MonthlyAgenda.java

```
 1  package caltool.model.view;
 2
 3  import caltool.model.caldb.*;
 4  import caltool.model.schedule.DayName;
 5  import mvp.*;
 6  import java.util.*;
 7
 8  /****
 9   *
10   * A MonthlyAgenda contains a full month name, the day of the week for its
11   * first day, and the number of days.  Scheduled item data are contained in a
12   * small daily view for each day of the month, organized in a fashion typical
13   * in paper calendars.
14   *                                                                      <p>
15   * The primary access interface is through the getFirstDay and getNextDay
16   * iterators.  These methods deliver each day of the month in turn, as a small
17   * day view object.
18   *                                                                      <p>
19   * The current implementation is a stub consisting of a sample 30-day month
20   * that starts on Tuesday.  The actual implementation will consult the
21   * CalendarDB to obtain real monthly data.
22   *
23   * @author Gene Fisher (gfisher@calpoly.edu)
24   * @version 13apr15
25   *
26   */
27
28  public class MonthlyAgenda extends Model {
29
30      /**
31       * Construct this with the given CalendarDB.  Call update to get the data
32       * values for the initially current month.
33       */
34      public MonthlyAgenda(CalendarDB calDB) {
35          this.calDB = calDB;
36          update(null, null);
37      }
38
39      /**
40       * Return the full month name as a single string.
41       */
42      public String getFullMonthName() {
43          return fullMonthName.toString();
44      }
45
46      /**
47       * Return the first day of the month as a SmallDayView, q.v.
48       */
49      public SmallDayView getFirstDay() {
50          return new SmallDayView(currentDate, DayName.values()[currentDay], null);
51      }
52
53      /**
54       * Return the second and subsequent days of the month.  Return null when
55       * all days have been produced.
56       */
57      public SmallDayView getNextDay() {
58          if (currentDate < numberOfDays) {
59              return new SmallDayView(++currentDate,
60                  DayName.values()[++currentDay % 7], null);
61          }
62          else {
63              currentDate = 1;
64              currentDay = firstDay.ordinal();
65              return null;
66          }
67      }
68
69      /**
70       * Return the number of weeks in the month.
71       */
72      public int getNumberOfWeeks() {
73          return (int) Math.ceil(
74              ((double)(numberOfDays + firstDay.ordinal())) / 7.0);
75      }
76
77      /**
78       * Build a complete Date out of the given date number and call the
79       * CalendarDB to select that date.  This is fixed for initial testing.
80       */
81      public void selectDate(int date) {
82          System.out.println("In MonthlyAgenda.selectDate(" + date + ")");
83      }
84
85      /**
86       * Update this' data based on the current selection in the current
87       * calendar.  For initial testing purposes, the fixed month of September
88       * 2015 is created, which starts on Tuesday and has 30 days.  In the
89       * refined implementation, the calendar db will be consulted to obtain the
90       * actual information for the currently selected month.
91       */
92      public void update(Observable o, Object arg) {
93
94          /*
95           * Define fixed data for initial testing purposes.
96           */
97          fullMonthName = new FullMonthName("September", 2015);
98          firstDay = DayName.Tuesday;
99          numberOfDays = 30;
100
101         /*
102          * Initialize generator state variables.
103          */
104         currentDate = 1;
105         currentDay = firstDay.ordinal();
106
107     }
108
109
110     /*-*
111      * Derived data.
112      */
```

```
113
114     /** Full name, consisting of month name and year. */
115     protected FullMonthName fullMonthName;
116
117     /** First day of the month */
118     protected DayName firstDay;
119
120     /** Number of days in the month */
121     protected int numberOfDays;
122
123     /** Array of small day views, each containing zero or more brief item
124      * descriptors for the items (if any) scheduled on that day.
125     protected SmallDayView[] smallDayViews;
126
127
128     /*-*
129      * Iterator state variables.
130      */
131
132     /** Iterator state variable containing the date number. */
133     protected int currentDate;
134
135     /** Iterator state variable containing the ordinal day position in a 6x7
136         grid.  */
137     protected int currentDay;
138
139     /** The caldb for getting current data */
140     CalendarDB calDB;
141
142 }
```