```
 1  package caltool.view.schedule;
 2
 3  import caltool.model.schedule.*;
 4  import caltool.view.*;
 5  import mvp.*;
 6  import javax.swing.*;
 7  import java.awt.*;
 8  import java.awt.event.*;
 9
10  /****
11   *
12   * Class ScheduleEventDialog provides a view of Event as an input to the
13   * scheduleEvent method.  Hence, the dialog is a view of both an Event object
14   * as well as the scheduleEvent method.  The data-entry components of the
15   * dialog constitute the Event view.  The 'OK' button is the view of the
16   * scheduleEvent method.
17   *                                                                        <p>
18   * The data components consist of JLabels, JTextFields, and a JComboBox.  The
19   * 'OK', 'Clear', and 'Cancel' buttons are JButtons.  The description of the <a
20   * href= "#compose()"> compose </a> method has details of how the components
21   * are laid out in the dialog window.
22   *                                                                        <p>
23   * The companion model for ScheduleEventDialog is the <a href= "Schedule.html">
24   * Schedule </a> class, since Schedule has the method that is invoked from the
25   * 'OK' button action listener.  See class <a href=
26   * "OKScheduleEventButtonListener"> OKScheduleEventButtonListener.html </a> for
27   * details of how the Schedule.scheduleEvent method is invoked.
28   *
29   * @author Gene Fisher (gfisher@calpoly.edu)
30   * @version 13apr15
31   *
32   */
33  public class ScheduleEventDialog extends View {
34
35      /**
36       * Construct this with the given Schedule as companion model.
37       */
38      public ScheduleEventDialog(Screen screen, Schedule schedule,
39              CalendarToolUI calToolUI) {
40
41          /*
42           * Call the parent constructor.
43           */
44          super(screen, schedule);
45
46          /*
47           * Set the maximum component height and width.  These the height value
48           * was empirically derived, i.e., I fiddled around with it while
49           * looking at the display.  The width value is presumably as big as the
50           * biggest screen we'll come across.  If not, deal with it.
51           */
52          maxComponentHeight = 1.9;
53          maxComponentWidth = 2000;
54      }
55
56      /**
57       * Compose this as a vertical Box of four rows.  Each row is a horizontal
58       * Box.  The first row contains a labeled text field for the event title.
59       * The second row has labeled text fields for the start and end dates.  The
60       * third row has a labeled combo box for the category selection and a
61       * labeled combo box for the event security.  Finally, the fourth row has
62       * the 'OK', 'Clear', and 'Cancel' buttons.
63       *                                                                    <p>
64       * Vertical and horizontal struts are used for spacing among all of the
65       * components.
66       */
67      public Component compose() {
68
69          /*
70           * Make a new window for this, which in Java will be a JFrame -- the
71           * standard outermost container for a Swing window.
72           */
73          window = new mvp.Window();
74
75          /*
76           * Add a JPanel to the window.  JPanel is the standard background
77           * container for holding Swing components.
78           */
79          panel = new JPanel();
80          window.add(panel);
81
82          /*
83           * Set the layout style of the panel to be a vertical box.
84           */
85          panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
86
87          /*
88           * Compose each of the rows and add to the vertically laid out panel.
89           * Put some spacing between each row, in the form of a vertical strut.
90           */
91          panel.add(Box.createVerticalStrut(15));
92          panel.add(composeTitleRow());
93          panel.add(Box.createVerticalStrut(15));
94          panel.add(composeStartAndEndDateRow());
95          panel.add(Box.createVerticalStrut(15));
96          panel.add(composeCategoryAndSecurityRow());
97          panel.add(Box.createVerticalStrut(25));
98          panel.add(composeButtonRow());
99          panel.add(Box.createVerticalStrut(15));
100
101         /*
102          * Set the window titlebar.
103          */
104         window.setTitle("Schedule an Event");
105
106         /*
107          * Call JFrame.pack to have Java size up the window properly.
108          */
109         window.pack();
110
111         /*
112          * Return the window to the caller.
```

```
113          */
114         return window;
115     }
116 
117     /**
118      * Compose the title row using a JLabel and JTextField.
119      */
120     protected Box composeTitleRow() {
121 
122         Box hbox = Box.createHorizontalBox();
123 
124         /*
125          * Construct the label and text field.
126          */
127         JLabel label = new JLabel("Title:  ");
128         titleTextField = new JTextField();
129 
130         /*
131          * Set the label font color to black, since I don't care for the
132          * default "Java blue".
133          */
134         label.setForeground(Color.black);
135 
136         /*
137          * Set the max height of the text field to keep it from resizing
138          * vertically in the vertical box layout of the panel.
139          */
140         titleTextField.setMaximumSize(
141             new Dimension(maxComponentWidth, (int)(maxComponentHeight *
142                 titleTextField.getFont().getSize())));
143 
144         /*
145          * Add the label and text field to the hbox and return it.  Use
146          * horizontal struts for spacing.
147          */
148         hbox.add(Box.createHorizontalStrut(15));
149         hbox.add(label);
150         hbox.add(titleTextField);
151         titleTextField.setAlignmentX(Component.RIGHT_ALIGNMENT);
152         hbox.add(Box.createHorizontalStrut(15));
153         return hbox;
154 
155     }
156 
157     /**
158      * Compose the start/end date row using two pairs of JLabel and JTextField.
