Independent Programming Assignment 3

Assigned/Due: See zyBooks for exact due date

*All portions of this project due by <u>11:55pm on the due date</u>.

YOU <u>MAY NOT</u> WORK WITH OR WRITE CODE TOGETHER WITH A PARTNER.

This assignment will be graded on the following criteria:

Compiles and runs	30%	(Your program **MUST** compile in order to be considered for grading)
Correctness	40%	(Your program must satisfy each requirement of the specifications)
Style	20%	(Your program must use comments and have user-friendly output)
Instructions	10%	(You must include any other materials requested in the lab)

Description

You recently invited your parents to visit you at APU for the weekend. To your surprise, you discovered afterwards that there are no hotels in the area. Needless to say, your parents did NOT enjoy your Dorito-crumb-littered futon bed. Thus, you have taken it upon yourself to build a new hotel chain: The Parental Paradise. Given your billionaire status, you plan to start out with 5 locations: Azusa, Glendora, Covina, San Dimas and Pasadena.

You plan to rent your hotel rooms out to anyone who is willing to pay the price. Although hotels typically have many rooms and you are flowing with cash, since you are new to the business, you have decided to start with a single penthouse suite at each hotel (only the best for your parents, of course). Thus, in this project, you will create a java application which acts as a hotel reservation system.

Assumptions

To make things less complicated, you may make the following assumptions:

1. There is only one room in each hotel.
2. Your guests will not stay for a period that spans over the end of one month and beginning of another month (wow, how convenient!).
3. We will assume ALL months have 31 days (simplifies error checking).

Requirements

1. Your program must first read from the **included Hotels.txt** file. It contains a number of lines of data; each line corresponds to a hotel in your chain (you may add more if you'd like and your code should not break). Each line is formatted as follows:

   Hotel(uniqueId, hotelName, streetAddress, city, stateAbbreviation, pricePerNight)

   You will need to create a simple *Hotel* class to encapsulate this data (template provided).

2. Your program must ask the user which hotel they'd like to stay in by listing all the hotels (with prices, address, etc.) with a number to the side, and then prompting the user to select a hotel by entering the corresponding number (corresponding to the unique ID).

   Once they've selected a hotel, you need to prompt the user for check-in month/day and check-out month/day. If they enter two different months, inform them that hotel policy does not allow them to check out during a different month (we have some interesting rules) and ask them to try again until they enter a valid range.

   Once they enter a valid range, according to the last paragraph, you must check all previously made reservations (which you will be storing in a file). If there is no conflicting reservation, inform the user that their reservation was successful and tell them how much they will owe for their entire stay (don't worry about tax). If there is a conflict with a previous booking, then inform the user and allow them to try different dates.

   The included runnable jar file will demonstrate exact syntax (**all months and days will be entered as integers**) and flow (**template provides most of syntax/flow for you**).

3. When you make a reservation, you must store it in a separate file (call it **Reservations.txt**) such that it can be referenced when the program is opened to make a reservation with a new customer later. Create a *Reservation* class to encapsulate this concept. To associate a particular reservation (which basically consists of the check-in month/day, check-out month/day) with a particular hotel, **you will also need to store a hotel id (same thing as the hotel's *uniqueId*) in the Reservation which directly corresponds to one of the hotel unique ids**. **You must use serialization techniques to save the reservations to a file**.

   HINT: You should create a Reservation class (template file provided), which encapsulates the above idea. You could then add an ArrayList of Reservations to your Hotel class. After you read in and create your hotel objects, you'd read in the reservations from file and add them to the hotel's reservation ArrayList (i.e., it would be a good idea for the Hotel class to have an "ArrayList<Reservation> reservations" member variable). This general pattern is modeled in the templates.

File Setup & Templates

Files should be setup in eclipse similarly to the lab(s) with file I/O. All .java files should be in the "src" folder and all .txt files should be right NEXT to the "src" folder (that is, directly in the project folder).

You have been provided with a populated Hotels.txt file, an empty Reservations.txt file, and three heavily-templated .java files. If you'd like, you can start from scratch and ignore the general structure I've created in my template files. However, students have traditionally struggled with starting from scratch on this project and so the main program flow has been provided for you in main() in CS125_Project3_Client.java…you do not need to edit main() other than your name if you choose to use the templates as a starting point.

The three java files contain 19 TODO statements to point out where code needs to go. They do not necessarily need to be implemented in numerical order, but I would strongly recommend completing the Reservations.java class first, the Hotel.java class second and THEN completing the method definitions in CS125_Project3_Client.java last.

Runnable Jar

A runnable Jar file has been provided along with the templates to allow you to test your logic/output against a demo program.

Submission Instructions

1.) Templates: Use the relevant template file(s) for this lab found in the Google Drive
   a. Use the specified .java source file template for each [CODE] problem and **DO NOT change the file or class names** (doing so will cause your code to receive a 0 by the autograder)
      i. Make sure to update the header and name *System.out.print* statements
   b. Submit the console results (showing input and output) for at least **2 different test cases** you created to convince yourself that your program is working properly; place test cases at the very end of your .java file in the space provided by the template for test cases.
      i. A single test case consists of 2-3 runs of the program showing that the program is correctly keeping track of reservations across multiple runs.

2.) zyBook Submission: **Make sure your name is on all files you turn in.**
   a. Submit the relevant .java file(s) under the appropriate lab assignment
      i. Check to make sure all the files contain your latest work before submitting
   b. The final grade and grading feedback will be returned to you via the Canvas assignment section