**CS5004 Final Project Report**

1. Academic Integrity Statement

文本, 信件

描述已自动生成

1. Introduction
2. Project Function

My hotel management system uses the MVC design pattern to separate logic and user interaction. It employs a custom generic LinkedList<T> to handle different objects, creating and processing lists of rooms, orders, and customers. Additionally, it fully utilizes advanced functions of the generic linked list, such as filter, map, and reduce, to select and calculate nodes that meet specific criteria, enabling management of rooms, orders, and finances.

Room Management Functions:

* View All Available Rooms: This feature lists all rooms that are currently vacant and ready for new reservations. It categorizes the available rooms based on room types and return the 4 types available LinkedList of their room number. For instance, when the operation is selected, it might show available Standard Rooms like "104" and "208" indicating that they are available for guests.
* Update Room Status: It allows admin to update the status of a room. Room statuses include Available, Reserved, Occupied, and Cleaning. By entering the room number and selecting a new status, the system immediately reflects this change. For example, changing room "312" from Reserved to Available makes it immediately open for new bookings.

Reservation Management Functions:

* Add a New Reservation: admin can add a new reservation for a customer by following prompts for customer selection, room selection, and check-in/check-out dates. Once all details are entered, the reservation is added to the system, and the selected room's status is updated to Reserved. If a VIP customer makes a reservation, the system applies any appropriate discounts to the pricing.
* Cancel a Reservation: admin can cancel existing reservations. The program displays current reservations, and by selecting a reservation ID, it cancels the reservation, making the rooms involved available again for booking.
* View Today's Reservations: Displays a list of all reservations starting or ending on the current day, which help admin manage day-to-day operations and ensure that guests are checked in and out smoothly.

Financial Management Function:

* View Today's Income: It calculates the total income from all of today's check-outs by recursively going through the LinkedList of today's bookings and then using 'reduce' to add up the income, which helps the admin to track daily revenue.

1. UML & Design

UML: 图示

描述已自动生成

Design:

* Model Layer:
* HotelModel forms the core of the business logic. It aggregates LinkedList instances for AbstractRoom, Reservation, and Customer, facilitating centralized data management. The methods getAvailableRooms() and calculateDailyRevenue() are prime examples of encapsulated operations that interact with the data structures to perform complex queries and calculations.
* AbstractRoom is an abstract superclass for different types of rooms (StandardRoom, DeluxeDoubleRoom, DoubleRoom, Suite). It defines shared attributes and an updateStatus method, ensuring consistent for subclasses, which display inheritance and polymorphism.
* Customer and Reservation classes represent entities with distinct responsibilities. Customer holds personal and VIP status information, while Reservation manages booking details and interacts with Customer, showcasing association and composition relationships.
* View Layer:
* HotelView handles all user interactions, displaying menus and capturing input. It operates independently of the model but provides the user interface for the operations defined in the model.
* Controller Layer:
* The HotelController serves as the command center, guiding the information exchange between the HotelModel(the system’s data hub) and the HotelView(the interface users interact with). . It directly employs methods such as listAvailableRooms() to display vacant accommodations and updateRoomStatus() to modify room availability

Data Structures:

* LinkedList<T> is a generic linked list implementation used throughout the model to manage collections of rooms, customers, and reservations. it not only facilitates fundamental list operations, such as adding (addAtBottom()) and removing (removeAtIndex()) elements but also integrates advanced functional programming techniques through higher-order functions. These include: filter(Predicate<T> predicate), map(Function<T,R> converter), reduce(T initialValue, BiFunction<T,T,T> combiner), countIf(Predicate<T> predicate),etc.

we mainly use the following three different linked lists to manage corresponding objects.

LinkedList<Room> - manages the information and status of all rooms.

LinkedList<Reservation> - manages all reservation information.

LinkedList<Customer> - manages all customer information, including VIP status.

Database:

* The hotel management system uses the HotelData class to set up initial room conditions, customer details, and reservations, creating a realistic hotel environment serves like simple database. Users interact with this setup through the HotelView, initiating tasks like making reservations or updating room statuses. These actions are handled by the HotelController, which updates the HotelModel to reflect changes dynamically, ensuring the system is always up-to-date based on user inputs and initial configurations.

1. Grade yourself

Reflecting on my performance and considering the comprehensive rubric, I confidently give myself full marks on all components except the code walkthrough. My project seamlessly integrated recursion, showcased through the recursive operations within the custom LinkedList class, enhancing code simplicity and readability. Abstract classes and interfaces form the backbone of my design, enabling a robust logical structure, as evidenced by the AbstractRoom class and IRoom interface. Generics and lambda expressions are skillfully employed throughout, enabling a level of abstraction that captures the essence of modern Java practices. Higher-order functions like map, filter, and fold are not just implemented but are integral to the system's functionality, particularly in data processing within the HotelModel.Us e generic LinkedList ADT, tying in with the MVC design pattern that strategically divides the application's concerns. SOLID principles are adhered to with diligence, which can be seen in the clear separation of concerns and single-responsibility classes.

1. Concept Map

|  |  |  |
| --- | --- | --- |
| Topic | Location | Description |
| Recursion | LinkedList.java | countIf, filter, map, reduce.etc. |
| Abstract Classes & Interfaces | AbstractRoom.java & IRoom.java & Node<T> | Used to establish a common structure for different types of objects. |
| Generics & Lambda | LinkedList<T>.java & Node<T>.java | Generics provide type safety and reusability for the linked list data structure. |
| Higher Order Functions | LinkedList<T>.java | filter, map, reduce.etc. |
| ADT | LinkedList<T>.java | Manage hotel Rooms, Reservation Customers and Finances |
| MVC Design Pattern | HotelModel java,  HotelController java,  HoterView java | Separate business logic and user interaction |
| SOLID Principles | Throughout code | Principles are evident in the design and structure of the classes and their relationships. |