A Hotel Reservation System automates the process of managing bookings, room availability, and guest information within a hotel. It streamlines operations by centralizing data related to rooms, guests, and reservations, offering staff a comprehensive tool to efficiently handle bookings, check room availability, and manage guest details. The system typically includes features for making, modifying, and canceling reservations, generating bills, and maintaining a record of all transactions. By digitizing these tasks, the system reduces manual errors, enhances customer service through quicker check-ins and check-outs, and provides real-time updates on room availability. Staff benefit from simplified workflows, improved accuracy in managing reservations, and better communication with guests, ultimately leading to smoother operations and enhanced guest satisfaction.

The Hotel Reservation System implemented in Java revolves around a set of interconnected classes and interfaces designed to manage guest bookings and room availability seamlessly. At its core are classes like Guest, which stores essential information such as the guest's name and contact number, and Room, which represents each unique accommodation in the hotel. Each room instance keeps track of its number, type (standard, deluxe, or suite), and whether it's currently available for booking, with methods in place to reserve or release rooms as needed.

Reservations are facilitated through the Reservation class, which ties together a guest, a specific room, and dates for check-in and check-out. It also implements the Billing interface to calculate the total bill based on the room type and the duration of the stay. This integration ensures that bookings are accurately reflected in room availability, coordinating bookings and releases smoothly through methods within the Room class.

The Hotel class serves as a central hub for managing both rooms and reservations. It maintains lists of Room objects and Reservation objects, enabling functionalities such as adding new rooms, checking room availability for specified dates, making reservations, modifying existing bookings (if rooms are available), canceling reservations, and checking out guests. These operations are crucial for maintaining operational efficiency and providing a positive experience for guests and staff alike.

Using Java's object-oriented capabilities, the system employs practical data structures like lists (ArrayList) to store and manage rooms and reservations effectively. The algorithms implemented for checking availability, handling modifications, and managing reservations are designed to handle various scenarios, ensuring robustness in handling guest bookings and adjustments seamlessly. Overall, the system aims to be user-friendly, scalable, and reliable, meeting the diverse needs of both guests and hotel personnel in managing reservations efficiently.

Classes

1. **Guest:**
    - Represents a guest with a name and phone number.
    - **Attributes:**
        - `name: String` - Represents the guest's name.
        - `phoneNumber: String` - Represents the guest's phone number.

- **Methods:**
  - `Guest(String name, String phoneNumber)`: Constructor to initialize the guest.
  - `getName(): String`: Returns the name of the guest.
  - `getPhoneNumber(): String`: Returns the phone number of the guest.

2. **Room:**
   - Represents a room in the hotel.
   - **Attributes:**
     - `roomNumber: int` - Represents the room number.
     - `roomType: RoomType` - Represents the type of the room (Standard, Deluxe, Suite).
     - `available: boolean` - Indicates if the room is available or booked.
   - **Methods:**
     - `Room(int roomNumber, RoomType roomType)`: Constructor to initialize the room.
     - `getRoomNumber(): int`: Returns the room number.
     - `getRoomType(): RoomType`: Returns the type of the room.
     - `isAvailable(): boolean`: Returns true if the room is available, false otherwise.
     - `bookRoom(): void`: Marks the room as booked.
     - `freeRoom(): void`: Marks the room as available.
3. **Reservation:**
   - Represents a reservation made by a guest for a room.
   - **Attributes:**
     - `guest: Guest` - The guest making the reservation.
     - `room: Room` - The room being reserved.
     - `checkInDate: LocalDate` - The date when the guest checks in.
     - `checkOutDate: LocalDate` - The date when the guest checks out.
   - **Methods:**
     - `Reservation(Guest guest, Room room, LocalDate checkInDate, LocalDate checkOutDate)`: Constructor to initialize the reservation.
     - `calculateBill(): double`: Calculates the total bill for the reservation based on room type and duration.
     - `getGuest(): Guest`: Returns the guest associated with the reservation.
     - `getRoom(): Room`: Returns the room reserved.
     - `getCheckInDate(): LocalDate`: Returns the check-in date.
     - `getCheckOutDate(): LocalDate`: Returns the check-out date.

Main Class: HotelReservationSystem

- Entry point for the Hotel Reservation System.
- Provides functionality for making, modifying, and canceling reservations, checking out guests, and printing all reservations.

Functionality

1. **Making a Reservation:**
    - Guests can reserve a room by specifying their name, phone number, desired room type, check-in date, and check-out date.
2. **Modifying a Reservation:**
    - Guests can modify an existing reservation by changing the check-in and check-out dates, provided the room type is available for the new dates.
3. **Canceling a Reservation:**
    - Guests can cancel their reservation, freeing up the room for future bookings.
4. **Checking Out a Guest:**
    - Hotel staff can check out a guest by entering the guest's name, freeing up the room and removing the reservation from the system.
5. **Printing All Reservations:**
    - Displays a list of all current reservations, including guest details, room number, check-in date, check-out date, and total bill.

Usage

1. Compile the Java file: `javac HotelReservationSystem.java`
2. Run the program: `java HotelReservationSystem`
3. Follow the on-screen prompts to interact with the Hotel Reservation System.

Dependencies

- None

Assumptions

- The system assumes a single hotel with multiple rooms of different types: Standard, Deluxe, and Suite.
- Each room type has a fixed rate per night.
- The system does not handle concurrent access by multiple users.

Here is the class diagram: