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

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

/****
 *
 * Class MonthlyAgendaDisplay is the companion view of a MonthlyAgenda model.
 * The fixed layout of the display is a three-part vertical box.  The box
 * contains a date banner, a row of header labels for the days of the week, and
 * the seven-column grid for the days of the month.  The size and layout of the
 * days grid is computed dynamically by the update method, based on the data
 * from the model.
 *
 * @author Gene Fisher (gfisher@calpoly.edu)
 * @version 6feb04
 *
 */
public class MonthlyAgendaDisplay extends CalendarToolWindow {

    /**
     * Construct this by constructing subpanels for the three parts of the
     * display.  Also construct an array of 31 day displays.  This array
     * provides direct access to the individual day displays by date number,
     * which is handy for referencing the companion day models directly.
     *
     * Compute the default size of the days grid to be 5 rows by 7 columns of a
     * default-size day display.  If there are 4 or 6 rows, then the default
     * rows are taller or shorter than they are wide.  This formating is per
     * the requirements.
     *
     * Initialize the displayedOnce flag to false.  The display is only set to
     * the default size the first time it is displayed.  After that, the
     * display retains its size, including any resizing done by the user.
     */
    public MonthlyAgendaDisplay(Screen s, MonthlyAgenda monthlyAgenda,
            CalendarToolUI calToolUI) {
        super(s, monthlyAgenda, calToolUI);
        days = new SmallDayViewDisplay[31];
        dateBanner = new JPanel(new GridLayout(1, 1));
        daysOfWeek = new JPanel(new GridLayout(1, 7));
        dayGrid = new JPanel(new GridLayout(0, 7));
        dayGrid.setBackground(Color.white);
        defaultSize = new Dimension(
            7 * defaultCellWidth, 5 * defaultCellHeight);
        displayedOnce = false;
    }

    /**
     * Compose this as a vertical box, consisting of a date-banner row, a
     * days-of-the-week labels row, and an empty days grid.  The grid will be
     * populated by update.
     */
    public Component compose() {

        /*
         * Make a new window for this.
         */
        window = new mvp.Window();

        /*
         * Make an outer box.
         */
        vbox = new JPanel();
        vbox.setLayout(new BoxLayout(vbox, BoxLayout.Y_AXIS));
        vbox.setBorder(BorderFactory.createLineBorder(Color.black));

        /*
         * Compose the top two rows.
         */
        JPanel bannerBox = composeDateBanner();
        JPanel labelBox = composeDaysOfWeek();

        /*
         * Add the date banner and days-of-week labels to the outer vbox.
         */
        vbox.add(bannerBox);
        vbox.add(labelBox);

        /*
         * Add the empty day grid to the vbox.  It will be populated by update.
         */
        vbox.add(dayGrid);

        /*
         * Add the vbox to the window and we're outta here.
         */
        window.add(vbox);
        window.setTitle("Monthly Agenda");
        return window;
    }

    /**
     * Compose the date banner.  For now it's a dummy label.  In the full
     * implementation, it will contain prev,next,today buttons and the
     * current month/year.
     */
    protected JPanel composeDateBanner() {
        JLabel bannerLabel = new JLabel(((MonthlyAgenda)model).
            getFullMonthName());
        JPanel bannerBox = new JPanel();

        bannerBox.setLayout(new BoxLayout(bannerBox, BoxLayout.Y_AXIS));
        bannerLabel.setForeground(Color.black);
        bannerLabel.setFont(bannerLabel.getFont().deriveFont(Font.BOLD));
        bannerLabel.setHorizontalAlignment(SwingConstants.CENTER);
```

```java
        dateBanner.add(bannerLabel);
//      dateBanner.setOpaque(false);
        dateBanner.setMaximumSize(new Dimension(
            2000, 2 * bannerLabel.getFont().getSize()));

        bannerBox.add(Box.createVerticalStrut(4));
        bannerBox.add(dateBanner);
        bannerBox.add(Box.createVerticalStrut(4));
        bannerBox.setBorder(BorderFactory.createLineBorder(Color.black));
//      bannerBox.setBackground(Color.white);

        return bannerBox;
    }

    /*
     * Compose the days-of-the-week labeling row.  It's a 1x7 grid of labels,
     * so they'll align properly with the columns of the day grid
     */
    JPanel composeDaysOfWeek() {
        JLabel dayLabel = new JLabel("");
        JPanel labelBox = new JPanel();

        labelBox.setLayout(new BoxLayout(labelBox, BoxLayout.Y_AXIS));
        for (int dayNumber = 0; dayNumber < 7; dayNumber++) {
            dayLabel = new JLabel(
                DayName.values()[dayNumber].toString().substring(0,3));
            dayLabel.setHorizontalAlignment(SwingConstants.CENTER);
            dayLabel.setForeground(Color.black);
            dayLabel.setFont(dayLabel.getFont().deriveFont(Font.BOLD));
            daysOfWeek.add(dayLabel);
        }
//      daysOfWeek.setOpaque(false);
        daysOfWeek.setMaximumSize(new Dimension(
            2000, 2 * dayLabel.getFont().getSize()));

