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

import caltool.schedule.*;
import java.io.*;

/****
 *
 * Class ItemKey is the Map key used to store items in a user calendar.  It has
 * five fields, some or all of which are used as the key for an item, per the
 * following table:
 *                                                                          <pre>
 *    Field     Used For: Appt  Meeting   Task   Event
 *    ===================================================
 *    date                yes    yes      yes     yes
 *    time                yes    yes      yes     no
 *    duration            yes    yes      no      no
 *    title               yes    yes      yes     yes
 *    priority            no     no       yes     no
 *                                                                         </pre>
 *
 * ItemKey is suitable for use in either a HashMap or a TreeMap, since it
 * specializes Object.hashCode (for the HashMap) and implements Comparable (for
 * the TreeMap).
 *
 */
public class ItemKey implements Comparable, Serializable {

    /**
     * Construct this with the given field values.
     */
    public ItemKey(Date date, Time time, Duration duration, String title,
            int priority) {
        this.date = (date != null) ? date : new Date();
        this.time = (time != null) ? time : new Time();
        this.duration = (duration != null) ? duration : new Duration();
        this.title = (title != null) ? title : "";
        this.priority = priority;
    }

    /**
     * Define equality for this as componentwise equality.
     */
    public boolean equals(Object obj) {
        ItemKey otherKey = (ItemKey) obj;

        return
            date.equals(otherKey.date) &&
            time.equals(otherKey.time) &&
            duration.equals(otherKey.duration) &&
            title.equals(otherKey.title) &&
            priority == otherKey.priority;
    }

    /**
     * Define this’ hash code as the sum of component hash codes.
     */
    public int hashCode() {
        return date.hashCode() + time.hashCode() + duration.hashCode() +
            title.hashCode() + new Integer(priority).hashCode();
    }

    /**
     * Define compareTo based on the total ordering of items defined in the
     * spec.  Viz., all items in Day[i] are less than all items in Day[j], when
     * the the calendar date of Day[i] precedes Day[j].  For a given day, the
     * order is:
     *                                                                      <ul>
     *    <li> events, ordered alphabetically by title </li>
     *    <li> tasks, ordered by start time (primary), priority (secondary),
     *           and title (tertiary) </li>
     *    <li> appointments and meetings, ordered by start time (primary),
     *           duration (secondary), and title (tertiary) </li>
     *                                                                     </ul>
     */
    public int compareTo(Object o) {
        ItemKey otherKey = (ItemKey) o;

        /*
         * Return immediately if the calendar day is different.
         */
        if (date.compareTo(otherKey.date) < 0)
            return -1;
        if (date.compareTo(otherKey.date) > 0)
            return 1;

        /*
         * Return an event as less than any other type of item.  Compare two
         * same-day events by title.
         */
        if (this.isEventKey()) {
            if (! otherKey.isEventKey()) {
                return -1;
            }
            else {
                return title.compareTo(otherKey.title);
            }
        }

        /*
         * Return a task as greater than any event and less than any
         * appointment or meeting.  Compare two same-day tasks by start time,
         * priority, and title.

         */
        if (this.isTaskKey()) {
            if (otherKey.isEventKey()) {
                return 1;
            }
            else if (otherKey.isAppointmentKey()) {
                return -1;
            }
```

```java
        }
        else {
            return compareTaskKeys(otherKey);
        }

        /*
         * Return an appointment/meeting as greater than any other type of
         * item. Compare two same-day appointments/meetings by start time,
         * duration, and title.
         */
        if (! otherKey.isAppointmentKey()) {
            return 1;
        }
        else {
            return compareAppointmentKeys(otherKey);
        }

    }

    /**
     * Convert this to a string representation.
     */
    public String toString() {
        String dateString = (date != null) ? date.toString() : "";
        String timeString = (time != null) ? time.toString() : "";
        String durationString = (duration != null) ? duration.toString() : "";
        String titleString = (title != null) ? title : "";
        return "{".concat(dateString).concat(",").concat(timeString).
            concat(",").concat(durationString).concat(",").concat(titleString).
                concat("}");
    }

    /**
     * Compare this and the given appointment/meeting key.
     */
    protected int compareAppointmentKeys(ItemKey apptKey) {
        /*
         * Implementation forthcoming.
         */
        return 0;
    }

    /**
     * Compare this and the given task key.
     */
    protected int compareTaskKeys(ItemKey taskKey) {
        /*
         * Implementation forthcoming.
         */
        return 0;
    }

    /**
     * Return true if this is an event key.  Per the field definition table, an
     * event key is uniquely determined by a non-null time field and null
     * priority field.
     */
    protected boolean isEventKey() {
        return
            time.isEmpty() && (priority == 0);
    }

    /**
     * Return true if this is an appointment/meeting key.  Per the field
     * definition table, an appointment key is uniquely determined by a
     * non-null duration field and null priority field.
     */
    boolean isAppointmentKey() {
        return
            !time.isEmpty() && (priority == 0);
    }

    /**
     * Return true if this is a task key.  Per the field definition table, a
     * task key is uniquely determined by a positive priority field.
     */
    protected boolean isTaskKey() {
        return
            priority > 0;
    }


    /** The date field used in all keys */
    protected Date date;

    /** The time field used in all keys except for events */
    protected Time time;

    /** The duration field used in appointment and meeting keys */
    protected Duration duration;

    /** The title field used in all keys */
    protected String title;

    /** The priority field in task keys only */
    int priority;
}
```