

```

1  package caltool.view.view;
2
3  import caltool.model.view.*;
4  import caltool.view.*;
5  import mvp.*;
6  import java.util.*;
7  import javax.swing.*;
8  import javax.swing.table.*;
9  import java.awt.*;
10
11  /**
12   *
13   * Class AppointmentsListDisplay is the companion view of an Appointments
14   * list. The display is a JTable. The columns of the table are fixed, per the
15   * requirements. The number of rows and number of rows visible are controlled
16   * by options settings. The default number of rows is the number of
17   * appointments in the three-week period starting one week from today's date.
18   * The default number of visible rows is 20.
19   *
20   * The model data for this display come from the <a href= ../view/Lists.html>
21   * Lists</a> model class. A DefaultTableModel is used as an adaptor between
22   * the model data and the JTable display. See the method and data field
23   * documentation for further explanation.
24   */
25  public class AppointmentsListDisplay extends CalendarToolWindow {
26
27      /**
28       * Construct with the given screen and Lists model. A local
29       * DefaultTableModel is constructed to hold the raw data collected from the
30       * model.
31       */
32      public AppointmentsListDisplay(Screen s, Lists lists,
33          CalendarToolUI calToolUI) {
34          super(s, lists, calToolUI);
35          String[] columnNames = {
36              "Title", "Date", "Time", "Duration", "Rekurs?", "Category",
37              "Location", "Security", "Priority"};
38
39          localData = new DefaultTableModel(columnNames, 0);
40          table = new JTable(localData);
41          displayedOnce = false;
42      }
43
44      /**
45       * Compose the initial layout with column headings and no row data.
46       */
47      public Component compose() {
48
49          JScrollPane scrollPane = new JScrollPane(table);
50
51          table.setPreferredScrollableViewportSize(new Dimension(
52              700, 20 * table.getRowHeight()));
53          table.setEnabled(false);
54          table.setShowGrid(true);
55          table.setShowHorizontalLines(true);
56          table.setShowVerticalLines(true);
57
58          window.add(scrollPane);
59          window.setTitle("Appointments, sorted by Date");
60
61          return window;
62      }
63
64      /**
65       * Display the model data produced by the method <a href=
66       * ../view/Lists.html#viewAppointmentsList()>
67       * Lists.viewAppointmentsList</a>. The height of the display is based on
68       * an option setting, independent from the length of list returned from the
69       * model method.
70       *
71       * In the current preliminary implementation, the height is set to the
72       * default value of 20 rows. This implementation will be refined to call
73       * an appropriate method in the options package.
74       */
75      public void update(Observable o, Object arg) {
76
77          /*
78           * Clear out all of the data rows.
79           */
80          localData.setRowCount(0);
81
82          /*
83           * Populate the data rows with model data, which come in the form of an
84           * array of AppointmentItems.
85           */
86          Object[] items = ((Lists)model).viewAppointmentsList();
87          for (int i = 0; i < items.length; i++) {
88              populateRow(i, (AppointmentListItem) items[i]);
89          }
90
91          if (!displayedOnce) {
92              displayedOnce = true;
93              window.pack();
94          }
95      }
96
97      /**
98       * Populate the ith table row with the data from the given appointment list
99       * item.
100      */
101      protected void populateRow(int i, AppointmentListItem item) {
102          localData.addRow(item.toArray());
103      }
104
105      /**
106       * Local data model. This is not the real data model, but rather a
107       * view-specific table model that is constructed by extracting data from
108       * the real model using the viewAppointmentsList method. There are in
109       * fact no persistent list data on the model side; rather, all calendar
110       * lists are computed dynamically from the underlying CalendarDB.
111       */
112      In this way, DefaultTableModel is being used as a form of adaptor class

```

caltool/view/view

## AppointmentsListDisplay.java

```
113         between the Lists model and the JTable-based display. */
114     protected DefaultTableModel localData;
115
116     /** The display view */
117     protected JTable table;
118
119     /** Flag that's true after the display has been shown the first time. */
120     boolean displayedOnce;
121
122 }
```