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

import caltool.schedule.*;
import caltool.schedule_ui.*;
import caltool.caltool_ui.*;
import mvp.*;
import java.util.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

/****
 *
 * Class AppointmentEditor specializes ScheduleAppointmentDialog to provide
 * editing access to scheduled appointments.  An appointment editor has the
 * same data fields as the scheduling dialog.  The editor display differs from
 * the scheduling dialog as follows:
 *                                                                 <ol type=a><li>
 *    there is a time and date summary in the editor, just below the window
 *    banner;
 *                                                                            <li>
 *    the command buttons at the bottom of the item display are different than
 *    in the scheduling dialog.
 *                                                                           </ol>
 * Left and right arrow buttons at the top of the display are used to view the
 * previous and next scheduled item.  The most important difference between the
 * editor the scheduling dialog are the command buttons along the bottom of the
 * display window. Specifically, the scheduling dialog has 'OK', 'Clear', and
 * 'Cancel' buttons, whereas the editor has 'Change', 'Delete', and 'Clear'.
 *
 */

public class AppointmentEditor extends ScheduleAppointmentDialog {

    /**
     * Construct this with the given Schedule as companion model.
     */
    public AppointmentEditor(Screen screen, Schedule schedule,
            CalendarToolUI calToolUI) {
        super(screen, schedule, calToolUI);
    }

    public Component compose() {
        super.compose();
        window.setTitle("Scheduled Appointment");
        return window;
    }

    /**
     * Stick in the date summary row at the top of main panel, then call the
     * parent composeRows.  It will do everything as in the scheduling dialog,
     * except it will call this' specialized version of composeButtonRow.
     */
    protected void composeRows() {
        panel.add(composeDateSummary());
        super.composeRows();
    }

    /**
     * Compose the date summary row consisting of a three-button group on the
     * left and a date string in the center.  The button group has a
     * left-pointing previous arrow, a 'Today' button, and a right-pointing
     * next arrow.  The date string is the complete time and date of the
     * scheduled appointment.
     *                                                                          <p>
     * This particular layout is accomplished with an outer JPanel with an
     * overlay layout, containing two hboxes with left- and center-alignments.
     * This allows two different horizontal layouts to appear in the same
     * horizontal row of the display.
     */
    protected JPanel composeDateSummary() {
        JPanel outer = new JPanel();
        outer.setLayout(new OverlayLayout(outer));
        outer.setBorder(BorderFactory.createLineBorder(Color.black));
        Box hbox1 = Box.createHorizontalBox();
        Box hbox2 = Box.createHorizontalBox();

        outer.add(hbox1);
        outer.add(hbox2);

        return outer;
    }

    /**
     * Compose the buttons row with three JButtons, a la the parent version of
     * this method, q.v.
     */
    protected Box composeButtonRow() {

        Box hbox = Box.createHorizontalBox();

        /*
         * Construct the three buttons.
         */
        JButton changeButton = new JButton("Change");
        JButton deleteButton = new JButton("Delete");
        JButton clearButton = new JButton("Clear");

        /*
         * Attach the appropriate action listeners to each button.
         */
        changeButton.addActionListener(
            new ChangeAppointmentButtonListener((Schedule) model, this));

        deleteButton.addActionListener(
            new DeleteAppointmentButtonListener((Schedule) model, this));

        clearButton.addActionListener(
            new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    clear();
```

```java
                }
            }
        );

        /*
         * Add them to the hbox and return it.
         */
        hbox.add(changeButton);
        hbox.add(Box.createHorizontalStrut(30));
        hbox.add(deleteButton);
        hbox.add(Box.createHorizontalStrut(30));
        hbox.add(clearButton);
        return hbox;

    }

    /**
     * Display the model data for the currently selected appointment.  This
     * method is only invoked if the current selected item is in fact an
     * appointment.  The appointment is sent in the second arg.  See the <a
     * href= "ItemEditor.html"> ItemEditor </a> for the details of how this
     * method is invoked.
     */
    public void update(Observable o, Object arg) {

        Appointment appt = (Appointment) arg;

        titleTextField.setText(appt.getTitle());
        startDateTextField.setText(appt.getDate().toString());
        if (appt.getEndDate() != null) {
            endDateTextField.setText(appt.getEndDate().toString());
        }
        else {
            endDateTextField.setText("");
        }
        startTimeTextField.setText(appt.getStartTime().toString());
        durationTextField.setText(appt.getDuration().toString());
        recurringInfo.update(null, appt.getRecurringInfo());
        categoryComboBox.setSelectedItem(appt.getCategory().toString());
        locationComboBox.setSelectedItem(appt.getCategory().toString());
        securityComboBox.setSelectedIndex(appt.getSecurity().ordinal());
        priorityComboBox.setSelectedIndex(appt.getPriority().ordinal());
        remindInfo.update(null, appt.getRemindInfo());
        detailsTextArea.setText(appt.getDetails());
    }

}
```