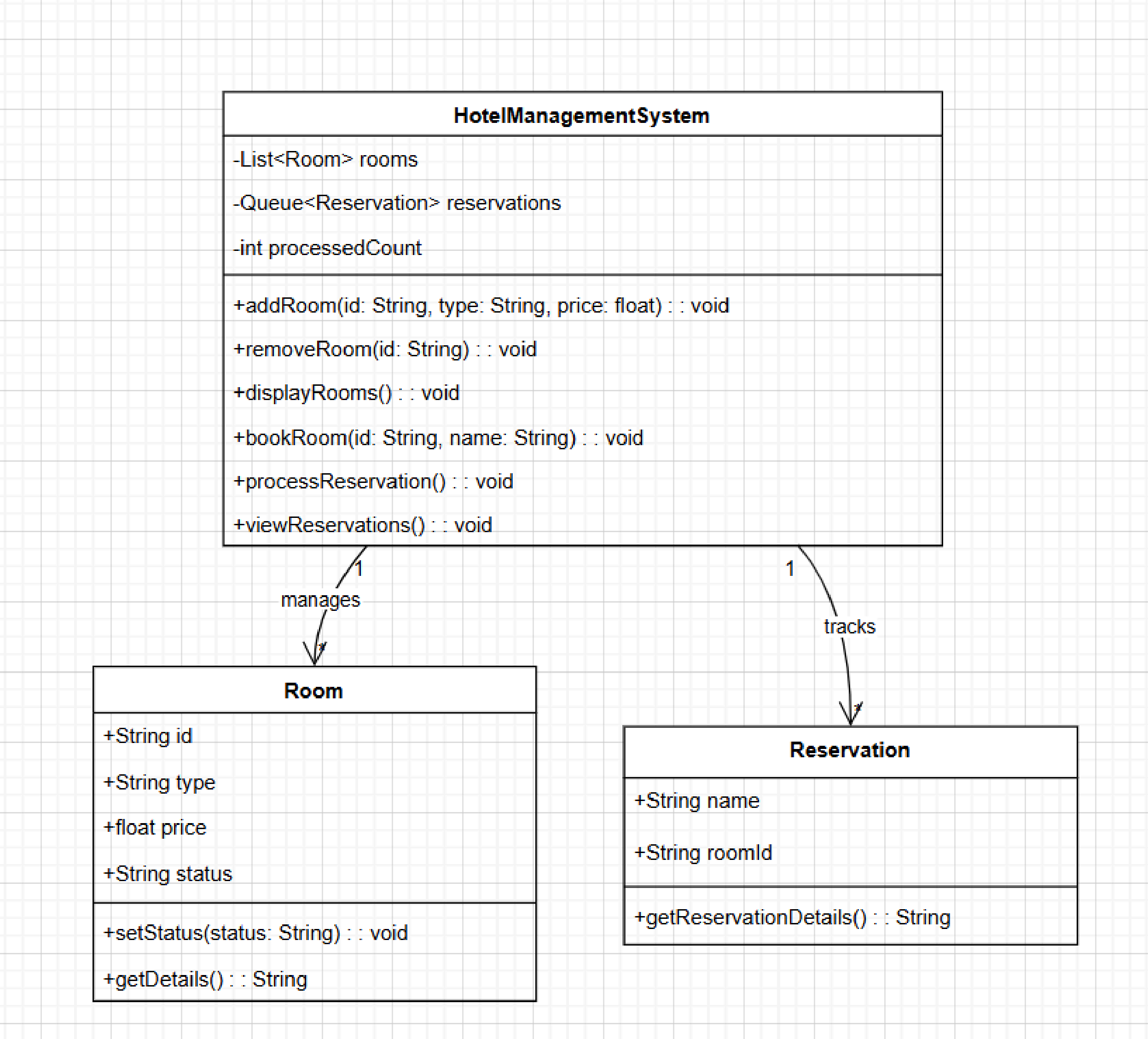
HotelManagementSystem
-List<Room> rooms
-Queue<Reservation> reservations
-int processedCount
+addRoom(id: String, type: String, price: float) : : void
+removeRoom(id: String) : : void
+displayRooms() : : void
+bookRoom(id: String, name: String) : : void
+processReservation() : : void
+viewReservations() : : void
1
manages
*
1
tracks
*
Room
+String id
+String type
+float price
+String status
+setStatus(status: String) : : void
+getDetails() : : String
Reservation
+String name
+String roomId
+getReservationDetails() : : String

Project Overview

The project is a **Hotel Management System** that provides functionalities for managing hotel rooms and customer reservations where users can:

- Add new rooms to the hotel system.
- Book rooms for customers.
- Process room reservations in the order they were received.
- Display all available or booked rooms with details.
- View the status of processed and pending reservations.
  This system integrates fundamental data structures and algorithms to efficiently manage the room inventory and reservation queue, ensuring an organized and user-friendly experience.

Data Structures and Algorithms Used in the Project

Data Structures

1. **List (`rooms`)**
   - **Purpose**: Stores all room details (ID, type, price, and status) as dictionaries.
   - **Usage**:
     - Add rooms.
     - Search for rooms to book or remove.
     - Display the details of all rooms.

