```
Loading vc-cvs...
package caltool.options_ui;

import caltool.options.*;
import caltool.caltool_ui.*;
import mvp.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

/****
 *
 * Initial version of OptionsUI class that constructs and composes its pulldown
 * menu and tabbed dialog.  Each item in the menu is a category of options,
 * with a corresponding pane in the tabbed dialog.  Selection of a menu item
 * makes the tabbed dialog active and selects the pane that corresponds to the
 * menu item.
 *
 */
public class OptionsUI extends CalendarToolWindow {

    /**
     * Construct this with the given screen, model, and parent view.  Construct
     * the panes for the tabbed dialog.  The parent view is used for setting
     * the initial position of this' window relative to the top-level menubar.
     */
    public OptionsUI(Screen screen, Options options,
            CalendarToolUI calToolUI) {
        super(screen, options, calToolUI);

        /*
         * Construct the menu.
         */
        optionsMenu = new OptionsMenu(screen, options, this);

        /*
         * Construct the tabbed pane and add its kids.
         */
        tabs = new JTabbedPane();
        timesAndDatesPanel = new TimesAndDatesOptionsPanel(screen, options);
        fontsPanel = new FontsOptionsPanel(screen, options);
        schedulingPanel = new SchedulingOptionsPanel(screen, options);
        viewingPanel = new ViewingOptionsPanel(screen, options);
        administrativePanel = new AdministrativeOptionsPanel(screen, options);
    }

    /*
     * Compose this by adding the panes to the dialog, adding the button row on
     * the bottom, and composing the menu.
     */
    public Component compose() {

        /*
         * Make the base layer a JPanel, as usual.
         */
        panel = new JPanel();
        window.add(panel);

        /*
         * Set the layout style of the panel to be a vertical box.
         */
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

        /*
         * Set the window title.
         */
        window.setTitle("Options");

        /*
         * Compose the tabbed dialog, button row, and pack things up.
         */
        composeTabs();
        composeButtonRow();
        window.pack();

        /*
         * Compose and return the menu.
         */
        return optionsMenu.compose();

    }

    protected void composeTabs() {

        Box hbox = Box.createHorizontalBox();

        hbox.add(Box.createHorizontalStrut(20));
        hbox.add(tabs);
        hbox.add(Box.createHorizontalStrut(20));

        panel.add(Box.createVerticalStrut(20));
        panel.add(hbox);
        panel.add(Box.createVerticalStrut(20));

        tabs.setPreferredSize(new Dimension(575,475));

        tabs.addTab("Times and Dates", timesAndDatesPanel.compose());
        tabs.addTab("Fonts", fontsPanel.compose());
        tabs.addTab("Scheduling", schedulingPanel.compose());
        tabs.addTab("Viewing", viewingPanel.compose());
        tabs.addTab("Administrative", administrativePanel.compose());

    }

    protected void composeButtonRow() {

        Box hbox = Box.createHorizontalBox();

        /*
         * Construct the buttons.
         */
        JButton applyButton = new JButton("Apply");
```

```
112         JButton saveAsButton = new JButton("Save As ...");
113         JButton loadButton = new JButton("Load ...");
114         JButton clearButton = new JButton("Clear");
115         JButton cancelButton = new JButton("Cancel");
116
117         /*
118          * Attach the appropriate action listeners to each button.
119          */
120         cancelButton.addActionListener(
121             new ActionListener() {
122                 public void actionPerformed(ActionEvent e) {
123                     hide();
124                 }
125             }
126         );
127         // ...
128
129         /*
130          * Set the initial states of the buttons (dummy processing until we
131          * connect with the model).
132          */
133         applyButton.setEnabled(false);
134         clearButton.setEnabled(false);
135
136         /*
137          * Add everything to the hbox and stick it in the panel.
138          */
139         hbox.add(applyButton);
140         hbox.add(Box.createHorizontalStrut(12));
141         hbox.add(saveAsButton);
142         hbox.add(Box.createHorizontalStrut(12));
143         hbox.add(loadButton);
144         hbox.add(Box.createHorizontalStrut(12));
145         hbox.add(clearButton);
146         hbox.add(Box.createHorizontalStrut(12));
147         hbox.add(cancelButton);
148
149 //      panel.add(Box.createVerticalStrut(20));
150         panel.add(hbox);
151         panel.add(Box.createVerticalStrut(20));
152
153     }
154
155     /**
156      * Select the tabbed with the name of the given string.
157      */
158     public void selectTab(String selection) {
159         tabs.setSelectedIndex(tabs.indexOfTab(selection));
160     }
161
162
163     /** The background panel */
164     protected JPanel panel;
165
166     /** The tabbed dialog */
167     protected JTabbedPane tabs;
168
169     /** The times and dates pane */
170     protected TimesAndDatesOptionsPanel timesAndDatesPanel;
171
172     /** The fonts pane */
173     protected FontsOptionsPanel fontsPanel;
174
175     /** The scheduling pane */
176     protected SchedulingOptionsPanel schedulingPanel;
177
178     /** The viewing pane */
179     protected ViewingOptionsPanel viewingPanel;
180
181     /** The administrative pane */
182     protected AdministrativeOptionsPanel administrativePanel;
183
184     /** The menu */
185     protected OptionsMenu optionsMenu;
186
187 }
```