```
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;

    }

    /**
```

```
224      * Compose the row containing the earliest end date and duration.
225      */
226     protected Box composeEarliestEndDateRow() {
227
228         Box hbox = Box.createHorizontalBox();
229
230         /*
231          * Construct the labels and text fields.  See internal comments in the
232          * composeTitle method for further explanatory details.
233          */
234         endDateLabel = new JLabel("Earliest End Date:  ");
235         endDateLabel.setForeground(Color.black);
236         endDateLabel.setEnabled(false);
237         endDateLabel.setAlignmentY(Box.BOTTOM_ALIGNMENT);
238
239         endDateTextField = new JTextField(15);
240         endDateTextField.setMaximumSize(
241             new Dimension(maxComponentWidth, (int)(maxComponentHeight *
242                 endDateTextField.getFont().getSize())));
243         endDateTextField.setEnabled(false);
244         endDateTextField.setAlignmentY(Box.BOTTOM_ALIGNMENT);
245
246         JLabel durationLabel = new JLabel("Duration:  ");
247         durationLabel.setForeground(Color.black);
248         durationLabel.setAlignmentY(Box.BOTTOM_ALIGNMENT);
249
250         durationTextField = new JTextField(15);
251         durationTextField.setMaximumSize(
252             new Dimension(maxComponentWidth, (int)(maxComponentHeight *
253                 durationTextField.getFont().getSize())));
254         durationTextField.setAlignmentY(Box.BOTTOM_ALIGNMENT);
255
256         /*
257          * Add them to the hbox and return it.
258          */
259         hbox.add(Box.createHorizontalStrut(15));
260         hbox.add(endDateLabel);
261         hbox.add(endDateTextField);
262         hbox.add(Box.createHorizontalStrut(10));
263         hbox.add(durationLabel);
264         hbox.add(durationTextField);
265         hbox.add(Box.createHorizontalStrut(15));
266         return hbox;
267
268     }
269
270     /**
271      * Compose the row containing the latest end date.
272      */
273     protected Box composeLatestEndDateRow() {
274
275         Box hbox = Box.createHorizontalBox();
276
277         /*
278          * Construct the labels and text fields.  See internal comments in the
279          * composeTitle method for further explanatory details.
280          */
281         latestEndDateLabel = new JLabel("Latest End Date:  ");
282         latestEndDateLabel.setForeground(Color.black);
283         latestEndDateLabel.setEnabled(false);
284
285         latestEndDateTextField = new JTextField(15);
286         latestEndDateTextField.setMaximumSize(
287             new Dimension(maxComponentWidth, (int)(maxComponentHeight *
288                 latestEndDateTextField.getFont().getSize())));
289         latestEndDateTextField.setEnabled(false);
290
291         /*
292          * Force decent-looking layout with a fixed-size horizontal strut.
293          * There's got to be a better way to do this, such as using the width
294          * of the Duration component, but so far the way has eluded me.
295          */
296         int blankSpacing = 248;
297
298         /*
299          * Add them to the hbox and return it.
300          */
301         hbox.add(Box.createHorizontalStrut(15));
302         hbox.add(latestEndDateLabel);
303         hbox.add(latestEndDateTextField);
304         hbox.add(Box.createHorizontalStrut(blankSpacing));
305         return hbox;
306
307     }
308
309     /**
310      * Compose the bottom part consisting of attendees, details, and minutes
311      * fields.
312      */
313     public Box composeBottomPart() {
314         Box vbox = Box.createVerticalBox();
315
316         vbox.add(composeAttendees());
317         vbox.add(Box.createVerticalStrut(15));
318         vbox.add(composeDetails());
319         vbox.add(Box.createVerticalStrut(15));
320         vbox.add(composeMinutes());
321         return vbox;
322
323     }
324
325     /**
326      * Compose the attendees area as a labeled, scrolling text area.  The label
327      * and the text area are in a left-aligned vbox.  The vbox is in a hbox
328      * with horizontal spacing on each side
329      */
330     protected Box composeAttendees() {
331         Box hbox = Box.createHorizontalBox();
332         Box vbox = Box.createVerticalBox();
333
334         JLabel label = new JLabel("Attendees:");
335         label.setForeground(Color.black);
```

```
336         label.setAlignmentX(Box.LEFT_ALIGNMENT);
337         attendeesTextArea = new JTextArea(4, 40);
338         JScrollPane scrollPane = new JScrollPane(attendeesTextArea);
339         scrollPane.setAlignmentX(Box.LEFT_ALIGNMENT);
340
341         vbox.add(label);
342         vbox.add(scrollPane);
343
344         hbox.add(Box.createHorizontalStrut(15));
345         hbox.add(vbox);
346         hbox.add(Box.createHorizontalStrut(15));
347
348         return hbox;
349     }
350
351     /**
352      * Compose the minutes row as a JLabel/JTextField pair.
