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

import mvp.*;

/****
 *
 * A Time consists of an hour, minute, and AM or PM indicator.  A time value is
 * expressed using a 12-hour or 24-hour clock style.  The clock style is set as
 * an option by the user.  If the clock style is 24-hour, the AmOrPm indicator
 * is nil.
 *
 */

public class Time extends Model {

    /**
     * Construct an empty time value.
     */
    public Time() {
        hour = 0;
        minute = 0;
        amOrPm = null;
        valid = true;
        empty = true;
    }

    /**
     * Construct a time from the given string. Set the valid field to false if
     * the given string does not parse to a valid time.  Note that the invalid
     * state representation is used instead of throwing an exception because
     * some users may want to delay the processing of invalid dates, and hence
     * may not be interested in handling an exception.
     */
    public Time(String time) {
        /*
         * Constant stubbed implementation.
         */
        hour = 12;
        minute = 0;
        amOrPm = null;

        valid = true;
        empty = false;
    }

    /**
     * Return true if his is an empty time.
     */
    public boolean isEmpty() {
        return empty;
    }

    /**
     * Return the string representation of this.
     */
    public String toString() {
        return Integer.toString(hour).concat(":").
            concat((minute < 10) ? "0" : "").concat(
                Integer.toString(minute)).
                    concat((amOrPm != null) ?
                        " " + amOrPm.toString() : "");
    }

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

        return
            hour == otherTime.hour &&
            minute == otherTime.minute &&
            amOrPm.equals(otherTime.amOrPm);
    }

    /**
     * Define the hash code for this as the sum of the components.  This hash
     * code is used in turn by ItemKey.hashCode.
     */
    public int hashCode() {
        return hour + minute + amOrPm.hashCode();
    }


    /*-*
     * Derived data fields
     */

    /** The hour component of a time value, between 1 and 12 or 0 and 24 based
        on the clock style in use
     */
    protected int hour;

    /** The minute component of a time value, between 0 and 59 */
    protected int minute;

    /** Standard suffix used in 12-hour time value */
    protected AmOrPm amOrPm;


    /*-*
     * Process data fields
     */

    /** True if this is a valid time */
    boolean valid;

    /** True if this is an empty time, indicated by hour = 0, minute = 0, and
        amOrPm = "empty". */
    boolean empty;

```

```
}
```