CS 475 – Concurrent & Distributed Systems – Fall 2017
Assignment 4
A Calendar Tool for Workgroups
Due Date: 11:59 PM, Wednesday, December 6, 2017

1 Introduction

The goal of this assignment is to introduce you to the use of Java RMI as middleware for constructing distributed applications by building a calendar tool for work groups. The calendar tool helps individuals manage their schedules. It also aids groups of people in coordinating common events such as meetings. Each user controls her own calendar. The user may view this calendar or post events (appointments, to-do lists, etc) to the calendar. The calendar is also a coordinating tool. A user may view the calendars of other users, and schedule shared events such as meetings.

Unlike Assignments 1-3, the requirements for this assignment are specified relatively loosely. In other words, while the description below specifies the required functionality, you have much more flexibility with respect to the interfaces of the classes you need to implement for the project, and the approach you use to implement the functionality.

1.1 Calendars

A calendar is a database of events. An event is a record that contains three attributes: a time interval, a text event description, and an access control field. Each attribute is defined as follows:

- Time Interval: This attribute denotes the range of time over which the event is active. An example is "9 – 10 AM, March 10, 2010". We say that event A contains event B if the interval of A contains the interval of B. We say that events A and B are conflicting if their intervals intersect.
- Text Description: This is a simple text description of the event. For example, "Squash game with Mary".
- Access Control: Defines access rights to the event. There are four types of access control attributes: Private, Public, Group, and Open. They define the mode of permissible access to calendar events. The owner of the calendar is permitted to add, delete or modify any type of events on the calendar.

  Private events are accessible only by the owner. Public events may be read by any user.

  A group event is an event that includes a list of group members. Any group member can read or modify a group event, provided that it does not conflict with other group events and that it is contained within an open event on the calendars of group members. Any user can read the time interval of a group event but cannot see any other event data unless he is a member of the group.

  Open events are public events used to specify periods that are open for scheduling group events. Any user can view open events and create a respective group event that is contained in the open event, provided this group event does not conflict with any other group event.

1.2 System Architecture

The system you will build consists of *two objects per user*: a Calendar object and an object that implements the Client User Interface. In addition, there is a Calendar Manager, which is used for creating and keeping track of Calendar objects.

Each user's calendar is implemented by a single calendar object. The calendar object exports the operations described in Section 2 via Java remote interfaces. These interfaces are used by other calendar objects or by user interface objects to access and manipulate calendars.
The user interface lets the user view and modify calendars. It communicates with its calendar object via RMI to query or update calendar databases. The user interface and calendar object reside either on the same machine or on different machines across the network. Although the user interface usually requests services from the calendar, occasionally the calendar initiates communication with the user interface program to notify the user of events.

2 Calendar Object

Each Calendar object maintains a database of events for its user. As such, it should support the following classes of services that essentially query and modify its database and that of other users. Optional parameters appear in square brackets.

Retrieve Event [user time-range]

Retrieve the schedule of a user for the specified time-range. If user is omitted, it defaults to the owner of the calendar. The Calendar object will need to communicate with another object to retrieve a schedule if it does not maintain the calendar of the specified user. Note that the calendar only returns events that the requesting user has privileges to view.

Schedule Event [user-list event]

Schedule an event in calendars of each user specified in user-list. If user-list is omitted, schedule the event in the local calendar. A group event may be scheduled in the calendar of each proposed user only if the Access Control field of the respective event in every specified calendar is currently set to Open.

3 User Interface Program

The User Interface program lets the user perform the operations defined in the previous section. In particular, this program presents the user with an interface that permits her to view calendars, to modify appropriate events, and to schedule group events. The user interface usually communicates with its Calendar object to perform these services, but occasionally, the Calendar object might need to notify the user of scheduled events such as appointments. In this system, the user interface interacts solely with its designated Calendar which acts as proxy for any operations that require the services of other Calendars.

