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

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

/****
 *
 * Class ScheduleMeetingDialog provides a view of Meeting as an input to the
 * meeting-scheduling methods listMeetingTimes, setMeetingOptions, and
 * confirmMeeting.  ScheduleMeetingDialog extends ScheduleAppointmentDialog,
 * since the two dialogs have much in common.  (This is to be expected given
 * that the Meeting model class is an extension of the Appointment model
 * class.)  The differences between the meeting and appointment dialogs are as
 * follows:
 *                                                                    <ul><li>
 *    The Start Date row is replaced with two rows for earliest and latest
 *    possible dates.
 *                                                                   </li><li>
 *    The label on the End Date text field is changed to `Earliest End Date'.
 *                                                                   </li><li>
 *    There is a `Latest End Date' row below the `Earliest End Date' row.
 *                                                                   </li><li>
 *    The configuration of the Recurring row has a monthly Details button
 *    instead of the days-of-the-week checkboxes.
 *                                                                   </li><li>
 *    There is an Attendees text area between the Remind row and the Details
 *    text area.
 *                                                                   </li><li>
 *    There is a Minutes row below Details.
 *                                                                   </li><li>
 *    The buttons at the bottom of the dialog have `List Times' and `Options'
 *    instead of `OK'.
 *                                                                  </li></ul>
 * The companion model for ScheduleAppointmentDialog is the <a href=
 * "../schedule/Schedule.html"> Schedule </a> class, since Schedule has the
 * methods that are invoked from the action listeners attached to the `List
 * Times' and `Options' buttons.  See <a href=
 * "ListMeetingTimesButtonListener.html"> ListMeetingTimesButtonListener </a>
 * and <a href= "MeetingOptionsButtonListener.html">
 * MeetingOptionsButtonListener </a> for details of how the appropriate
 * Schedule methods are invoked.
 *                                                                         <p>
 * The additional design comments in the definition of <a href=
 * ScheduleAppointmentDialog.html> ScheduleAppointmentDialog </a> are also
 * relevant here.
 *
 */

public class ScheduleMeetingDialog extends ScheduleAppointmentDialog {

    public ScheduleMeetingDialog(Screen screen, Schedule schedule,
            CalendarToolUI calToolUI) {
        super(screen, schedule, calToolUI);
    }

    /**
     * Compose this in six parts: (1) a top part consisting of the title,
     * possible (start) dates, end dates, start times, and duration components;
     * (2) a part consisting of recurring info components; (3) a middle part
     * with category, location, security, and priority; (4) reminder info
     * components; (5) a bottom part with attendees, details, and minutes
     * components; (6) the button row consisting of the 'OK', 'Clear', and
     * 'Cancel' buttons.
     */
    public Component compose() {

        /*
         * Add a JPanel to this' window, which was created in the parent class'
         * constructor.  JPanel is the standard background container for
         * holding Swing components.
         */
        panel = new JPanel();
        window.add(panel);

        /*
         * Set the layout style of the panel to be a vertical box.
         */
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

        /*
         * Compose the content rows.
         */
        composeRows();

        /*
         * Set the window titlebar.
         */
        window.setTitle("Schedule a Meeting");

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

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

    /**
     * Compose each of the rows and add to the vertically laid out panel.
     * Put some around spacing between each row, in the form of a vertical
     * strut.
     */
    protected void composeRows() {
        panel.add(Box.createVerticalStrut(15));
```

```
        panel.add(composeTopPart());
        panel.add(Box.createVerticalStrut(15));
        panel.add(composeRecurringInfo());
        panel.add(Box.createVerticalStrut(15));
        panel.add(composeMiddlePart());
        panel.add(Box.createVerticalStrut(15));
        panel.add(composeRemindInfo());
        panel.add(Box.createVerticalStrut(15));
        panel.add(composeBottomPart());
        panel.add(Box.createVerticalStrut(15));
        panel.add(composeButtonRow());
        panel.add(Box.createVerticalStrut(15));
    }

    /**
     * Compose the top part of the dialog, consisting of the title, possible
     * (start) dates, start times, end date, and duration.  The components are
     * laid out in a five-row vertical box.  The title row is on top; it is
     * composed as a horizontal box containing a JLabel and JTextField.  The
     * second and third rows of top-part components is are horizontal boxes,
     * composed in turn of two horizontal boxes containing JLabel/JTextField
     * pairs for the possible dates and start times.  The fourth row consists
     * of a JLabel/JTextField pair for the end date and the duration.  The
     * duration is in turn a horizontal box consisting of a JLabel, and two
     * vertical box JLabel/JTextField pairs for the hour and minute components
     * of the duration.  The fifth row is a single MultLineLabel/JTextfield
     * pair for the lates end date.
     */
    protected Box composeTopPart() {
        Box vbox = Box.createVerticalBox();

        vbox.add(composeTitleRow());
        vbox.add(Box.createVerticalStrut(15));
        vbox.add(composeEarliestStartDateRow());
        vbox.add(Box.createVerticalStrut(15));
        vbox.add(composeLatestStartDateRow());
        vbox.add(Box.createVerticalStrut(15));
        vbox.add(composeEarliestEndDateRow());
        vbox.add(Box.createVerticalStrut(15));
        vbox.add(composeLatestEndDateRow());

        return vbox;
    }

    /**
     * Compose the row containing the earliest possible (start) date and time.
     */
    protected Box composeEarliestStartDateRow() {
        return composeDateRow("Earliest");
    }