        labelBox.add(Box.createVerticalStrut(4));
        labelBox.add(daysOfWeek);
        labelBox.add(Box.createVerticalStrut(4));
        labelBox.setBorder(BorderFactory.createLineBorder(Color.black));
//      labelBox.setBackground(Color.white);

        return labelBox;
    }

    /**
     * Display the model data in the appropriate daily positions.  The data are
     * produced by firstDay and nextDay iterator methods.  The display is a
     * 7-column grid, with 4, 5, or 6 rows, depending on the configuration of
     * the month.
     *
     * In the current implementation, the display is fully redrawn at each call
     * to update, with no display efficiencies implemented.  Possible display
     * efficiencies that might be implemented include the following. (1) If the
     * model data have not changed at all, no updating is performed.  This is
     * the presumably rare case where the user has executed a 'Goto Date'
     * command for the current month.  (2) If the number of weeks in the new
     * model month is the same as the current model month, the row boxes are
     * not reallocated.
     */
    public void update(Observable o, Object arg) {
        int row = 0;            // Week row number
        int dayPosition;        // Ordinal position 0-41 of the current day
        int i;                  // Loop index
        SmallDayViewDisplay     // Loop var for each day's display
            dayViewDisplay;
        int numberOfWeeks =     // Number of weeks === number or rows
            ((MonthlyAgenda)model).getNumberOfWeeks();
        Dimension curSize =     // Current x/y size of grid
            vbox.getSize();
        Dimension cellDimension // Size of one day cell
            = new Dimension(
                (int) (curSize.getWidth() / 7),
                    (int) (curSize.getHeight() / numberOfWeeks));

        /*
         * Clear everything out and set the number of rows to the number of
         * weeks in the model month.
         */
        Arrays.fill(days, null);
        dayGrid.removeAll();
        GridLayout layout = (GridLayout) dayGrid.getLayout();
        layout.setRows(numberOfWeeks);

        /*
         * Put empty grey boxes up to the first day position.
         */
        SmallDayView dayView = ((MonthlyAgenda)model).getFirstDay();
        dayPosition = dayView.getDay().ordinal();
        for (i = 0; i < dayPosition; i++)
            dayGrid.add(greyDay());

        /*
         * Populate the individual day displays with model data.
         */
        for (; dayView != null;
                dayView = ((MonthlyAgenda)model).getNextDay(), dayPosition++) {
            dayViewDisplay = new SmallDayViewDisplay(
                screen, dayView, (MonthlyAgenda)model, cellDimension);
            days[dayView.getDate()] = dayViewDisplay;
            dayGrid.add(dayViewDisplay.getWidget());
        }

        /*
         * Put empty grey boxes up to the last day position in the last row.
         */
        for (i = dayPosition; i % 7 != 0; i++) {
            dayGrid.add(greyDay());
        }

        /*
         * Set grid to the default size if this is the first time it's being
         * displayed.  Otherwise, leave its size as it was.  In either case,
```

```java
         * pack the grid in order to "burn in" the layout.
         */
        window.getContentPane().setBackground(Color.blue);
        if (! displayedOnce) {
            dayGrid.setPreferredSize(defaultSize);
            displayedOnce = true;
            window.pack();
        }
    }

    /**
     * Build an empty grey-background, black-border day display.  A fresh one
     * of these needs to be allocated for each use since JFC doesn't play
     * reuses of a components in containers.
     */
    protected JPanel greyDay() {
        JPanel panel = new JPanel();
        panel.setBackground(Color.lightGray);
        panel.setBorder(BorderFactory.createLineBorder(Color.black));

        return panel;
    }

    /** Array of day displays for convenient access by date number.  This array
        contains references to the same day-display objects that are laid out
        in the day grid. */
    protected SmallDayViewDisplay days[];

    /** Outermost box of the laid-out display. */
    protected JPanel vbox;

    /** The date banner at the top of the display. */
    protected JPanel dateBanner;

    /** The days-of-the week labeling row. */
    protected JPanel daysOfWeek;

    /** The day grid. */
    protected JPanel dayGrid;

    /** Number or weeks (hence display rows) in the current display. */
    protected int numberOfWeeks;

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

    /** Initial default size of the day grid. */
    protected Dimension defaultSize;

    /** Default constant for the height of one day display cell. */
    protected final int defaultCellHeight = 75;

    /** Default constant for the width of one day display cell. */
    protected final int defaultCellWidth = 75;

}
```