);

   item = new JMenuItem("Exit");
   item.addActionListener(new MenuListener());
   notesMenu.add(item);
}

/**
   Creates vertical menu associated with Views
   menu item on the menu bar.
*/

public void createViews()
{

}
```

```
/**
   Creates the look and feel submenu.
*/

public void createLookAndFeel()
{

}

/**
   Creates the scroll bars submenu.
*/

public void createScrollBars()
{

}

/**
   Private inner class that handles the Menu object's
   action events.
*/

private class MenuListener implements ActionListener
{

   public void actionPerformed(ActionEvent e)
   {
      String actionCommand = e.getActionCommand();
      if (actionCommand.equals("Save Note 1"))
         note1 = theText.getText();
      else if (actionCommand.equals("Save Note 2"))
         note2 = theText.getText();
      else if (actionCommand.equals("Clear"))
         theText.setText("");
      else if (actionCommand.equals("Open Note 1"))
         theText.setText(note1);
      else if (actionCommand.equals("Open Note 2"))
         theText.setText(note2);
      else if (actionCommand.equals("Exit"))
         System.exit(0);
     // ADD 6 BRANCHES TO THE ELSE-IF STRUCTURE
     // TO ALLOW ACTION TO BE PERFORMED FOR EACH
     // MENU ITEM YOU HAVE CREATED
```

```
            else
               theText.setText("Error in memo interface");
         }
      }

      /**
         The main method creates an instance of the
         NoteTaker class which causes it to display
         its window.
      */

      public static void main(String[] args)
      {
         NoteTaker gui = new NoteTaker();
         gui.setVisible(true);
      }
}
```