    /**
     * Compose the row containing the latest possible (start) date and time.
     */
    protected Box composeLatestStartDateRow() {
        return composeDateRow("Latest");
    }

    /**
     * Compose a date row, with s = "Earliest" or "Latest", and the other args
     * eqal to the earliert or latest data fields, resp.
     */
    protected Box composeDateRow(String s) {

        Box hbox = Box.createHorizontalBox();

        /*
         * Construct the labels and text fields.  See internal comments in the
         * composeTitle method for further explanatory details.
         */
        JLabel dateLabel = new JLabel(s + " Date:  ");
        if (s.equals("Earliest"))
            startDateLabel = dateLabel;
        else
            latestStartDateLabel = dateLabel;

        JTextField dateTextField = new JTextField(15);
        dateTextField.setMaximumSize(
            new Dimension(maxComponentWidth, (int)(maxComponentHeight *
                dateTextField.getFont().getSize())));
        if (s.equals("Earliest"))
            startDateTextField = dateTextField;
        else
            latestStartDateTextField = dateTextField;

        JLabel timeLabel = new JLabel(s + " Time:  ");
        timeLabel.setForeground(Color.black);

        JTextField timeTextField = new JTextField(15);
        timeTextField.setMaximumSize(
            new Dimension(maxComponentWidth, (int)(maxComponentHeight *
                timeTextField.getFont().getSize())));
        if (s.equals("Earliest"))
            startTimeTextField = timeTextField;
        else
            latestStartTimeTextField = timeTextField;

        /*
         * Add them to the hbox and return it.
         */
        hbox.add(Box.createHorizontalStrut(15));
        hbox.add(dateLabel);
        hbox.add(dateTextField);
        hbox.add(Box.createHorizontalStrut(10));
        hbox.add(timeLabel);
        hbox.add(timeTextField);
        hbox.add(Box.createHorizontalStrut(15));
        return hbox;

    }

    /**
```

```
     * Compose the row containing the earliest end date and duration.
     */
    protected Box composeEarliestEndDateRow() {

        Box hbox = Box.createHorizontalBox();

        /*
         * Construct the labels and text fields.  See internal comments in the
         * composeTitle method for further explanatory details.
         */
        endDateLabel = new JLabel("Earliest End Date:  ");
        endDateLabel.setForeground(Color.black);
        endDateLabel.setEnabled(false);
        endDateLabel.setAlignmentY(Box.BOTTOM_ALIGNMENT);

        endDateTextField = new JTextField(15);
        endDateTextField.setMaximumSize(
            new Dimension(maxComponentWidth, (int)(maxComponentHeight *
                endDateTextField.getFont().getSize())));
        endDateTextField.setEnabled(false);
        endDateTextField.setAlignmentY(Box.BOTTOM_ALIGNMENT);

        JLabel durationLabel = new JLabel("Duration:  ");
        durationLabel.setForeground(Color.black);
        durationLabel.setAlignmentY(Box.BOTTOM_ALIGNMENT);

        durationTextField = new JTextField(15);
        durationTextField.setMaximumSize(
            new Dimension(maxComponentWidth, (int)(maxComponentHeight *
                durationTextField.getFont().getSize())));
        durationTextField.setAlignmentY(Box.BOTTOM_ALIGNMENT);

        /*
         * Add them to the hbox and return it.
         */
        hbox.add(Box.createHorizontalStrut(15));
        hbox.add(endDateLabel);
        hbox.add(endDateTextField);
        hbox.add(Box.createHorizontalStrut(10));
        hbox.add(durationLabel);
        hbox.add(durationTextField);
        hbox.add(Box.createHorizontalStrut(15));
        return hbox;

    }

    /**
     * Compose the row containing the latest end date.
     */
    protected Box composeLatestEndDateRow() {

        Box hbox = Box.createHorizontalBox();

        /*
         * Construct the labels and text fields.  See internal comments in the
         * composeTitle method for further explanatory details.
         */
        latestEndDateLabel = new JLabel("Latest End Date:  ");
        latestEndDateLabel.setForeground(Color.black);
        latestEndDateLabel.setEnabled(false);

        latestEndDateTextField = new JTextField(15);
        latestEndDateTextField.setMaximumSize(
            new Dimension(maxComponentWidth, (int)(maxComponentHeight *
                latestEndDateTextField.getFont().getSize())));
        latestEndDateTextField.setEnabled(false);

