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

import caltool.schedule.*;
import caltool.caldb.*;
import mvp.*;

/****
 *
 * Class View is the top-level model class in the view package.  It provides
 * methods to view the calendar at the five structural levels of a calendar:
 * item, day, week, month, and year.  There are also methods to go to the
 * previous and next views at any level, as well as an method to go to a
 * specific date.  Methods are provided to view lists of scheduled items in a
 * variety of ways.  Methods are provided to view other users' calendars and to
 * view a list of active viewing windows.  View filtering are capabilities are
 * defined in the Filter submodel.
 *
 */
public class View extends Model {

    public View(mvp.View view, CalendarDB caldb) {
        super(view);

        this.caldb = caldb;

        dailyAgenda = new DailyAgenda(caldb);
        weeklyAgendaTable = new WeeklyAgendaTable(caldb);
        weeklyAgendaList = new WeeklyAgendaList(caldb);
        monthlyAgenda = new MonthlyAgenda(caldb);
        lists  = new Lists(caldb);

        appointmentsHidden = false;
    }

    /*-*
     * Derived methods
     */

    /**
     * Produce the currently selected scheduled item.
     */
    public ScheduledItem viewItem() {
        return caldb.getCurrentCalendar().getSelectedItem();
    }

    /**
     * Produce the daily agenda for the currently selected date, or for today's
     * date if no other date is currently selected.
     */
    public DailyAgenda viewDay() {
        dailyAgenda.update(null, null);
        return null;
    }

    /**
     * Produce the monthly agenda for the currently selected date, or for
     * today's date if no other date is currently selected.
     */
    public MonthlyAgenda viewMonth() {
        monthlyAgenda.update(null, null);
        return monthlyAgenda;
    }

    /**
     * Return the lists model class that has the methods to compute the differe
     * forms of lists.
     */
    public Lists getLists() {
        return lists;
    }

    /**
     * Select the date in the current calendar.  The most typical reason for
     * date selection is as the argument to a view command.
     */
    public void selectDate(Date date) {
        calDB.setSelectedDate(date);
    }

    /**
     * Select the date given a single date number.  Figure out the complete
     * date based on the currently active view window.
     */
    public void selectDate(int date) {
        System.out.println(date);
    }


    /**
     * Toggle the show/hide state for appointments.
     */
    public void toggleShowHideAppointments() {
        appointmentsHidden = ! appointmentsHidden;
    }

    /**
     * Return true if appointments are hidden, false if not.
     */
    public boolean areAppointmentsHidden() {
        return appointmentsHidden;
    }

    /**
     * Toggle the show/hide state for meetings.
     */
    public void toggleShowHideMeetings() {
        meetingsHidden = ! meetingsHidden;
    }

    /**
     * Return true if meetings are hidden, false if not.
     */
    public boolean areMeetingsHidden() {
        return meetingsHidden;
    }

    /**
     * Toggle the show/hide state for tasks.
     */
    public void toggleShowHideTasks() {
        tasksHidden = ! tasksHidden;
    }

    /**
     * Return true if tasks are hidden, false if not.
     */
    public boolean areTasksHidden() {
        return tasksHidden;
    }

    /**
     * Toggle the show/hide state for events.
     */
    public void toggleShowHideEvents() {
        eventsHidden = ! eventsHidden;
    }

    /**
     * Return true if events are hidden, false if not.
     */
    public boolean areEventsHidden() {
        return eventsHidden;
    }

    /** Calendar database in which viewed items are stored */
    protected CalendarDB calDB;

    /** The current instance of DailyAgenda that computes the view for the
     * user-selected day.  This must be refined into a collection of some form
     * to support multi-window mode. */
    protected DailyAgenda dailyAgenda;

    /** The current instance of WeeklyAgendaTable that computes the view for
     * the user-selected week.  This must be refined into an collection of some
     * for to support multi-window mode. */
    protected WeeklyAgendaTable weeklyAgendaTable;

    /** The current instance of WeeklyAgendaList that computes the view for the
     * user-selected week.  This must be refined into a collection of some form
     * to support multi-window mode. */
    protected WeeklyAgendaList weeklyAgendaList;

    /** The current instance of MonthlyAgenda that computes the view for the
     * user-selected month.  This must be refined into a collection of some
     * form to support multi-window mode. */
    protected MonthlyAgenda monthlyAgenda;

    /** The lists submodel */
    protected Lists lists;

    /** The filter submodel */
    protected Filter filter;

    /** The windows submodel */
    protected Windows windows;

    /** The shown/hidden state of appointments */
    boolean appointmentsHidden;

    /** The shown/hidden state of meetings */
    boolean meetingsHidden;

    /** The shown/hidden state of tasks */
    boolean tasksHidden;

    /** The shown/hidden state of events */
    boolean eventsHidden;

    /** The CalendarDB used to get the currently selected date and to pass on
     * to subviews */
    CalendarDB caldb;
}
```