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

import caltool.caldb.*;
import caltool.schedule.DayName;
import mvp.*;
import java.util.*;

/****
 *
 * A MonthlyAgenda contains a full month name, the day of the week for its
 * first day, and the number of days.  Scheduled item data are contained in a
 * small daily view for each day of the month, organized in a fashion typical
 * in paper calendars.
 *                                                                          <p>
 * The primary access interface is through the getFirstDay and getNextDay
 * iterators.  These methods deliver each day of the month in turn, as a small
 * day view object.
 *                                                                          <p>
 * The current implementation is a stub consisting of a sample 30-day month
 * that starts on Tuesday.  The actual implementation will consult the
 * CalendarDB to obtain real monthly data.
 *
 * @author Gene Fisher (gfisher@calpoly.edu)
 * @version 4feb05
 *
 */

public class MonthlyAgenda extends Model {

    /**
     * Construct this with the given CalendarDB.  Call update to get the data
     * values for the initially current month.
     */
    public MonthlyAgenda(CalendarDB calDB) {
        this.calDB = calDB;
        update(null, null);
    }

    /**
     * Return the full month name as a single string.
     */
    public String getFullMonthName() {
        return fullMonthName.toString();
    }

    /**
     * Return the first day of the month as a SmallDayView, q.v.
     */
    public SmallDayView getFirstDay() {
        return new SmallDayView(currentDate, DayName.values()[currentDay], null);
    }

    /**
     * Return the second and subsequent days of the month.  Return null when
     * all days have been produced.
     */
    public SmallDayView getNextDay() {
        if (currentDate < numberOfDays) {
            return new SmallDayView(++currentDate,
                DayName.values()[++currentDay % 7], null);
        }
        else {
            currentDate = 1;
            currentDay = firstDay.ordinal();
            return null;
        }
    }

    /**
     * Return the number of weeks in the month.
     */
    public int getNumberOfWeeks() {
        return (int) Math.ceil(
            ((double)(numberOfDays + firstDay.ordinal())) / 7.0);
    }

    /**
     * Build a complete Date out of the given date number and call the
     * CalendarDB to select that date.  This is fixed for initial testing.
     */
    public void selectDate(int date) {
        System.out.println("In MonthlyAgenda.selectDate(" + date + ")");
    }

    /**
     * Update this' data based on the current selection in the current
     * calendar.  For initial testing purposes, the fixed month of September
     * 1998 is created, which starts on Tuesday and has 30 days.  In the
     * refined implementation, the calendar db will be consulted to obtain the
     * actual information for the currently selected month.
     */
    public void update(Observable o, Object arg) {

        /*
         * Define fixed data for initial testing purposes.
         */
        fullMonthName = new FullMonthName("September", 1998);
        firstDay = DayName.Tuesday;
        numberOfDays = 30;

        /*
         * Initialize generator state variables.
         */
        currentDate = 1;
        currentDay = firstDay.ordinal();

    }


    /*-*
     * Derived data.
     */

    /** Full name, consisting of month name and year. */
    protected FullMonthName fullMonthName;

    /** First day of the month */
    protected DayName firstDay;

    /** Number of days in the month */
    protected int numberOfDays;

    /** Array of small day views, each containing zero or more brief item
     * descriptors for the items (if any) scheduled on that day.
    protected SmallDayView[] smallDayViews;


    /*-*
     * Iterator state variables.
     */

    /** Iterator state variable containing the date number. */
    protected int currentDate;

    /** Iterator state variable containing the ordinal day position in a 6x7
        grid.  */
    protected int currentDay;

    /** The caldb for getting current data */
    CalendarDB calDB;

}
```