        /*
         * Force decent-looking layout with a fixed-size horizontal strut.
         * There's got to be a better way to do this, such as using the width
         * of the Duration component, but so far the way has eluded me.
         */
        int blankSpacing = 248;

        /*
         * Add them to the hbox and return it.
         */
        hbox.add(Box.createHorizontalStrut(15));
        hbox.add(latestEndDateLabel);
        hbox.add(latestEndDateTextField);
        hbox.add(Box.createHorizontalStrut(blankSpacing));
        return hbox;

    }

    /**
     * Compose the bottom part consisting of attendees, details, and minutes
     * fields.
     */
    public Box composeBottomPart() {
        Box vbox = Box.createVerticalBox();

        vbox.add(composeAttendees());
        vbox.add(Box.createVerticalStrut(15));
        vbox.add(composeDetails());
        vbox.add(Box.createVerticalStrut(15));
        vbox.add(composeMinutes());
        return vbox;

    }

    /**
     * Compose the attendees area as a labeled, scrolling text area.  The label
     * and the text area are in a left-aligned vbox.  The vbox is in a hbox
     * with horizontal spacing on each side
     */
    protected Box composeAttendees() {
        Box hbox = Box.createHorizontalBox();
        Box vbox = Box.createVerticalBox();

        JLabel label = new JLabel("Attendees:");
        label.setForeground(Color.black);
```

```
        label.setAlignmentX(Box.LEFT_ALIGNMENT);
        attendeesTextArea = new JTextArea(4, 40);
        JScrollPane scrollPane = new JScrollPane(attendeesTextArea);
        scrollPane.setAlignmentX(Box.LEFT_ALIGNMENT);

        vbox.add(label);
        vbox.add(scrollPane);

        hbox.add(Box.createHorizontalStrut(15));
        hbox.add(vbox);
        hbox.add(Box.createHorizontalStrut(15));

        return hbox;
    }

    /**
     * Compose the minutes row as a JLabel/JTextField pair.
     */
    protected Box composeMinutes() {

        Box hbox = Box.createHorizontalBox();
        JLabel label = new JLabel("Minutes:  ");
        label.setForeground(Color.black);

        minutesTextField = new JTextField();
        minutesTextField.setMaximumSize(
            new Dimension(maxComponentWidth, (int)(maxComponentHeight *
                minutesTextField.getFont().getSize())));

        hbox.add(Box.createHorizontalStrut(15));
        hbox.add(label);
        hbox.add(minutesTextField);
        hbox.add(Box.createHorizontalStrut(15));
        return hbox;

    }

    /**
     * Compose the buttons row with four JButtons.  The action listeners for
     * Clear and Cancel buttons are straightforward.  The action listener for
     * the ‘List Times’ and ‘Options’ buttons communicate with the Schedule
     * model.  See the descriptions of <a href=
     * "ListMeetingTimesButtonListener.html"> ListMeetingTimesButtonListener
     * </a> and <a href= "MeetingOptionsButtonListener.html">
     * MeetingOptinsButtonListener </a> for explanatory details.
     */
    protected Box composeButtonRow() {

        Box hbox = Box.createHorizontalBox();

        /*
         * Construct the three buttons.
         */
        JButton listTimesButton = new JButton("List Times ...");
        JButton optionsButton = new JButton("Options ...");
        JButton clearButton = new JButton("Clear");
        JButton cancelButton = new JButton("Cancel");

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

        optionsButton.addActionListener(
            new MeetingOptionsButtonListener((Schedule) model, this));

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

        cancelButton.addActionListener(
            new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    hide();
                }
            }
        );

        /*
         * Add them to the hbox and return it.
         */
        hbox.add(listTimesButton);
        hbox.add(Box.createHorizontalStrut(30));
        hbox.add(optionsButton);
        hbox.add(Box.createHorizontalStrut(30));
        hbox.add(clearButton);
        hbox.add(Box.createHorizontalStrut(30));
        hbox.add(cancelButton);
        return hbox;

    }


    /**
     * Clear each of the text fields of this to empty.  Reset the combo boxes
     * to no selection.  NOTE: This method needs to be refined to use default
     * values for clearing, once options and defaults functionality is
     * implemented.  It also needs to be refined to clear the recurring and
     * remind check boxes and associated components.
     */
    protected void clear() {
        super.clear();
        latestStartDateTextField.setText("");
        latestEndDateTextField.setText("");
        latestStartTimeTextField.setText("");
        attendeesTextArea.setText("");
        minutesTextField.setText("");
```

```java
    }

    /** The latest possible (start) date label */
    protected JLabel latestStartDateLabel;

    /** The latest possible (start) date text field */
    protected JTextField latestStartDateTextField;

    /** The latest possible start time text field */
    protected JTextField latestStartTimeTextField;

    /** The latest possible end date label */
    protected JLabel latestEndDateLabel;

    /** The latest possible end text field */
    protected JTextField latestEndDateTextField;

    /** The attendees text area */
    protected JTextArea attendeesTextArea;

    /** The minutes text field */
    protected JTextField minutesTextField;
}
```