```
Loading vc-cvs...
```

```
package caltool.schedule_ui;

import caltool.schedule.*;
import mvp.*;
import javax.swing.*;
import java.awt.*;

/****
 *
 * Class ScheduleEventDialog provides a view of Event as an input to the
 * scheduleEvent method.  Hence, the dialog is a view of both an Event object
 * as well as the scheduleEvent method.  The data-entry components of the
 * dialog are the Event view.  The 'OK' button is the view of the ScheduleEvent
 * method.
 *
 * The data components consist of JLabels, JTextFields, and a JComboBox.  The
 * 'OK', 'Clear', and 'Cancel' buttons are JButtons.  The description of the <a
 * href= "#compose()"> compose </a> method has details of how the components
 * are laid out in the dialog window.
 *
 * The companion model for ScheduleEventDialog is the Schedule class, since
 * Schedule has the method that must be invoked from the 'OK' button action
 * listener.  See class <a href= "OKScheduleEventButtonListener">
 * OKScheduleEventButtonListener.html </a> for details of how the
 * Schedule.scheduleEvent method is invoked.
 *
 */

public class ScheduleEventDialogGridBag extends View {

    public ScheduleEventDialogGridBag(Screen screen, Schedule schedule) {

        /*
         * Call the parent constructor.
         */
        super(screen, schedule);

        /*
         * Allocate a constraints object for use with the gridbag layout.
         */
        constraints = new GridBagConstraints();
    }

    public Component compose() {

        /*
         * Make a new window for this, which in Java will be a JFrame -- the
         * standard outermost container for a Swing window.
         */
        window = new mvp.Window();

        /*
         * Add a JPanel to the window.  JPanel is the standard background
         * container for holding Swing components.
         */
        panel = new JPanel();
        window.add(panel);

        /*
         * Control the panel layout using a GridBagLayout.
         */
        layout = new GridBagLayout();
        panel.setLayout(layout);

        /*
         * Compose each of the rows in turn by adding them to the grid.
         */
        composeTitleRow();
        composeStartAndEndDateRow();
        composeCategoryAndLocationRow();
        composeButtonRow();

        /*
         * Set the window titlebar.
         */
        window.setTitle("Schedule an Event");

        /*
         * Call JFrame.pack to have Java size up the window properly.
         */
        window.pack();

        /*
         * Return the window to the caller.
         */
        return window;
    }

    protected void composeTitleRow() {

        /*
         * Construct the label and text field for the title component of the
         * dialog.
         */
        JLabel label = new JLabel("Title:");
        JTextField textField = new JTextField();

        /*
         * Insert the label as the first component of a new grid row.
         */
        il(label, 0, 0);
//      insertLabel(label, 0.0, true, 0);

        /*
         * Insert the text field as the second component of the row.
         */
        itf(textField, 1, 0, 300);
//      insertLabeledItem(textField, 1.0, 400, true, 0);

    }

```

```
112     protected void composeStartAndEndDateRow() {
113
114         /*
115          * Construct the labels and text fields.
116          */
117         JLabel startLabel = new JLabel("Start Date:");
118         JTextField startTextField = new JTextField();
119         JLabel endLabel = new JLabel("End Date:");
120         JTextField endTextField = new JTextField();
121
122         /*
123          * Insert all four components.
124          */
125         il(startLabel, 0, 1);
126         itf(startTextField, 1, 1, 200);
127         il(endLabel, 2, 1);
128         itf(startTextField, 3, 1, 200);
129
130         /*
131         insertLabel(startLabel, 0.0, false, 4);
132         insertLabeledItem(startTextField, 1.0, 200, false, 4);
133         insertLabel(endLabel, 1.0, true, 0);
134         insertLabeledItem(endTextField, 1.0, 200, true, 0);
135          */
136     }
137
138     protected void composeCategoryAndLocationRow() {
139
140     }
141
142     protected void composeButtonRow() {
143
144     }
145
146
147     protected void il(JLabel l, int x, int y) {
148         constraints.gridx = x;
149         constraints.gridx = y;
150         constraints.fill = GridBagConstraints.NONE;
151         constraints.anchor = GridBagConstraints.EAST;
152         constraints.insets = new Insets(10, 10, 10, 2);
153         constraints.ipadx = 0;
154         layout.setConstraints(l, constraints);
155         panel.add(l);
156     }
157
158     protected void itf(JTextField tf, int x, int y, int pad) {
159         constraints.gridx = x;
160         constraints.gridx = y;
161         constraints.fill = GridBagConstraints.BOTH;
162         constraints.anchor = GridBagConstraints.WEST;
163         constraints.insets = new Insets(10, 2, 10, 10);
164         constraints.ipadx = pad;
165         layout.setConstraints(tf, constraints);
166         panel.add(tf);
167     }
168
169     protected void insertLabel(JLabel label, double weightx,
170             boolean relative, int width) {
171         constraints.weightx = weightx;
172         constraints.fill = GridBagConstraints.NONE;
173         constraints.anchor = GridBagConstraints.EAST;
174         constraints.insets = new Insets(10, 10, 10, 2);
175         constraints.ipadx = 0;
176         if (relative) constraints.gridwidth = GridBagConstraints.RELATIVE;
177         else constraints.gridwidth = width;
178         layout.setConstraints(label, constraints);
179         panel.add(label);
180     }
181
182     protected void insertLabeledItem(Component item, double weightx, int ipadx,
183             boolean remainder, int width) {
184         constraints.weightx = weightx;
185         constraints.fill = GridBagConstraints.BOTH;
186         constraints.insets = new Insets(10, 2, 10, 10);
187         constraints.ipadx = ipadx;
188         if (remainder) constraints.gridwidth = GridBagConstraints.REMAINDER;
189         else constraints.gridwidth = 1;
190         layout.setConstraints(item, constraints);
191         panel.add(item);
192     }
193
194     /** The grid bag that controls component layout in the panel */
195     protected GridBagLayout layout;
196
197     /** The background panel of this */
198     protected JPanel panel;
199
200     /** Gridbag constraints used to control the layout of each dialog
201         component. */
202     GridBagConstraints constraints;
203
204 }
```