****

**NAME NASIR KHAN**

**ROLL NO 2212231**

**PROGRAM BSCS**

**SECTION 3-D**

**COURSE DSA LAB**

**SIR NAME SAYED M.HASSAN**

**HOTEL MANAGEMENT SYSTEM**

**INTRODUCTION:**

The provided Java code constitutes a basic Hotel Management System designed to handle room occupancy, guest information, and room status within a hotel. This system employs object-oriented principles and a linked list data structure to manage rooms and their respective statuses.

At its core, the code defines a `Room` class encapsulating room details such as room number, occupancy status, and guest information. This class features methods to occupy a room, vacate it, and retrieve its current status.

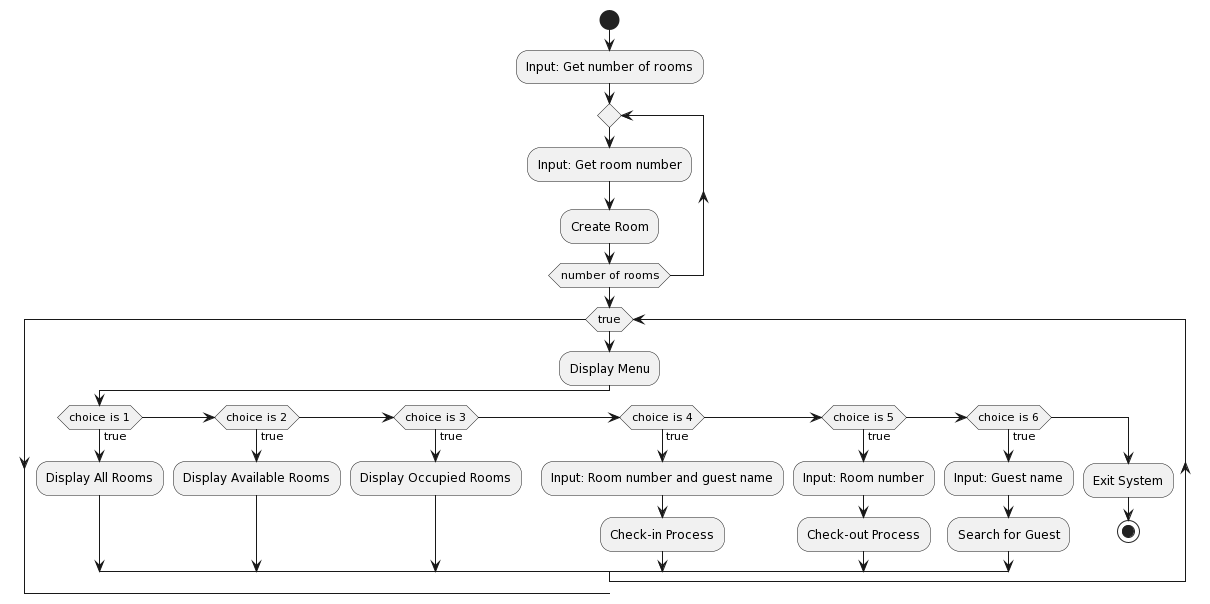
Additionally, the system includes a `Node` class that serves as a building block for a linked list. This class creates nodes containing room objects and references to subsequent nodes.

The main component, the “Hotel Management System” class, manages a collection of rooms utilizing a linked list structure. It provides functionalities like adding rooms to the hotel, displaying all rooms, listing available and occupied rooms, occupying and vacating specific rooms, and searching for guests in the rooms.

The `Main` class orchestrates user interactions through a menu-driven interface, allowing users to interact with the system by inputting their choices. Users can view room status, occupancy details, occupy or vacate rooms, search for guests by name, and exit the program as needed.

Although this system offers a foundational structure for hotel management, it may benefit from enhancements such as input validation for user entries, handling edge cases like attempting to occupy or vacate an already occupied or vacant room, and expanding functionalities to encompass more complex hotel management operations.