This assignment is not about user interface design. We want you to focus on the distributed computing aspects of the problem. A simple text-based interface will suffice. However, if you are familiar with Java, it is relatively straightforward to design a simple GUI for this application.

4 Calendar Manager

The Calendar Manager acts as a *class factory* for Calendar objects. Each Calendar User Interface program should have a command-line argument that is the name of its user. If a calendar does not already exist for that user, the User Interface object should create a calendar object for that user by invoking the appropriate method in the Calendar Manager object.

The User Interface can also query the Calendar Manager to obtain a list of names of other users in the work group.

5 Guidelines for this Assignment

This assignment deals with the design and implementation of the Calendar Tool described above. In particular, you have to design and implement the following:

- Calendar and Calendar Manager classes
- User Interface program, and
- Appropriate primitives, data structures and algorithms for robustly implementing the calendar database, and for communicating between Calendar objects.

For this assignment**, you have to use the software architecture shown in Figure 1 below**. In this architecture, all the calendar objects and the calendar manager reside in the same server process. You can assume that all these objects are transient, i.e., their lifespan does not exceed that of the server process that contains them.

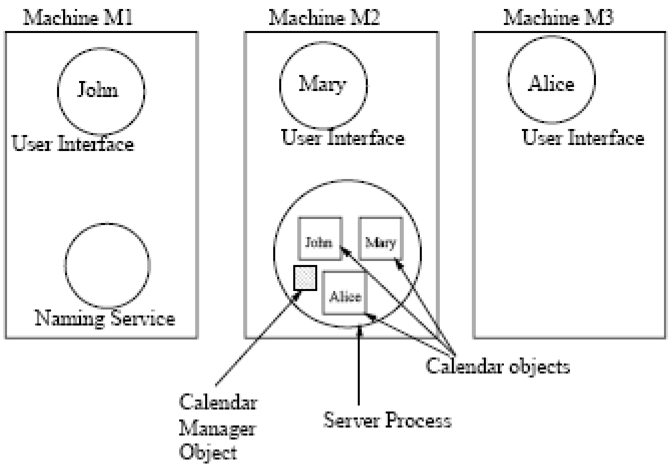

Figure 1: Software architecture for the Calendar tool

Keep in mind that multiple clients may invoke the methods of the Calendar and Calendar Manager objects concurrently. **Where necessary, your code should use the appropriate synchronization mechanisms to ensure correct operation (including avoiding potential deadlocks).**

6 Submission

You have to submit source code for the following: (i) the interfaces you declare (ii) the client program, (iii) the calendar and calendar manager classes (iv) the server program. In addition, you should submit a makefile or a README file with instructions as to how to compile your code.

You should create a tar file or zip archive containing all the files above and submit via Blackboard.

Your code should be reasonably well commented so that I can understand what it is doing.

For this assignment, you will need to give me a demo of your application in action. You will need to demonstrate the correctness of your implementation. In particular, you will need to demonstrate

- That users can schedule events in their personal calendars, as well as group events involving multiple users
- That users can view their own calendars as well as the public and open events in the calendars of other users

- That the user interface is able to alarm the user when events like appointments are about to occur.

Your implementation should have a reasonable level of user-friendliness and error checking built in. **The script that you will need to follow for your demo will be posted on Blackboard**.

Grading

Your grade for the assignment will be based both upon your demo as well as the correctness and quality of your design as follows:

- Design – 25%
- Correctness and functionality – 75%

The full breakdown of the points for correctness and functionality will be posted on the assignment's Blackboard page.

An easy way to get full credit for the design is to schedule a meeting with me where we go over your design. (Of course, to get full credit, you will then need to follow the recommended design in your project.) **I will only be available for design review meetings during the week of November 27. You should come prepared for the design review meeting with the code for the interfaces of the calendar and calendar manager classes, as well as a list of issues you would like to go over.**

The correctness and functionality of your implementation will be judged on the basis of your demo as well as a code review.