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

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

/****
 *
 * Class RecurringInfoSubDialog is a component of the ScheduleAppointment,
 * ConfirmMeeting, and ScheduleTask Dialogs.  The companion model is
 * RecurringInfo.  RecurringInfoSubDialog displays the row of recurring
 * information in its varying forms, including the monthly details dialog when
 * appropriate.
 *
 * A meeting request dialog has a slightly different version of the recurring
 * info row.  The subclass RecurringInfoMeetingRequestSubDialog displays this
 * information.
 *
 */
public class RecurringInfoSubdialog extends View {

    /**
     * Construct this with a parent view.  Since this is a subview, it need not
     * have its own model, since it only handles view details.  The parent
     * dialogs for appointments, meetings, and tasks communicate with the
     * model.
     */
    public RecurringInfoSubdialog(Screen screen,
            ScheduleAppointmentDialog parentView) {
        super(screen, null);
        this.parentView = parentView;

        maxComponentHeight = 1.9;
        maxComponentWidth = 1200;
    }

    /**
     * Compose this as the default configuration consisting of a recurring
     * checkbox, a interval combo box, and the weekly interval details.
     */
    public Component compose() {

        Box hbox = Box.createHorizontalBox();

        JLabel recurringLabel = new JLabel("Recurring?  ");
        recurringLabel.setForeground(Color.black);
        recurringCheckBox = new JCheckBox();
        recurringCheckBox.addActionListener(
            new RecurringCheckBoxListener(this, parentView));

        intervalLabel = new JLabel("Interval:  ");
        intervalLabel.setForeground(Color.black);
        intervalLabel.setEnabled(false);
        String[] selections = {"weekly", "biweekly", "monthly", "yearly"};
        intervalComboBox = new JComboBox(selections);
        intervalComboBox.setEnabled(false);
        intervalComboBox.setMaximumSize(
            new Dimension(maxComponentWidth, (int)(maxComponentHeight *
                intervalComboBox.getFont().getSize())));

        hbox.add(Box.createHorizontalStrut(15));
        hbox.add(recurringLabel);
        hbox.add(recurringCheckBox);
        hbox.add(Box.createHorizontalStrut(10));
        hbox.add(intervalLabel);
        hbox.add(intervalComboBox);
        hbox.add(Box.createHorizontalStrut(15));
        hbox.add(composeWeeklyCheckBoxes());
        hbox.add(Box.createHorizontalStrut(15));

        return hbox;

    }

    /**
     * Compose the seven day-of-the-week checkboxes.
     */
    protected Box composeWeeklyCheckBoxes() {
        Box hbox = Box.createHorizontalBox();

        hbox.add(composeDayCheckBox(sunCheckBox = new JCheckBox(),
            sunLabel = new JLabel("S")));
        sunLabel.setForeground(Color.black);
        hbox.add(composeDayCheckBox(monCheckBox = new JCheckBox(),
            monLabel = new JLabel("M")));
        monLabel.setForeground(Color.black);
        hbox.add(composeDayCheckBox(tueCheckBox = new JCheckBox(),
            tueLabel = new JLabel("T")));
        tueLabel.setForeground(Color.black);
        hbox.add(composeDayCheckBox(wedCheckBox = new JCheckBox(),
            wedLabel = new JLabel("W")));
        wedLabel.setForeground(Color.black);
        hbox.add(composeDayCheckBox(thuCheckBox = new JCheckBox(),
            thuLabel = new JLabel("T")));
        thuLabel.setForeground(Color.black);
        hbox.add(composeDayCheckBox(friCheckBox = new JCheckBox(),
            friLabel = new JLabel("F")));
        friLabel.setForeground(Color.black);
        hbox.add(composeDayCheckBox(satCheckBox = new JCheckBox(),
            satLabel = new JLabel("S")));
        satLabel.setForeground(Color.black);

        return hbox;
    }

    /**
     * Compose one of the weekly check boxes as a vbox with a label on top and
     * check box below.  I couldn't find any way to have JFC put its built-in
```

```java
     * check box label on top instead of to the right.  Whatever.
     */
    Box composeDayCheckBox(JCheckBox cbox, JLabel label) {
        Box vbox = Box.createVerticalBox();
        label.setAlignmentX(Box.CENTER_ALIGNMENT);
        label.setEnabled(false);
        cbox.setAlignmentX(Box.CENTER_ALIGNMENT);
        cbox.setEnabled(false);
        vbox.add(label);
        vbox.add(cbox);

        return vbox;
    }

    public void update(Observable o, Object arg) {
        // to be refined
    }


    /** The parent apppointment dialog view. */
    ScheduleAppointmentDialog parentView;

    /** The checkbox that indicates whether the scheculed item recurs. */
    protected JCheckBox recurringCheckBox;

    /** The interval combobox for weekly, biweekly, monthly, or yearly. */
    protected JComboBox intervalComboBox;

    /** The label for the interval combobox.  It's a data field since it needs
        to be set enabled or disabled dynamically */
    protected JLabel intervalLabel;

    /** The Sunday checkbox. */
    protected JCheckBox sunCheckBox;

    /** The Sunday label. */
    protected JLabel sunLabel;

    /** The Monday checkbox. */
    protected JCheckBox monCheckBox;

    /** The Monday label. */
    protected JLabel monLabel;

    /** The Tuesday checkbox. */
    protected JCheckBox tueCheckBox;

    /** The Tuesday label. */
    protected JLabel tueLabel;

    /** The Wednesday checkbox. */
    protected JCheckBox wedCheckBox;

    /** The Wednesday label. */
    protected JLabel wedLabel;

    /** The Thursday checkbox. */
    protected JCheckBox thuCheckBox;

    /** The Thursday label. */
    protected JLabel thuLabel;

    /** The Friday checkbox. */
    protected JCheckBox friCheckBox;

    /** The Friday label. */
    protected JLabel friLabel;

    /** The Saturday checkbox. */
    protected JCheckBox satCheckBox;

    /** The Saturday label. */
    protected JLabel satLabel;

    /** The max height of a text field or combobox; this is necessary since
        these components strech when the outer frame is resized, and look very
        funky when they do */
    protected final double maxComponentHeight;

    /** The max width of any component; this is only necessary because the max
        height cannot be set separately, so we must pick some max width. */
    protected final int maxComponentWidth;

}
```