

```

Loading vc-cvs...
1 package caltool.caldb;
2
3 import caltool.schedule.*;
4 import java.io.*;
5
6 /**
7  *
8  * Class ItemKey is the Map key used to store items in a user calendar. It has
9  * five fields, some or all of which are used as the key for an item, per the
10 * following table:
11 *
12 * Field      Used For: Appt  Meeting  Task  Event
13 * =====
14 * date       yes  yes  yes  yes
15 * time       yes  yes  yes  no
16 * duration   yes  yes  no  no
17 * title      yes  yes  yes  yes
18 * priority   no   no   yes  no
19 *
20 *
21 * ItemKey is suitable for use in either a HashMap or a TreeMap, since it
22 * specializes Object.hashCode (for the HashMap) and implements Comparable (for
23 * the TreeMap).
24 *
25 */
26 public class ItemKey implements Comparable, Serializable {
27
28     /**
29     * Construct this with the given field values.
30     */
31     public ItemKey(Date date, Time time, Duration duration, String title,
32                    int priority) {
33         this.date = (date != null) ? date : new Date();
34         this.time = (time != null) ? time : new Time();
35         this.duration = (duration != null) ? duration : new Duration();
36         this.title = (title != null) ? title : "";
37         this.priority = priority;
38     }
39
40     /**
41     * Define equality for this as componentwise equality.
42     */
43     public boolean equals(Object obj) {
44         ItemKey otherKey = (ItemKey) obj;
45
46         return
47             date.equals(otherKey.date) &&
48             time.equals(otherKey.time) &&
49             duration.equals(otherKey.duration) &&
50             title.equals(otherKey.title) &&
51             priority == otherKey.priority;
52     }
53
54     /**
55     * Define this' hash code as the sum of component hash codes.
56
57     */
58     public int hashCode() {
59         return date.hashCode() + time.hashCode() + duration.hashCode() +
60             title.hashCode() + new Integer(priority).hashCode();
61     }
62
63     /**
64     * Define compareTo based on the total ordering of items defined in the
65     * spec. Viz., all items in Day[i] are less than all items in Day[j], when
66     * the the calendar date of Day[i] precedes Day[j]. For a given day, the
67     * order is:
68     *
69     * <li> events, ordered alphabetically by title </li>
70     * <li> tasks, ordered by start time (primary), priority (secondary),
71     * and title (tertiary) </li>
72     * <li> appointments and meetings, ordered by start time (primary),
73     * duration (secondary), and title (tertiary) </li>
74     *
75     */
76     public int compareTo(Object o) {
77         ItemKey otherKey = (ItemKey) o;
78
79         /*
80         * Return immediately if the calendar day is different.
81         */
82         if (date.compareTo(otherKey.date) < 0)
83             return -1;
84         if (date.compareTo(otherKey.date) > 0)
85             return 1;
86
87         /*
88         * Return an event as less than any other type of item. Compare two
89         * same-day events by title.
90         */
91         if (this.isEventKey()) {
92             if (! otherKey.isEventKey()) {
93                 return -1;
94             }
95             else {
96                 return title.compareTo(otherKey.title);
97             }
98         }
99
100        /*
101        * Return a task as greater than any event and less than any
102        * appointment or meeting. Compare two same-day tasks by start time,
103        * priority, and title.
104        */
105        if (this.isTaskKey()) {
106            if (otherKey.isEventKey()) {
107                return 1;
108            }
109            else if (otherKey.isAppointmentKey()) {
110                return -1;
111            }

```

```

112     }
113     else {
114         return compareTaskKeys(otherKey);
115     }
116
117     /*
118     * Return an appointment/meeting as greater than any other type of
119     * item. Compare two same-day appointments/meetings by start time,
120     * duration, and title.
121     */
122     if (! otherKey.isAppointmentKey()) {
123         return 1;
124     }
125     else {
126         return compareAppointmentKeys(otherKey);
127     }
128
129 }
130
131 /**
132  * Convert this to a string representation.
133  */
134 public String toString() {
135     String dateString = (date != null) ? date.toString() : "";
136     String timeString = (time != null) ? time.toString() : "";
137     String durationString = (duration != null) ? duration.toString() : "";
138     String titleString = (title != null) ? title : "";
139     return "[".concat(dateString).concat(",").concat(timeString).
140         concat(",").concat(durationString).concat(",").concat(titleString).
141         concat("]");
142 }
143
144 /**
145  * Compare this and the given appointment/meeting key.
146  */
147 protected int compareAppointmentKeys(ItemKey apptKey) {
148     /*
149     * Implementation forthcoming.
150     */
151     return 0;
152 }
153
154 /**
155  * Compare this and the given task key.
156  */
157 protected int compareTaskKeys(ItemKey taskKey) {
158     /*
159     * Implementation forthcoming.
160     */
161     return 0;
162 }
163
164 /**
165  * Return true if this is an event key. Per the field definition table, an
166  * event key is uniquely determined by a non-null time field and null
167  * priority field.
168
169     */
170     protected boolean isEventKey() {
171         return
172             time.isEmpty() && (priority == 0);
173     }
174
175     /**
176     * Return true if this is an appointment/meeting key. Per the field
177     * definition table, an appointment key is uniquely determined by a
178     * non-null duration field and null priority field.
179     */
180     protected boolean isAppointmentKey() {
181         return
182             !time.isEmpty() && (priority == 0);
183     }
184
185     /**
186     * Return true if this is a task key. Per the field definition table, a
187     * task key is uniquely determined by a positive priority field.
188     */
189     protected boolean isTaskKey() {
190         return
191             priority > 0;
192     }
193
194     /** The date field used in all keys */
195     protected Date date;
196
197     /** The time field used in all keys except for events */
198     protected Time time;
199
200     /** The duration field used in appointment and meeting keys */
201     protected Duration duration;
202
203     /** The title field used in all keys */
204     protected String title;
205
206     /** The priority field in task keys only */
207     int priority;
208 }

```