353      */
354     protected Box composeMinutes() {
355
356         Box hbox = Box.createHorizontalBox();
357         JLabel label = new JLabel("Minutes:  ");
358         label.setForeground(Color.black);
359
360         minutesTextField = new JTextField();
361         minutesTextField.setMaximumSize(
362             new Dimension(maxComponentWidth, (int)(maxComponentHeight *
363                 minutesTextField.getFont().getSize())));
364
365         hbox.add(Box.createHorizontalStrut(15));
366         hbox.add(label);
367         hbox.add(minutesTextField);
368         hbox.add(Box.createHorizontalStrut(15));
369         return hbox;
370
371     }
372
373     /**
374      * Compose the buttons row with four JButtons.  The action listeners for
375      * Clear and Cancel buttons are straightforward.  The action listener for
376      * the ‘List Times’ and ‘Options’ buttons communicate with the Schedule
377      * model.  See the descriptions of <a href=
378      * "ListMeetingTimesButtonListener.html"> ListMeetingTimesButtonListener
379      * </a> and <a href= "MeetingOptionsButtonListener.html">
380      * MeetingOptinsButtonListener </a> for explanatory details.
381      */
382     protected Box composeButtonRow() {
383
384         Box hbox = Box.createHorizontalBox();
385
386         /*
387          * Construct the three buttons.
388          */
389         JButton listTimesButton = new JButton("List Times ...");
390         JButton optionsButton = new JButton("Options ...");
391         JButton clearButton = new JButton("Clear");
392         JButton cancelButton = new JButton("Cancel");
393
394         /*
395          * Attach the appropriate action listeners to each button.
396          */
397         listTimesButton.addActionListener(
398             new ListMeetingTimesButtonListener((Schedule) model, this));
399
400         optionsButton.addActionListener(
401             new MeetingOptionsButtonListener((Schedule) model, this));
402
403         clearButton.addActionListener(
404             new ActionListener() {
405                 public void actionPerformed(ActionEvent e) {
406                     clear();
407                 }
408             }
409         );
410
411         cancelButton.addActionListener(
412             new ActionListener() {
413                 public void actionPerformed(ActionEvent e) {
414                     hide();
415                 }
416             }
417         );
418
419         /*
420          * Add them to the hbox and return it.
421          */
422         hbox.add(listTimesButton);
423         hbox.add(Box.createHorizontalStrut(30));
424         hbox.add(optionsButton);
425         hbox.add(Box.createHorizontalStrut(30));
426         hbox.add(clearButton);
427         hbox.add(Box.createHorizontalStrut(30));
428         hbox.add(cancelButton);
429         return hbox;
430
431     }
432
433
434     /**
435      * Clear each of the text fields of this to empty.  Reset the combo boxes
436      * to no selection.  NOTE: This method needs to be refined to use default
437      * values for clearing, once options and defaults functionality is
438      * implemented.  It also needs to be refined to clear the recurring and
439      * remind check boxes and associated components.
440      */
441     protected void clear() {
442         super.clear();
443         latestStartDateTextField.setText("");
444         latestEndDateTextField.setText("");
445         latestStartTimeTextField.setText("");
446         attendeesTextArea.setText("");
447         minutesTextField.setText("");
```

```
448      }
449
450      /** The latest possible (start) date label */
451      protected JLabel latestStartDateLabel;
452
453      /** The latest possible (start) date text field */
454      protected JTextField latestStartDateTextField;
455
456      /** The latest possible start time text field */
457      protected JTextField latestStartTimeTextField;
458
459      /** The latest possible end date label */
460      protected JLabel latestEndDateLabel;
461
462      /** The latest possible end text field */
463      protected JTextField latestEndDateTextField;
464
465      /** The attendees text area */
466      protected JTextArea attendeesTextArea;
467
468      /** The minutes text field */
469      protected JTextField minutesTextField;
470  }
```