159      */
160     protected Box composeStartAndEndDateRow() {
161 
162         Box hbox = Box.createHorizontalBox();
163 
164         /*
165          * Construct the labels and text fields.  See internal comments in the
166          * composeTitle method for further explanatory details.
167          */
168         JLabel startLabel = new JLabel("Start Date:  ");
169         startLabel.setForeground(Color.black);
170         startDateTextField = new JTextField(15);
171         startDateTextField.setMaximumSize(
172             new Dimension(maxComponentWidth, (int)(maxComponentHeight *
173                 startDateTextField.getFont().getSize())));
174         JLabel endLabel = new JLabel("End Date:  ");
175         endLabel.setForeground(Color.black);
176         endDateTextField = new JTextField(15);
177         endDateTextField.setMaximumSize(
178             new Dimension(maxComponentWidth, (int)(maxComponentHeight *
179                 startDateTextField.getFont().getSize())));
180 
181         /*
182          * Add them to the hbox and return it.
183          */
184         hbox.add(Box.createHorizontalStrut(15));
185         hbox.add(startLabel);
186         hbox.add(startDateTextField);
187         hbox.add(Box.createHorizontalStrut(10));
188         hbox.add(endLabel);
189         hbox.add(endDateTextField);
190         hbox.add(Box.createHorizontalStrut(15));
191         return hbox;
192 
193     }
194 
195     /**
196      * Compose the category/security row using a JComboBox and JTextField, with
197      * JLabels next to each.
198      */
199     protected Box composeCategoryAndSecurityRow() {
200 
201         Box hbox = Box.createHorizontalBox();
202 
203         /*
204          * Construct the labels and text fields.  See internal comments in the
205          * composeTitle method for further explanatory details.
206          */
207         JLabel categoryLabel = new JLabel("Category:  ");
208         categoryLabel.setForeground(Color.black);
209         categoryComboBox = new JComboBox();
210         categoryComboBox.addItem("none");
211 
212         JLabel securityLabel = new JLabel("Security:  ");
213         securityLabel.setForeground(Color.black);
214         String[] selections = {"public", "private"};
215         securityComboBox = new JComboBox(selections);
216 
217         /*
218          * Add them to the hbox and return it.
219          */
220         hbox.add(Box.createHorizontalStrut(15));
221         hbox.add(categoryLabel);
222         hbox.add(categoryComboBox);
223         hbox.add(Box.createHorizontalStrut(10));
224         hbox.add(securityLabel);
```

```
225         hbox.add(securityComboBox);
226         hbox.add(Box.createHorizontalStrut(15));
227         return hbox;
228
229     }
230
231     /**
232      * Compose the buttons row with three JButtons.  The action listeners for
233      * Clear and Cancel buttons are straightforward.  The action listener for
234      * the OK button is responsible for communication with the Schedule model.
235      * See the description of <a href= "OKScheduleEventButtonListener.html>"
236      * OKScheduleEventButtonListener </a> for explanatory details.
237      *
238      */
239     protected Box composeButtonRow() {
240
241         Box hbox = Box.createHorizontalBox();
242
243         /*
244          * Construct the three buttons.
245          */
246         JButton okButton = new JButton("OK");
247         JButton clearButton = new JButton("Clear");
248         JButton cancelButton = new JButton("Cancel");
249
250         /*
251          * Attach the appropriate action listeners to each button.
252          */
253         okButton.addActionListener(
254             new OKScheduleEventButtonListener((Schedule) model, this));
255
256         clearButton.addActionListener(
257             new ActionListener() {
258                 public void actionPerformed(ActionEvent e) {
259                     clear();
260                 }
261             }
262         );
263
264         cancelButton.addActionListener(
265             new ActionListener() {
266                 public void actionPerformed(ActionEvent e) {
267                     hide();
268                 }
269             }
270         );
271
272         /*
273          * Add them to the hbox and return it.
274          */
275         hbox.add(okButton);
276         hbox.add(Box.createHorizontalStrut(30));
277         hbox.add(clearButton);
278         hbox.add(Box.createHorizontalStrut(30));
279         hbox.add(cancelButton);
280         return hbox;
281
282     }
283
284     /**
285      * Clear each of the text fields of this to empty.  Reset the combo box to
286      * no selection.  NOTE: This method needs to be refined to use default
287      * values for clearing, once options and defaults functionality is
288      * implemented.
289      */
290     protected void clear() {
291         titleTextField.setText("");
292         startDateTextField.setText("");
293         endDateTextField.setText("");
294         categoryComboBox.setSelectedIndex(0);
295         securityComboBox.setSelectedIndex(0);
296     }
297
298     /** The background panel of this */
299     protected JPanel panel;
300
301     /** The title text field */
302     protected JTextField titleTextField;
303
304     /** The start date text field */
305     protected JTextField startDateTextField;
306
307     /** The end date text field */
308     protected JTextField endDateTextField;
309
310     /** The Categories combo box */
311     protected JComboBox categoryComboBox;
312
313     /** The security text field */
314     protected JComboBox securityComboBox;
315
316     /** The max height of a text field or combobox; this is necessary since
317         these components stretch when the outer frame is resized, and look very
318         funky when they do. */
319     protected final double maxComponentHeight;
320
321     /** The max width of any component; this is only necessary because the max
322         height cannot be set separately, so we must pick some max width. */
323     protected final int maxComponentWidth;
324
325 }
```