211lab12_S11

Lab 12: Graphical User Interfaces (GUI's) in Java

For most of the relevant Information on how to build GUIs in Java, see the following reference: Java GUI Programming Primer

The content below discusses material not explicitly covered in the above reference.

Few people code GUI's by hand these days--it's just too complicated. Typically, one would use a "GUI builder" to drag-and-drop GUI components onto a design canvas, greatly simplifying and speeding up the GUI creation process. But it is very helpful to know the basics of how to build a GUI by hand because one often needs to go in an manually tweak the GUI code even when using a sophisticated GUI builder tool.

Basic structure of a simple GUI application:

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

class MyGUIApp extends JFrame {

    // Use fields for GUI components that need to be referenced outside of the GUI initialization process
(initGUI)

    /**
     * Constructor for the GUI.
     */
    public MyGUIApp() {
        // initialize fields here.  Surrounding your code in a protective try-catch is recommended.

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Causes the application to end if the GUI frame is
closed.
        initGUI();  // initialize the GUI components
    }

    /**
     * Actually start the GUI
     */
    public void start() {
       // Do any last second initializations, if needed.
       setVisible(true);  // Make the GUI visible
    }

    /**
     * Initialize the GUI components
     */
    private void initGUI() {
        Container contentPane = getContentPane(); // all GUI components are added to the frame's ContentPane.
                                                  // You can also just call the accessor each time as well and
not use a local variable.

        // instantiate, initialize and connect all the GUI's components here.
        // Don't forget to add them to the container that holds them, e.g. the ContentPane.

    }

    /**
     * Main method that starts the app.
     * In general, this would not be in this class but rather in a separate "controller" class
     * with slightly different coding (as required for a true Model-View-Controller system).
     * We put it here for now, for simplicity's sake only.
     */
    public static void main(String[] args) {
        // initialize and start the GUI on the GUI thread.
        SwingUtilities.invokeLater(new Runnable() {
           public void run() {
              MyGUIApp view = new MyGUIApp();  // instantiate the GUI but don't show it yet.
              // Model classes would be instantiated here plus any other required objects.
              view.start();  // Start the application by making the GUI visible.
           }
        });
    }
}
```

Some typical GUI initializations:

Set title and size of the frame:

```
setTitle("My Incredible GUI App"); // set the frame's title
setSize(400, 300);  // set frame size to 400x300
```

Panels in main frame:

```
 JPanel controlPnl = new JPanel();
 controlPnl.setBackground(Color.BLUE); // Set panel color to blue
 contentPane.add(controlPnl, BorderLayout.NORTH);   // Fixed-size panel for control components, e.g. buttons,
textfields, etc, at the top of the frame.

 JPanel displayPnl = new JPanel();
 contentPane.add(displayPnl, BorderLayout.CENTER);   // Adjustable-size panel for display components, e.g. text
areas, etc, in the middle of  the frame.
```

A Button with a listener on the control panel:

```
JButton runBtn = new JButton("Run!");
runBtn.addActionListener(new ActionListener() {

      /**
       * This method runs when the button is clicked.
       */
      public void actionPerformed(ActionEvent evt) {
           // Do stuff here.  Generally, delegate to a non-GUI object (the "model").
      }
   });
controlPnl.add(runBtn);
```

A Label and a TextField with a listener on the control panel:

```
 JLabel infoLbl = new JLabel();

 infoLbl.setText("Enter text here:");
 controlPnl.add(infoLbl);

final JTextField infoTF = new JTextField("default text");  // don't need "final" below if this line was
defining a field instead of a local variable.
infoTF.setPreferredSize(new java.awt.Dimension(75, 23)); // set default text field size to 75x23 pixels.
infoTF.addActionListener(new ActionListener() {

      /**
       * This method runs when "Enter" is pressed after entering text into the text field.
       */
      public void actionPerformed(ActionEvent evt) {
           String text = infoTF.getText();  // the text of the text field.
           // Do stuff here. Generally, delegate to a non-GUI object (the "model").

      }
   });

controlPnl.add(infoTF);
```

A text area with scroll bars in the center of the frame:

```
JScrollPane scrollPn = new JScrollPane();
contentPane.add(scrollPn, BorderLayout.CENTER);  // No center panel needed. ScrollPane added directly to frame.
JTextArea displayTA = new JTextArea("initial text");
scrollPn.setViewportView(displayTA);  // put the text area in the scroll pane
```

SerialVersionUID compiler warning:

A very common compiler warning to receive is the following for your frame class and other customized GUI components:

```
The serializable class XXX does not declare a static final serialVersionUID field of type long
```

If you do, add the following field to the class:

```
private static final long serialVersionUID = 42L; // Use a random, unique integer value here
```

For more information, see this web page .