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

import caltool.schedule.*;
import caltool.options.*;
import mvp.*;
import java.util.*;

/****
 *
 * The main data components of a user UserCalendar are a collection of
 * scheduled items and calendar-specific settings.  Calendar bookkeeping
 * components are the ID of the user who owns the calendar, the file it’s
 * stored on, the currently selected date, and a flag indicating if the
 * calendar requires saving.
 *                                                                          <p>
 * In the current design, the concrete representation of the scheduled item
 * list is a TreeMap.  UserCalendar provides a getItem method to look up a
 * scheduled item by its unique key.  Based on the specs, the unique key for
 * each type of item is as follows:
 *                                                                        <pre>
 *     Item             Unique Key
 *     ====================================================================
 *     Appointment      {date, start time, duration, title}
 *     Meeting          {date, start time, duration, title}
 *     Task             {date, time, title, priority}
 *     Event            {date, title}
 *                                                                       </pre>
 * UserCalendar also provides an array-valued getItems method to retrieve all
 * of the items that are scheduled in a specified interval of date/time.  This
 * method is used by the caltool viewing methods to access the scheduled items
 * for a given day, week, or month.
 *                                                                          <p>
 * UserCalendar provides general-purpose methods to support the higher-level
 * model classes in the schedule and view packages.  The general-purpose
 * methods of UserCalendar do no input validity checking, assuming it has been
 * performed by the higher-level model methods.
 *
 */
public class UserCalendar extends Model {

    /*-*
     * Public methods.
     */

    /**
     * Construct this by constructing and initializing all components.
     */
    public UserCalendar(String uid) {
        items = new TreeMap();
        settings = null;
        this.uid = uid;
        file = null;
        selectedDate = null;
        requiresSaving = false;
        selectedItem = null;

        /*
         * For initial testing purposes, construct a fixed item to use as the
         * currently selected item.
         */
        selectedItem = new Appointment(
                "Dentist",                      // Title
                new caltool.schedule.Date(      // Date
                    "September 25, 1998"),
                null,                           // End Date
                new Time("8 AM"),               // Time
                new Duration(1, 30),            // Duration
                null,                           // Recurring info
                new Category("personal"),       // Category
                "1342 Sycamore Dr",             // Location
                Security.PublicTitle,           // Security
                Priority.Must,                  // Priority
                new RemindInfo(true,            // Remind info
                    new RemindWhen(1,
                        ReminderTimeUnit.DaysBefore),
                    RemindWhere.OnScreen),
                ""                              // Details
        );
    }

    /**
     * Add the given item to this.items.  Note that this method has no
1     * precondition.  All the validity and no-duplication requirements for the
     * given item are checked at the level of the Schedule model.
     *                                                                    <pre>
     * pre: ;
     *
     * post:
     *       //
     *       // The input item is added to items via items.put, which means
     *       // that item is added if an item of the same key is not already
     *       // there.  This is marked as changed via Observable.setChanged().
     *       //
     *       (items’ == items.put(item.getKey(), item))
     *
     *           &&
     *
     *       this’.hasChanged();
     *                                                                   </pre>
     */
    public void add(ScheduledItem item) {

        /*
         * Put the given item into the items map with its generated unique key.
         */
        items.put(item.getKey(), item);

        /*
         * Indicate that this has changed in case anyone is observing.  The
         * setChanged method is inherited from Model, which in turn inherits
         * them from Observable.
         */
        setChanged();

    }

    /**
     * Delete the given item from this.items.  Note that this method has no
     * precondition.  All the validity and no-duplication requirements for the
     * given item are checked at the level of the Schedule model.
     *                                                                  <pre>
     * pre: ;
     *
     * post:
     *       //
     *       // The input item is added to items via HashMap.put, which means
     *       // that item is added if an item of the same key is not already
     *       // there.  This is marked as changed via Observable.setChanged().
     *       //
     *       (items' == items.remove(item.getKey(), item))
     *
     *           &&
     *
     *       this'.hasChanged();
     *                                                                 </pre>
     */
    public void delete(ScheduledItem item) {

        items.remove(item);

        /*
         * Indicate that this has changed in case anyone is observing.  The
         * setChanged method is inherited from Model, which in turn inherits
         * them from Observable.
         */
        setChanged();
    }