**Flow chart:**

****

**PSEUDO CODE:**

**Class Room:**

**Properties:**

**roomNumber: integer**

**IsOccupied: boolean**

**guestsName: string**

**Constructor Room(roomNumber):**

**Initialize roomNumber with provided value**

**Set IsOccupied to false**

**Set guestsName to an empty string**

**Method CheckIn(guestName):**

**Set IsOccupied to true**

**Set guestsName to guestName**

**Method CheckOut():**

**Set IsOccupied to false**

**Clear guestsName**

**Method getStatus():**

**If IsOccupied is true:**

**Return "Occupied by " + guestsName**

**Else:**

**Return "Available"**

**Class Node:**

**Properties:**

**room: Room**

**next: Node**

**Constructor Node(room):**

**Initialize room with provided room**

**Set next to null**

**Class HotelManagementSystem:**

**Properties:**

**start: Node**

**Method addRooms(room):**

**Create a new Node with provided room**

**If start is null:**

**Set start to the new Node**

**Else:**

**Traverse to the end of the linked list and add the new Node**

**Method displayAllTheRooms():**

**Print "All Rooms:"**

**Traverse the linked list:**

**Print roomNumber and its status**

**Method displayAllAvailableRooms():**

**Print "Available Rooms:"**

**Traverse the linked list:**

**If room is not occupied:**

**Print roomNumber**

**Method displayOccupiedRooms():**

**Print "Occupied Rooms:"**

**Traverse the linked list:**

**If room is occupied:**

**Print roomNumber and guestsName**

**Method CheckInRooms(roomNumber, guestName):**

**Traverse the linked list:**

**If roomNumber matches:**

**Check in the guest**

**Print room occupancy status**

**Exit loop**

**Method CheckOutingRooms(roomNumber):**

**Traverse the linked list:**

**If roomNumber matches:**

**Check out the guest**

**Print room vacancy status**

**Exit loop**

**Method searchTheGuest(guestName):**

**Initialize found as false**

**Traverse the linked list:**

**If room is occupied and guestsName matches:**

**Print roomNumber where the guest is staying**

**Set found to true**

**If found is false:**

**Print guest not found message**

**Main:**

**Create a Scanner object s1**

**Create a new HotelManagementSystem object Hotel**

**Get numberOfRooms from user input**

**Loop i from 1 to numberOfRooms:**

**Get roomNumber from user input**

**Add a new Room with roomNumber to Hotel**

**Loop indefinitely:**

**Display menu options**

**Get user choice**

**Perform action based on user choice:**

**Display all rooms**

**Display available rooms**

**Display occupied rooms**

**Check in a guest to a room**

**Check out a guest from a room**

**Search for a guest in rooms**

**Exit the program if requested by the user**

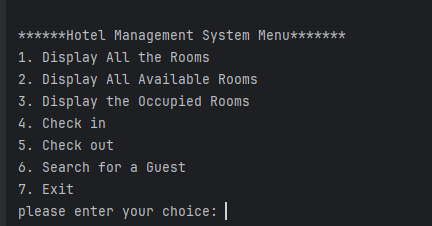
**SOURCE CODE:**

import java.util.Scanner;  
  
class Room {  
 int roomNumber;  
 boolean IsOccupied;  
 String guestsName;  
  
 public Room(int roomNumber) {  
 this.roomNumber = roomNumber;  
 this.IsOccupied = false;  
 this.guestsName = "";  
 }  
  
 public void CheckIn(String guestName) {  
 this.IsOccupied = true;  
 this.guestsName = guestName;  
 }  
  
 public void CheckOut() {  
 this.IsOccupied = false;  
 this.guestsName = "";  
 }  
  
 public String getStatus() {  
 return IsOccupied ? "Occupied by " + guestsName : "Available";  
 }  
}  
  
class Node {  
 Room room;  
 Node next;  
  
 public Node(Room room) {  
 this.room = room;  
 this.next = null;  
 }  
}  
public class HotelManagementSystem { // hotel management class for performing multiplr task  
  
