Spring 2022 / CMPS 251 / Project / Phase 1

Deadline: Sat Apr 16 2022 @ 23:59.

Penalty: -1% per hour for late submissions.

Description

The goal of this project is to develop a system to manage a hotel booking system using object-oriented programming while satisfying a list of requirements. This system will be developed in two phases: The first phase is a console application with basic text input and output and the second phase is an application with a graphical user interface and object serialization for data persistence.

Requirements

The main requirement of the hotel is to allow its guests to make and cancel reservations, check in and out of their reserved rooms, and pay for the duration of their stay in the hotel.

The following are a few sample interactions with the system:

1. A guest calls to book a room in the hotel. A staff member gathers the guest's details and then is presented with the list of available rooms along with their type and features. The staff member can further filter the list of rooms based on the guest's selection, for example, room area, high-rise floor, whether a room has a balcony or a kitchenette, and other features. The staff member then proceeds to book a room for a certain date range (start and end date) and provides the guest with a booking number to confirm their reservation.

2. A guest presents itself to reception with an ID for checkin. The information in that ID is used along with the booking number to check in a guest into their room.

3. A guest presents itself to reception with their ID for checkout. The guest pays their due amount and receives an invoice.

4. A guest calls reception and cancels their booking.

5. A staff logs into the system and can access the list of guests currently residing in the hotel, the list of previous guests, the list of reserved rooms, the list of available rooms, and other useful lists.

You need to build main console class that uses a collection of classes and fulfill the aforementioned requirements of the hotel. This main class will be in charge of all user input and output. Your initial screen should allow a staff member to login and track their actions as every action in the system must be associated with a staff member. You can hardcode all user input during this phase while developing and testing your application.

You should handle as many edge cases as possible, for example, an invalid booking id, use exceptions to handle erroneous input, for example, the end-user inputting a letter where a digit is expected, and make use of the object-oriented concepts you have practiced during the lab sessions when designing your application, namely, encapsulation, association, aggregation, composition, inheritance, abstraction, and polymorphism.

Instructions

1. All members of the team must contribute equally to the implementation and documentation of the project.
2. All files, classes, interfaces, attributes, methods, parameters, and return values must be documented using Javadoc comments/tags. In addition, every source file should have a list of authors along with a date and version number using the corresponding Javadoc tags.
3. Each team must submit an archive export of their project's source files and generated documentation along with a project report listing the team members, a short description of the project and its implementation, a list of the issues and challenges that were encountered and how they were resolved, and screenshots of input/output for test cases covering every aspect of your project; each screenshot should have a brief description of the corresponding test case.
4. Each member must submit a personal report describing their exact contribution to the project and elaborating on the advantages/disadvantages and challenges encountered while working in a team. This is an individual report that is not shared among team members and will be submitted separately.

Evaluation

[50%] Complete and correct implementation of the project requirements.

[10%] Quality and tidiness of the source files, both code and comments.

[10%] Documentation authoring and generation using Javadoc.

[20%] Project report presentation and source/documentation archive.

[10%] Personal report.