Code for the Meeting Scheduler

Write the object-oriented code to implement the design of the meeting scheduler problem.

We'll cover the following

We've reviewed the different aspects of the meeting scheduler and observed the attributes attached to the problem using various UML diagrams. Let's explore the more practical side of things, where we will work on implementing the meeting scheduler using multiple languages. This is usually the last step in an object-oriented design interview process.

We have chosen the following languages to write the skeleton code of the different classes present in the meeting scheduler:

- Java
- C#
- Python
- C++
- JavaScript

Meeting scheduler classes

In this section, we will provide the skeleton code of the classes designed in the class diagram lesson.

> **Note:** For simplicity, we are not defining getter and setter functions. The reader can assume that all class attributes are private and accessed through their respective public getter methods and modified only through their public method functions.

User

The `User` class refers to a participant taking part in a meeting. A user can either accept or reject an invitation. The definition of this class is given below:

```
public class User {
  private String name;
  private String email;
  public void respondInvitation(Notification invite);
}
```

The User class

Interval

The `Interval` class denotes the meeting interval (the start and end time).

```
public class Interval {
  private Date startTime;
  private Date endTime;
}
```

The Interval class

Meeting room

The `MeetingRoom` class will represent the meeting rooms, each having a specific capacity, a boolean to check if a room is available, and a list of intervals for which the room is booked. The definition of the class is provided below:

```
public class MeetingRoom {
  private int id;
  private int capacity;
  private boolean isAvailable;
  private List<Interval> bookedIntervals;
}
```

The MeetingRoom class

Calendar

The `Calendar` class contains a list of meetings to keep track of all the scheduled meetings. The definition of this class is provided below:

```
public class Calendar {
  private List<Meeting> meetings;
}
```

The Calendar class

Meeting

The `Meeting` class outlines the meeting details such as the number and list of participants, meeting time interval, and meeting room. It also has the option to add more participants. The definition of this class is shown below:

```
public class Meeting {
  private int id;
  private int participantsCount;
  private List<User> participants;
  private Interval interval;
  private MeetingRoom room;
  private String subject

  public void addParticipants(List<User> participants);
}
```

The Meeting class

Meeting scheduler

The `MeetingScheduler` class is the main class of the meeting scheduler and contains the organizer, which is responsible for scheduling and canceling a meeting as well as booking or releasing a room. It also checks if any meeting rooms are available for a meeting. In addition, there will be only one instance of the scheduler in the meeting scheduler. Therefore, the `MeetingScheduler` class will be a Singleton class to ensure that only one instance for the scheduler is created in the entire system.

The definition of this class is shown below:

```
public class MeetingScheduler {
  private User organizer;
  private Calendar calendar;
  private List<MeetingRoom> rooms;

  // Scheduler is a singleton class that ensures it will have only one active instance at a time
  private static MeetingScheduler scheduler = null;

  // Created a static method to access the singleton instance of Scheduler class
  public static MeetingScheduler getInstance() {
    if (scheduler == null) {
      scheduler = new MeetingScheduler();
    }
    return scheduler;
  }

  public boolean scheduleMeeting(List<User> users, Interval interval);
  public boolean cancelMeeting(List<User> users, Interval interval);
  public boolean bookRoom(MeetingRoom room, int numberOfPersons, Interval interval);
  public boolean releaseRoom(MeetingRoom room, Interval interval);
  public MeetingRoom checkRoomsAvailability(int numberOfPersons, Interval interval);
}
```

The MeetingScheduler class

Notification

The `Notification` class is responsible for sending notifications to users about any new meetings or cancelations. The definition of this class is provided below:

```
public class Notification {
  private int notificationId;
  private string content;
  private Date creationDate;

  public boolean sendNotification(User user);
  public boolean cancelNotification(User user);
}
```

The Notification class

Wrapping up

We've explored the complete design of the meeting scheduler in this chapter. We've looked at how the meeting scheduler can be visualized using various UML diagrams and designed using object-oriented principles and design patterns.

Back

Activity Diagram for the Meeting Scheduler

Complete

Next

Getting Ready: Movie Ticket Booking System