CPSC 1181 - Lab 10 [35 marks]

Objectives:

- Build a multithreaded server that implements a supplied protocol.

Submission:

- Zip up all of the Java files and submit them to D2L prior to the due date. **Due date set in D2L.**
- Unzipped submissions or submissions containing .class or other unneeded files will be penalized.

Overview

In this lab you will create a Server application that takes reservations for a single room hotel. It will allow multiple clients to connect simultaneously. To prevent corrupted data, the reservation data must use locking/unlocking.

Exercise 1

Create a class called Hotel. The hotel class will track the current reservations for your one-room hotel **for the month of May only. So it must only maintain reservations for the 1st through 31st of May.** You can design the class to suit your needs, but it must contain the following public methods.

- `String makeReservation(String name, int firstDay, int lastDay)`
  - Attempt to make a reservation from the firstDay to the lastDay (inclusive) for the person specified by name
  - **Restrictions**
    - A user may only have one reservation. If they already have a reservation, do not allow a new one
    - Do not make a reservation if values firstDay and lastDay are invalid in any way.
    - Only make a reservation if **all of the requested days are available**
  - Return a string with the information about the reservation if it was made or describing the reason the reservation was not made.
- `String cancelReservation(String name)`
  - If the specified person has a reservation, cancel it (all of their days become available) and return a message confirming this was done.
  - Otherwise, return a message stating the user did not have a reservation to cancel
- `String reservationInformation()`
  - Return a string containing the full reservation information for the month. For each day display either available or the name of the person who has the reservation for that day. Use newline characters to format the data well

Exercise 2

Add locking to the Hotel class. Since it is going to be accessed by multiple threads, possibly simultaneously, locking is required to ensure the data remains valid. Just lock all of the reservation information when necessary. This is a simple case, so you can either use a Lock object, or the synchronized keyword. **There is no need to use a condition, wait, await, signalAll, or notifyAll.**

Exercise 3

Create the server itself. The table below contains the protocol that the server should follow. After each command the server should respond with the String output from the appropriate Hotel method. The basic structure of a multithreaded server will follow that from our code examples discussed in lecture.

- The server should handle exceptions so that even if one client connection crashes, the server continues to run
- If the server receives an invalid command

CPSC 1181 - Lab 10 [35 marks]

Protocol

- When a client connects the server should send a welcome message to the client. Then await commands as specified in the protocol.
- When a client first connects set the user's initial name to a default value until the client updates it.
- The commands are sent as strings and each 'parameter' is sent separately. So the USER command requires two writes to the output stream. One for the command "USER" and one for the new name.

Client Request	Description
USER ***n***	Change the current user to the String n. Server respose: "Hello, " + n
RESERVE ***first last***	Attempt to make a reservation for the current user from day first to day last (inclusive)
CANCEL	Cancel any reservations made for the current user.
STATUS	Send availability information as provided by the Hotel method.
QUIT	Quit the connection. Server response: "Closing Connection"
Anything else…	Close the connection. Server response: "Invalid command: Closing Connection"

You have been provided with HotelClient.java. This is a client that allows you to interactively make reservations. If your server follows the protocol as specified, the client will function with your server.

Bonus: Exercise 4
Write your own client. Your client should test your server's commands**. Don't involve user input**, just test all of the commands in the protocol ensuring that everything is working as it should.

Marking Rubric:

Style, Convention, Documentation [5 marks]
Hotel [15 marks]
+3 data members/constructor
+5 makeReservation
+3 cancelReservation
+2 reservationInformation
+2 correctly using locking

HotelServer [15 marks]
+5 Basic Multithreaded Server Code
+10 Runnable class that handles clients.

HotelTestClient [5 marks]
+5 bonus marks