 Node start;  
  
 public void addRooms(Room room) {  
 Node newNode = new Node(room);  
 if (start == null) {  
 start = newNode;  
 } else {  
 Node current = start;  
 while (current.next != null) {  
 current = current.next;  
 }  
 current.next = newNode;  
 }  
 }  
  
 public void displayAllTheRooms() { //method for displaying all the rooms doesn't matter available or not  
 System.*out*.println("All Rooms:");  
 Node current = start;  
 while (current != null) {  
 System.*out*.println("Room " + current.room.roomNumber + " - " + current.room.getStatus());  
 current = current.next;  
 }  
 }  
  
 public void displayAllAvailableRooms() {//method for displaying all the available rooms  
 System.*out*.println("Available Rooms:");  
 Node current = start;  
 while (current != null) {  
 if (!current.room.IsOccupied) {  
 System.*out*.println("Room " + current.room.roomNumber);  
 }  
 current = current.next;  
 }  
  
 System.*out*.println("there is no room available");  
  
 }  
  
 public void displayOccupiedRooms() { //method for displaying only occupied room  
 System.*out*.println("Occupied Rooms:");  
 Node current = start;  
 while (current != null) {  
 if (current.room.IsOccupied) {  
 System.*out*.println("Room " + current.room.roomNumber + " - Occupied by " + current.room.guestsName);  
 }  
 current = current.next;  
 }  
 System.*out*.println("there is no room occupied");  
 }  
  
 public void CheckInRooms(int roomNumber, String guestName) { //method for occupying a room  
 Node current = start;  
 while (current != null) {  
 if (current.room.roomNumber == roomNumber) {  
 current.room.CheckIn(guestName);  
 System.*out*.println("Room " + roomNumber + " has been occupied by " + guestName);  
 return;  
 }  
 current = current.next;  
 }  
 System.*out*.println("Room " + roomNumber + " not found or already occupied.");  
 }  
  
 public void CheckOutingRooms(int roomNumber) { //method to vacate a room  
 Node current = start;  
 while (current != null) {  
 if (current.room.roomNumber == roomNumber) {  
 current.room.CheckOut();  
 System.*out*.println("Room " + roomNumber + " has been vacated.");  
 return;  
 }  
 current = current.next;  
 }  
 System.*out*.println("Room " + roomNumber + " not found or already vacant.");  
 }  
  
 public void searchTheGuest(String guestName) { // method for searching the guest  
 boolean found = false;  
 Node current = start;  
 while (current != null) {  
 if (current.room.IsOccupied && current.room.guestsName.equalsIgnoreCase(guestName)) {  
 found = true;  
 System.*out*.println(guestName + " is staying in Room " + current.room.roomNumber);  
 }  
 current = current.next;  
 }  
 if (!found) {  
 System.*out*.println("sorry" + guestName + " is not found in any room.");  
 }  
 }  
}

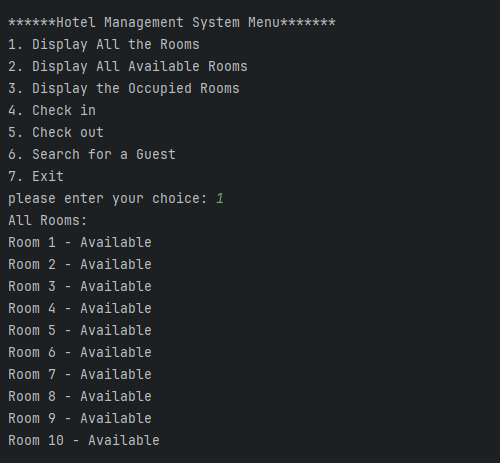
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner s1 = new Scanner(System.*in*);  
 HotelManagementSystem Hotel = new HotelManagementSystem();  
  