    /**
     * Return the scheduled item of the given unique key.
     *                                                                  <pre>
     * pre: ;
     *
     * post:
     *       //
     *       // If there is an item with the given key in this.items, then the
     *       // return value is that item, otherwise the return is null.
     *       //
     *       (exists (item in items) (item.getKey().equals(key)) &&
     *                               (return == item)
     *           ||
     *
     *       (return == null);
     *                                                                 </pre>
     */
    public ScheduledItem getItem(ItemKey key) {
        return (ScheduledItem) items.get((Object) key);
    }

    /**
     * Return an array of items in the given date range.  The start date must
     * be <= the end date.
     */
    public ScheduledItem[] getItems(caltool.schedule.Date startDate,
            caltool.schedule.Date endDate) {

        /*
         * Implementation forthcoming.
         */

        return null;
    }

    /**
     * Return the previous item in item-key order after the item with the given
     * key.  Return null if the given key is that of the first item
     */
    public ScheduledItem getPrev(ItemKey key) {

        try {
            return (ScheduledItem) items.get(items.headMap(key).lastKey());
        }
        catch (NoSuchElementException e) {
            return null;
        }

    }

    /**
     * Return the next item in item-key order after the item with the given
     * key.  Return null if the given key is that of the last item
     */
    public ScheduledItem getNextItem(ItemKey key) {

        Iterator it = items.tailMap(key).keySet().iterator();
        if (! it.hasNext()) {
            return null;
        }
        it.next();
        if (it.hasNext()) {
            return (ScheduledItem) items.get(it.next());
        }
        else {
            return null;
        }

    }

    /**
     * Return the user id of this calendar.
     */
    String getUid() {
        return uid;
```

```java
    }

    /**
     * Return the file on which this calendar was most recently stored.
     */
    java.io.File getFile() {
        return file;
    }

    /**
     * Set the file on which this calendar is currently stored.
     */
    void setFile(java.io.File file) {
        this.file = file;
    }

    /**
     * Get the calendar-specific settings for this calendar.
     */
    CalendarSpecificSettings getSettings() {
        return settings;
    }

    /**
     * Return the date most recently selected by the user via clicking in some
     * view.
     */
    public caltool.schedule.Date getSelectedDate() {
        return selectedDate;
    }

    /**
     * Set the currently selected date to the given date.
     */
    public void setSelectedDate(caltool.schedule.Date date) {
        selectedDate = date;
    }

    /**
     * Return the item most recently selected by the user via clicking in some
     * view.
     */
    public ScheduledItem getSelectedItem() {
        return selectedItem;
    }

    /**
     * Set the currently selected date to the given date.
     */
    public void setSelectedItem(ScheduledItem item) {
        selectedItem = item;
    }

    /**
     * Convert this to a printable string.  The items are dumped last, since
     * there may be a lot of them.  Note that the settings field is only
     * printed shallow since no methods of this change the contents of
     * settings.
     */
    public String toString() {
        return
            "User id: " + uid + "\n" +
            "File: " + (file == null ? "null" : file.toString()) + "\n" +
            "Selected date: " +
                (selectedDate == null ? "null" : selectedDate.toString()) +
                    "\n" +
            "Requires saving: " + String.valueOf(requiresSaving) + "\n" +
            "Selected item: " +
                (selectedItem == null ? "null" : selectedItem.toString()) +
                    "\n" +
            items.size() + " Items: \n" + items.toString() + "\n" +
            settings + "\n";
    }

    /**
     * Return the number of items in this.items, for testing purposes.
     */
    public int numItems() {
        return items.size();
    }

    /*-*
     * Derived data fields
     */

    /** The collection of all scheduled items for this calendar */
    TreeMap items;

    /** Calendar-specific settings for this calendar */
    CalendarSpecificSettings settings;

    /** Id of user who owns this calendar */
    String uid;

    /** File this calendar is stored on */
    java.io.File file;

    /** Currently selected date, if any */
    caltool.schedule.Date selectedDate;

    /** True if this requires saving */
    boolean requiresSaving;


    /*-*
     * Additional data fields
     */

    /** Currently selected item, if any */
    ScheduledItem selectedItem;

}
```