Example:

```
rooms.append({"id": room_id, "type": room_type, "price": room_price,
"status": "available"})
```

2. **Queue (`reservations using deque`)**
   - **Purpose**: Manages reservations using a **First-In, First-Out (FIFO)** mechanism.
   - **Usage**:
     - Add new reservations to the queue.
     - Process reservations in the order they were received.

Example:

```
reservations.append({"name": name, "room_id": room_id})
reservation = reservations.popleft()
```

3. **Dictionary (within `rooms` and `reservations`)**
   - **Purpose**: Organizes data for rooms and reservations with attributes like ID, type, price, status, and customer name.
   - **Usage**:
     - Represents each room and reservation for easy access and modification.

Algorithms

1. **Linear Search**
   - **Purpose**:
     - To locate rooms by their ID for booking or removal.
   - **Implementation**: Iterates through the list of rooms to find a matching room ID.
   - **Complexity**: **O(n)**, where `n` is the number of rooms.

Example:

```
for room in rooms:
    if room["id"] == room_id:
        # Perform booking or removal
```

2. **FIFO Queue Operations**
   - **Purpose**: Ensures reservations are processed in the same order they were made.
   - **Implementation**: Using `popleft()` from the `deque` library.

Example:

```
reservation = reservations.popleft()
```

Features Implemented in the Project

1. **Room Management**
   - **Add Room**: Add a new room with attributes like ID, type, price, and availability status.
   - **Remove Room**: Remove a room if it is not booked.
   - **Display Rooms**: View all room details, including ID, type, price, and booking status.
2. **Room Booking**
   - Book an available room for a customer by updating the room's status and adding the reservation to the queue.
3. **Reservation Processing**
   - Process the next reservation in the queue using FIFO order and display the processed customer's details.
4. **Reservation Status**
   - Track and display the number of processed and pending reservations, along with the pending reservation details.
5. **Interactive Menu**
   - A user-friendly menu system allows users to navigate through functionalities like adding/removing rooms, booking rooms, and processing/viewing reservations.

The python code for the project:

```
from collections import deque

# Data
rooms = []
```

```
reservations = deque()
processed_count = 0

def add_room():
    """Add a new room to the hotel."""
    room_id = input("Enter room ID (e.g., 101): ")
    room_type = input("Enter room type (e.g., Single, Double): ")
    try:
        room_price = float(input("Enter room price per night: "))
        rooms.append({"id": room_id, "type": room_type, "price": room_price, "status": "available"})
        print(f"Room {room_id} added successfully!")
    except ValueError:
        print("Invalid price. Please enter a numeric value.")

def display_rooms():
    """Show all rooms and their details."""
    if not rooms:
        print("No rooms available!")
        return
    print("\nRooms:")
    for room in rooms:
        print(f"Room {room['id']}: {room['type']} | ${room['price']} | {room['status']}")

def remove_room():
    """Remove a room from the system."""
    room_id = input("Enter the room ID to remove: ")
    for room in rooms:
        if room["id"] == room_id:
            if room["status"] == "available":
                rooms.remove(room)
                print(f"Room {room_id} has been removed successfully.")
                return
            else:
                print(f"Cannot remove Room {room_id} because it is currently booked.")
                return
    print("Room not found. Please check the room ID and try again.")

def book_room():
    """Book an available room."""
    room_id = input("Enter the room ID to book: ")
    for room in rooms:
        if room["id"] == room_id:
            if room["status"] == "available":
                room["status"] = "booked"
                name = input("Enter your name: ")
                reservations.append({"name": name, "room_id": room_id})
                print(f"Room {room_id} booked successfully for {name}!")
                return
            else:
                print(f"Room {room_id} is already booked.")
                return
    print("Room not found. Please check the room ID and try again.")

def process_reservation():
    """Process the next reservation."""
    global processed_count
    if reservations:
        reservation = reservations.popleft()
        processed_count += 1
        print(f"Processing reservation for {reservation['name']} in Room {reservation['room_id']}.")
    else:
        print("No reservations to process.")

def view_reservations():
    """View processed and pending reservations."""
    print(f"\nReservations Processed: {processed_count}")
    print(f"Reservations Pending: {len(reservations)}")
    if reservations:
        print("Pending Reservations:")
        for reservation in reservations:
            print(f"Name: {reservation['name']}, Room ID: {reservation['room_id']}")

def main():
    """Main menu for the Hotel Management System."""
    while True:
        print("\n--- Hotel Management System ---")
        print("1. Add Room")
        print("2. Display Rooms")
        print("3. Remove Room")
        print("4. Book Room")
        print("5. Process Reservation")
        print("6. View Reservations")
        print("7. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
            add_room()
        elif choice == "2":
            display_rooms()
        elif choice == "3":
            remove_room()
        elif choice == "4":
            book_room()
        elif choice == "5":
            process_reservation()
        elif choice == "6":
            view_reservations()
        elif choice == "7":
            print("Goodbye!")
            break
        else:
            print("Invalid choice! Try again.")

# Corrected entry point
if _name_ == "_main_":
    main()
```