Concurrent & Distributed Systems

Assignment 1 2022

Due: Last Week of Term

Description:

Write a Java `Hotel` class which has methods to book rooms and check room bookings which could be called concurrently by multiple threads. Your task is to make sure there is no interference between threads (for example, no double-bookings) and at the same time maximum concurrency is allowed in the implementation of the class.

Your solution should be accompanied by a 1- or 2-page explanation of your program. The program and the report are marked together (there is no fixed split of marks between them).

Each hotel contains zero or more rooms and each room has a unique room number (the room numbers are not necessarily consecutive). Multiple users interact with the booking system to check the availability of rooms and to make bookings. Each booking has a unique booking reference generated by the clients of the booking system. Rooms can be booked for one or more (not necessarily consecutive) days. A booking can be for a single room, or (optionally) for several rooms on the specified day or days. Days are numbered sequentially from 0 (corresponding to some arbitrary date, e.g., 01/01/1970) and bookings can be arbitrarily far in the future.

Design and implement a Java `Hotel` class which provides the following public methods:

- `Hotel(int[] roomNums)`: constructs a hotel with room numbers as specified in *roomNums*. The rooms are initially unbooked.
- `boolean roomBooked(int[] days, int roomNum)`: returns true if the room *roomNum* is booked on any of the days specified in *days,* otherwise returns false.
- `boolean bookRoom(String bookingRef, int[] days, int roomNum)`: create a booking with reference *bookingRef* for the room *roomNum* for each of the days specified in *days.* Returns true if it is possible to book the room on the given days, otherwise (if the room is booked on any of the specified days) returns false.
- `boolean updateBooking(String bookingRef, int[] days, int roomNum) throws NoSuchBookingException`: updates the booking with reference *bookingRef* so that it now refers to the specified *roomNum for* each of the days specified in *days.* Returns true if it is possible to update the booking (i.e., the new booking does not clash with an existing booking), otherwise returns false and leaves the original booking unchanged. If there is no booking with the specified reference throws `NoSuchBookingException`.
- `void cancelBooking(String bookingRef) throws NoSuchBookingException`: cancels the booking with reference *bookingRef.* The room booked under this booking reference becomes unbooked for the days of the booking. If there is no booking with the specified reference throws `NoSuchBookingException`.

Note that the methods do not throw any checked exceptions other than those specified. A NoSuchBookingException class is provided as part of the assignment.

For extra credit you may optionally also provide methods:

- `boolean roomsBooked(int[] days, int[] roomNums)`: returns true if any of the rooms in *roomNums* is booked on any of the days specified in *days,* otherwise returns false.
- `boolean bookRooms(String bookingRef, int[] days, int[] roomNums)`: create a booking with reference *bookingRef* for the rooms in *roomNums* for each of the days specified in *days.* Returns true if it is possible to book the rooms on the given days, otherwise returns false.
- `boolean updateBooking(String bookingRef, int[] days, int[] roomNums) throws NoSuchBookingException`: updates the booking with reference *bookingRef* so that it now refers to the specified *roomNums* for each of the days specified in *days.* Returns true if it is possible to update the booking (i.e., the new booking does not clash with an existing booking), otherwise returns false and leaves the original booking unchanged. If there is no booking with the specified reference throws `NoSuchBookingException`.

If you are implementing the extended specification, the specification of the `cancelBooking()` method becomes:

- `void cancelBooking(String bookingRef) throws NoSuchBookingException`: cancels the booking with reference *bookingRef.* All the rooms booked under this booking reference become unbooked for the days of the booking. If there is no booking with the specified reference throws `NoSuchBookingException`.

Your implementation should allow safe concurrent access to instances of the `Hotel` class by multiple threads.

Notes:

The aim of the assignment is to gain experience in writing concurrent programs, not to produce a fully functional hotel booking system. Basically you need to write a class which stores information about the relationship between booking references (Strings), rooms (ints), and which days they are booked for (ints). Please don't be tempted to over-interpret the problem, e.g., by introducing "real" dates, or a user interface (in the code you submit).

The methods throw no checked exceptions other than those specified. You may assume that the dates and room numbers supplied as arguments are valid and booking references are unique. (This is arguably bad practice, but it simplifies things.) The set of possible days and rooms does not change during the execution of the program. However the set of bookings does change. For example, it is important to distinguish the case where a booking could not be updated because the new dates or room(s) conflict which an existing booking, and the case in which the booking cannot be updated because it hasn't been created yet (by `bookRoom()`) or has already been deleted (by `cancelBooking()`). `updateBooking()` and `cancelBooking()` should therefore throw `NoSuchBookingException` if the specified booking doesn't exist.