Calendar problem documentation

# Model

## Time Class

### Constructor

***Time (int, int)*** take takes as parameters two integers: the first integer represents the hour, and the second integer represents the minutes. The constructor calls the validation method to ensure the correctness of the data.

### Validator

***validate (int, int)*** take takes as parameters two integers: the first integer represents the hour, and the second integer represents the minutes. It checks if the provided hours and minutes are correct i.e., 0 ≤ hours ≤ 23 and

0 ≤ minutes ≤ 59.

### Getters

***getHours ()*** and ***getMinutes ()*** are getter methods that retrieve the hours and minutes attributes respectively of a Time object.

### Setters

***setHours (int)*** and ***setMinutes (int)*** are setter methods that update the hours and minutes attributes respectively of a Time object to a given value.

### Comparators

***equals (Object)*** method checks if a given object is equal to the current Time object.

***smallerThan (Object)*** method checks if a given object is smaller than the current Time object.

***smallerThanOrEqual (Object)*** method checks if a given object is smaller than or equal to the current Time object.

***greaterThan (Object)*** method checks if a given object is greater than the current Time object.

***greaterThanOrEqual (Object)*** method checks if a given object is greater than or equal to the current Time object.

### Difference

***minus (Time)*** method returns the difference in minutes between the current Time object and the given Time object.

## Interval Class

### Constructor

***Interval (Time, Time)*** takes as parameters two objects of type Time. The first object is considered to be the lower bound of the interval and the second object is the upper bound of the interval i.e.: lower bound ≤ upper bound.

### Getters

***getLowerBound ()*** and ***getUpperBound ()*** are getter methods that retrieve the lower and upper bound respectively of the current Interval object.

### Overlap

***overlap (Interval)*** method checks if the current Interval object and the given Interval object are overlapping either entirely ([a, b] ⊂ [c, d]) or partially ((a ⊂ [c, d] and b ∉ [c, d]) or (b ⊂ [c, d] and a ∉ [c, d])).

# Repository

## Calendar Class

### Constructor

***Calendar(Time, Time)*** takes as parameters two objects of type Time and sets the minimum and maximum time ranges of the current Calendar Object. Initially, the Calendar object has an empty list of Intervals.

### Getters

***getMinRange (), getMaxRange()*** and ***getCalendarIntervals ()*** are getter methods that retrieve the minimum time range, the maximum time range and the list of time intervals respectively of the current Calendar object.

### Setters

***setMinRange (Time), setMaxRange(Time)*** and ***getCalendarIntervals (List<Interval>)*** are setter methods that update the minimum time range, the maximum time range and the list of time intervals respectively of the current Calendar object to a given value.

### Add Intervals

***addTimeInterval (Time, Time)*** method creates an Interval object with the two given Time objects and checks if the interval lies within the time limits of the current calendar. If the interval is out of bounds, the method throws an exception and the interval is not added to the calendar list, otherwise, it is added to the calendar list of intervals.

# Controller

## Controller Class

### Constructor

Controller (Calendar, Calendar, int) takes as parameters two Calendar objects and an integer which represents the meeting time expressed in minutes.

### Minimum

***min (Time, Time)*** method computes the minimum time between the two given parameters.

### Maximum

***max (Time, Time)*** method computes the maximum time between the two given parameters.

### Compute meeting time limits

***computeMeetingTimeLimits (Calendar, Calendar)*** method takes as parameters two Calendar objects and, based on each calendar’s time ranges, computes the time range for the wanted meeting. The lower bound is computed as the maximum between the two calendar’s lower ranges and the upper bound is computed as the minimum between the two calendar’s upper ranges. These will be the limits of the possible meeting scedule.

### Compute free time

***computeFreeTime (Calendar)*** method takes as a parameter a Calendar object and computes the free time based on the booked time intervals from its list. The method returns another Calendar, this time it contains the time ranges and a list of free time intervals for the given Calendar.

### Find available meeting time

***findAvailableMeetingTime ()*** method computes the possible free time intervals common for both Calendar attributes of the class. The method returns a new Calendar with the time limits computed by the method described above and a list of commonly available meeting times. The commonly available free time is computed based on the overlapping system. If a free interval from the first calendar overlaps with a free interval from the second calendar or vice-versa, then the common free time interval lower limit will be the maximum of the two overlapping intervals’ lower ranges and the common free time interval upper limit will be the minimum of the two overlapping intervals’ upper ranges.

# User Interface

## UI Class

### Constructor

***UI ()*** constructor builds a new consoled-based UI object. The UI class has an attribute of type Controller which is instantiated after the input is correctly provided by the user.

### Get minimum range

***getMinRange()*** method asks the user to provide the minimum time range for a calendar in the format “hh:mm” (e.g. 09:00, 14:03). If the format of the time range is not correctly provided or the time is invalid, the application will notify the user and ask them to provide the minimum time range again.

### Get maximum range

***getMaxRange()*** method asks the user to provide the maximum time range for a calendar in the format “hh:mm” (e.g. 09:00, 14:03). If the format of the time range is not correctly provided or the time is invalid, the application will notify the user and ask them to provide the maximum time range again.

### Get input

***getInput()*** method calls the getMinRange() and getMaxRange() functions one time. Following, the system asks the user to provide booked time intervals in the format “hh:mm,hh:mm” (e.g. 09:00,10:05). If the format of the interval is not correctly provided or the time is invalid, the application will notify the user and the interval will not be added to the calendar list. The process repeats until the user types “ok”, meaning they will not provide any more booked time intervals for the current calendar and the system will move on.

### Start

***start()*** method calls the getInput() function two times. Firstly, the user is asked to provide the necessary information for the first calendar. Secondly, the user is asked to provide the necessary information for the second calendar. In the end, the user is asked to input the meeting time expressed in minutes. The controller of the UI class is instantiated with the two calendars and the findAvailableMeetingTimes () function is called. The system displays the available intervals and exits.

# Exceptions

## InvalidBookedTimeException Class

This exception notifies the user that the booked time interval provided for a calendar exceeds the time ranges.

## InvalidTimeException Class

This exception notifies the user that the time provided is invalid.