 System.*out*.print("Enter the number of rooms in the hotel: ");  
 int numberOfRooms = s1.nextInt();  
  
 for (int i = 1; i <= numberOfRooms; i++) {  
 System.*out*.print("Enter the room number " + i + ": ");  
 int roomNumbers = s1.nextInt();  
 Hotel.addRooms(new Room(roomNumbers));  
 }  
  
 while (true) {  
 System.*out*.println("\n\*\*\*\*\*\*Hotel Management System Menu\*\*\*\*\*\*\*");  
 System.*out*.println("1. Display All the Rooms");  
 System.*out*.println("2. Display All Available Rooms");  
 System.*out*.println("3. Display the Occupied Rooms");  
 System.*out*.println("4. Check in");  
 System.*out*.println("5. Check out");  
 System.*out*.println("6. Search for a Guest");  
 System.*out*.println("7. Exit");  
  
 System.*out*.print("please enter your choice: ");  
 int choice = s1.nextInt();  
  
 switch (choice) {  
 case 1:  
 Hotel.displayAllTheRooms();  
 break;  
 case 2:  
 Hotel.displayAllAvailableRooms();  
 break;  
 case 3:  
 Hotel.displayOccupiedRooms();  
 break;  
 case 4:  
 System.*out*.print("please enter room number to occupy: ");  
 int occupyRoomNumber = s1.nextInt();  
 s1.nextLine(); // Consume newline  
 System.*out*.print("please enter guest name: ");  
 String guestName = s1.nextLine();  
 Hotel.CheckInRooms(occupyRoomNumber, guestName);  
 break;  
 case 5:  
 System.*out*.print("please enter the room number to vacate: ");  
 int vacateRoomNumber = s1.nextInt();  
 Hotel.CheckOutingRooms(vacateRoomNumber);  
 break;  
 case 6:  
 s1.nextLine(); // Consume newline  
 System.*out*.print("please enter guest name to search: ");  
 String searchGuestName = s1.nextLine();  
 Hotel.searchTheGuest(searchGuestName);  
 break;  
 case 7:  
 System.*out*.println("Exiting...");  
 s1.close();  
 System.*exit*(0);  
 break;  
 default:  
 System.*out*.println("Invalid choice enter a valid option between(1 to 6)");  
 }  
 }  
  
 }  
}

**Snippets:**

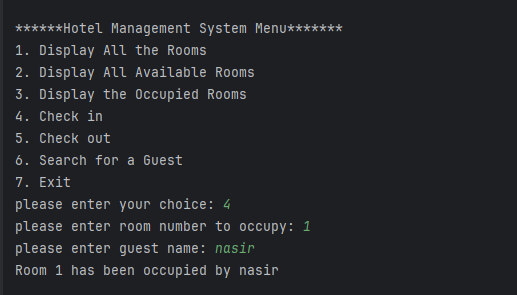
**1**



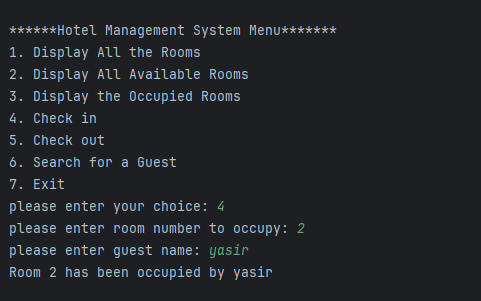
**2**



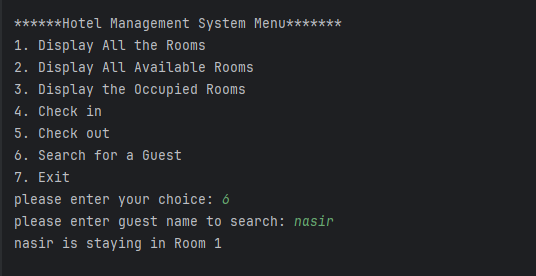
**3**



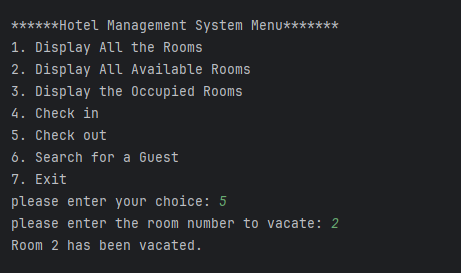
**4**



**5**



**6**



**7**

