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

import caltool.view.*;
import caltool.caltool_ui.*;
import mvp.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;
import java.awt.*;

/****
 *
 * Class AppointmentsListDisplay is the companion view of an Appointments
 * list.  The display is a JTable.  The columns of the table are fixed, per the
 * requirements.  The number of rows and number of rows visible are controlled
 * by options settings.  The default number of rows is the number of
 * appointments in the three-week period starting one week from today's date.
 * The default number of visible rows is 20.
 *                                                                          <p>
 * The model data for this display come from the <a href= ../view/Lists.html>
 * Lists</a> model class.  A DefaultTableModel is used as an adaptor between
 * the model data and the JTable display.  See the method and data field
 * documentation for further explanation.
 */
public class AppointmentsListDisplay extends CalendarToolWindow {

    /**
     * Construct with the given screen and Lists model.  A local
     * DefaultTableModel is constructed to hold the raw data collected from the
     * model.
     */
    public AppointmentsListDisplay(Screen s, Lists lists,
            CalendarToolUI calToolUI) {
        super(s, lists, calToolUI);
        String[] columnNames = {
            "Title", "Date", "Time", "Duration", "Recurs?", "Category",
            "Location", "Security", "Priority"};

        localData = new DefaultTableModel(columnNames, 0);
        table = new JTable(localData);
        displayedOnce = false;
    }

    /**
     * Compose the initial layout with column headings and no row data.
     */
    public Component compose() {

        JScrollPane scrollPane = new JScrollPane(table);

        table.setPreferredScrollableViewportSize(new Dimension(
            700, 20 * table.getRowHeight()));
        table.setEnabled(false);
        table.setShowGrid(true);
        table.setShowHorizontalLines(true);
        table.setShowVerticalLines(true);

        window.add(scrollPane);
        window.setTitle("Appointments, sorted by Date");

        return window;
    }

    /**
     * Display the model data produced by the method <a href=
     * ../view/Lists.html#viewAppointmentsList()>
     * Lists.viewAppointmentsList</a>.  The height of the display is based on
     * an option setting, independent from the length of list returned from the
     * model method.
     *
     * In the current preliminary implementation, the height is set to the
     * default value of 20 rows.  This implementation will be refined to call
     * an appropriate method in the options package.
     */
    public void update(Observable o, Object arg) {

        /*
         * Clear out all of the data rows.
         */
        localData.setRowCount(0);

        /*
         * Populate the data rows with model data, which come in the form of an
         * array of AppointmentItems.
         */
        Object[] items = ((Lists)model).viewAppointmentsList();
        for (int i = 0; i < items.length; i++) {
            populateRow(i, (AppointmentListItem) items[i]);
        }

        if (! displayedOnce) {
            displayedOnce = true;
            window.pack();
        }

    }

    /**
     * Populate the ith table row with the data from the given appointment list
     * item.
     */
    protected void populateRow(int i, AppointmentListItem item) {
                localData.addRow(item.toArray());
    }

    /** Local data model.  This is not the real data model, but rather a
        view-specific table model that is constructed by extracting data from
        the real model using the viewAppointmentsList method.  There are in
        fact no persistent list data on the model side; rather, all calendar
        lists are computed dynamically from the underlying CalendarDB.
                                                                          <p>
```

```
        In this way, DefaultTableModel is being used as a form of adaptor class
        between the Lists model and the JTable-based display. */
    protected DefaultTableModel localData;

    /** The display view */
    protected JTable table;

    /** Flag that's true after the display has bee shown the first time. */
    boolean displayedOnce;

}
```