CPSC 1181 – Assignment 7 [30 marks]

Objectives:

- Build a hotel reservation application and use multithreading to simulate multiple customers simultaneously attempting to reserve the room

Submission:

- Submit a **zip file** containing **only** the .java files to Brightspace prior to the **due date set in Brightspace**. Do not include class or other files
- Submissions that are less than 24 hours late receive a 1% per hour late penalty. Submission that are more than 24 hours late will not be accepted.
  - **An extension will be applied if you submit late and have an extension remaining**
- Submissions that are unzipped or that contain .class or other unneeded files will be penalized.
- **IMPORTANT: If you are unsure of your submission for any reason, submit it AND email it to me.**

Working With a Partner

- You must tell me **in-person** with your partner that you are working as partners to have permission. This must be done before the assignment is posted.
- If one partner no longer has an extension, neither partner can use an extension.
- Only one group submission is required: submitted by either partner
  - If both submit, we will mark whichever we open first
- **Add a second @author to your JavaDoc**
- **You must inform me if you are not working with your partner from the previous assignment.**

Overview

In this lab you will create a Hotel application that takes reservations for a small (1 room) hotel. It will allow multiple clients to use the system simultaneously.

Make sure you restrict yourself to the techniques taught in the lectures for the course. There are other methods to accomplish multithreading that we are not using.

You are given a class called Hotel. **Download and do not edit Hotel.java. Include the file in your submission so that the marker does not need to add it manually.** The hotel tracks the current reservations for your hotel **for the month of December only. So it only maintains reservations for the 1st through 31st of December of the one room.**

It contains the following methods.

- boolean requestReservation(String user, int firstDay, int lastDay)
  - Attempt to make a reservation from the firstDay to the lastDay (inclusive) for the person specified by name. If firstDay == lastDay that is ok and is a one day reservation.
  - **Restrictions: a reservation will not be made if…**
    - The user already has a reservation
    - any of the requested days are not available
  - Return true if the reservation was made, false if it was not, throws an exception if firstDay and/or lastDay are invalid
- boolean cancelReservation(String user)
  - If the specified person has a reservation, it will be cancelled (all of their days become available) and returns true
  - Otherwise, returns false if they did not have a reservation to cancel
- String reservationInformation()

- Returns a string containing the full reservation information for the month. For each day displays either available or the name of the person who has the reservation for that day.

Exercise 1
Write an automated PeriodicCustomer class that implements Runnable. Your customer should have a name and should test your hotel's commands (The customer will need a reference to the hotel object). Don't involve user input, just test all of the commands ensuring that everything is working as it should. **This will be somewhat similar to the periodicDeposit and periodicWithdrawal classes from the slides.**

- The client should periodically do either attempt to make a reservation or cancel a reservation. Each time choose randomly between the two. (So, the first action may be cancel, the customer might try to make a reservation when they already have one, etc., even though those don't make sense)
  - Make a new reservation starting and ending on random days. **Make sure that the first and last days are valid.** Your random numbers should include every possible valid first/last day combination.
  - Cancel current reservations.
- After doing so, print a message to the console stating what happened. The message should always include the customer's name.
  - Reservation made: Chris from 3 through 5
  - Reservation unsuccessful: Chris from 3 through 5
  - Reservations successfully canceled for Chris
  - Reservations not canceled for Chris, no current reservation.
- The Customer should "shut down" if it has been interrupted. (It's ok to just check for the exception created by the call to sleep) Display a message to the console such as "Periodic Test Customer Chris Shutting Down" before the run method exits.
- If not interrupted, the customer should continue indefinitely. You can use a while(true) here.

Exercise 2
Create a HotelTester class that will create at least three new threads, each with its own customer object. After the program has been running long enough for the customers to attempt at least 5 actions each, interrupt all of the threads. Use the join function to ensure that all of the threads have shut down, and then print out the current reservation information for the hotel.

Marking Rubric:

Style, Convention, Documentation [5 marks]

PeriodicCustomer [15 marks]

HotelTester [10 marks]