CA 3: Experiential Learning

Group Members:

| Sr. No. | PRN | Name of Student | Mail ID |
| --- | --- | --- | --- |
| 1 | 22070122206 | Shruti Bist | shruti.bist.btech2022@sitpune.edu.in |
| 2 | 22070122216 | Soumili Biswas | soumili.biswas.btech2022@sitpune.edu.in |
| 3 | 22070122219 | Srishti Parulekar | srishti.parulekar.btech2022@sitpune.edu.in |

**Problem Statement**: Developing an efficient Hotel Management System for reservation, billing and invoicing purposes.

**Introduction:**

Our project is a user-friendly hotel management system designed to streamline room bookings, billing, and data management. Here's how users can interact with the system:

**Features:**

**Room Selection**: Users can choose from various room types (Classic, Deluxe, and Suite) on different floors, enabling personalized room selection.

**Booking**: Users can book rooms by specifying occupancy, room type, floor, and room number. The system automates room allocation and calculates bills based on room type and stay duration.

**Billing**: The system calculates total bills for guests per room, enhancing financial operations and customer service.

**Search Room Details**: Users can access room details, including status, occupants, check-in/check-out dates, and total charge presented in an organized tabular format.

**Room Cancellation**: Users can cancel their current bookings in case of emergency checkout.

**Data Persistence**: The code uses file handling to ensure data retention across program runs, initializing room data and saving details to a CSV file.

**User Interaction:** The main function features a user interaction loop, allowing users to book rooms, check billing, and continue the process efficiently.

**Object Oriented Programming Concepts Used:**

**Encapsulation:**

- ‘**Room**’ class members : *‘floor\_no`, ‘room\_no`, ‘status`, ‘occupants`, ‘checkin\_date`, ‘checkout\_date` and ‘total\_bill`* are encapsulated within the class, ensuring data privacy and access control.

**Inheritance:**

- The classes ‘*ClassicRoom’, ‘DeluxeRoom*’ and ‘*SuiteRoom’* inherit from the base ‘*Room*’ class. They specialize in room types, demonstrating the concept of inheritance.

- In this code, the base class is Room, and there are three derived classes: ClassicRoom, DeluxeRoom, and SuiteRoom. Each of these derived classes inherits from the single base class Room. This is an example of *Single Inheritance.*

**Polymorphism (Method Overriding):**

- The *‘calculateTotalBill’* function in the base ‘*Room*’ class is declared as a virtual function, enabling dynamic method binding.

- The derived classes (*‘ClassicRoom’, ‘DeluxeRoom’, ‘SuiteRoom’*) override this function to provide specialized implementations, showcasing polymorphism.

**Composition**:

- The ‘*Hotel*’ class manages a map of rooms (*‘rooms*’), demonstrating composition. It represents the relationship between a hotel and its constituent rooms.

**Dynamic Memory Allocation:**

- In the *‘initializeCSVFile*’ method, dynamic memory allocation using the ‘*new*’ operator is used to create room objects at runtime, based on the specified room types and floors.

**Friend Class:**

- The ‘*Hotel*’ class is declared as a friend class of the ‘*Room*’ class, allowing it to access and manipulate private members of *‘Room’*. This promotes encapsulation while enabling the hotel to manage room data effectively.

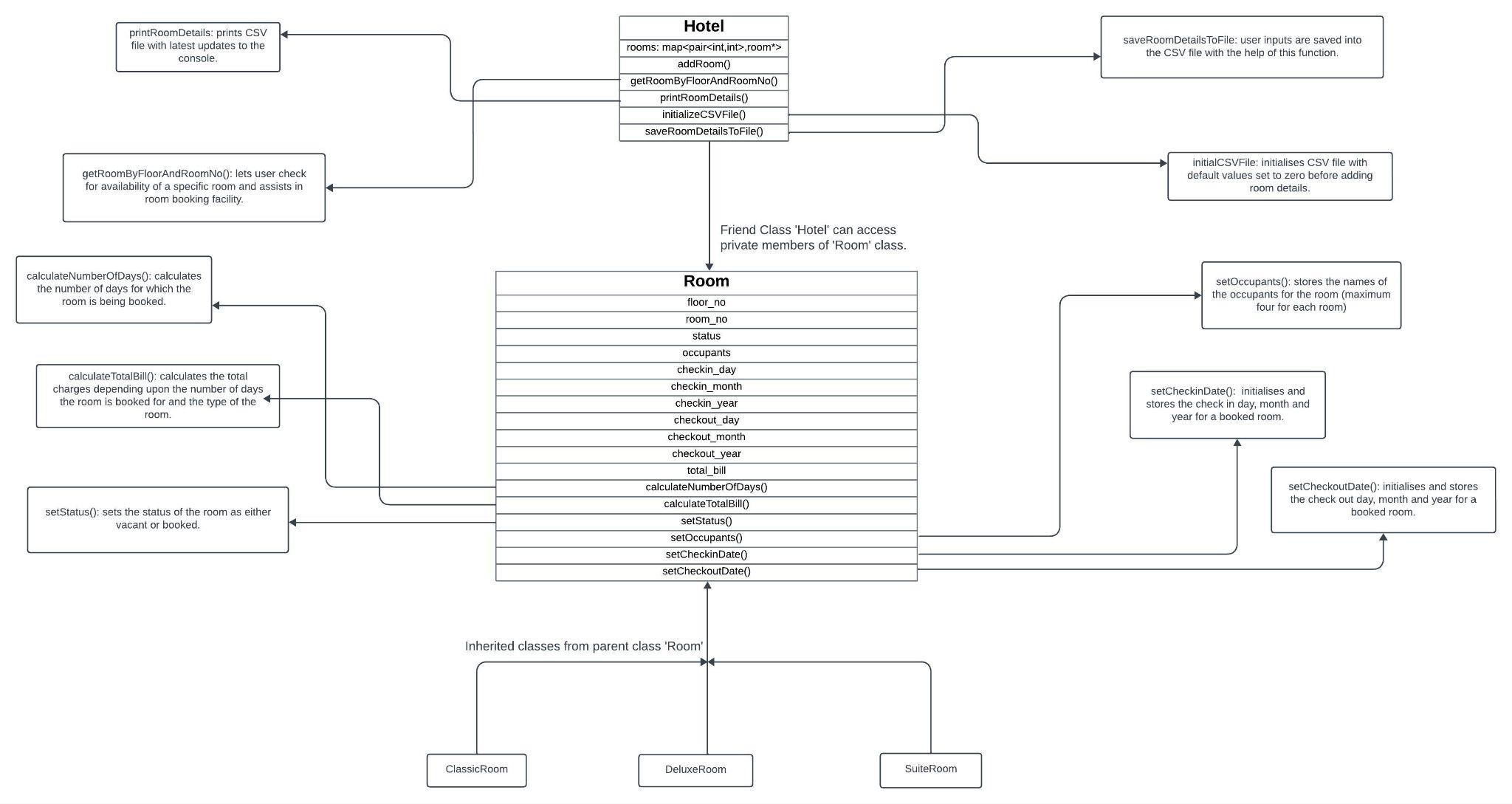
**Abstraction:**

- Abstraction in this code is primarily achieved through class definitions, providing simplified representations of real-world entities, hiding complex internal details.

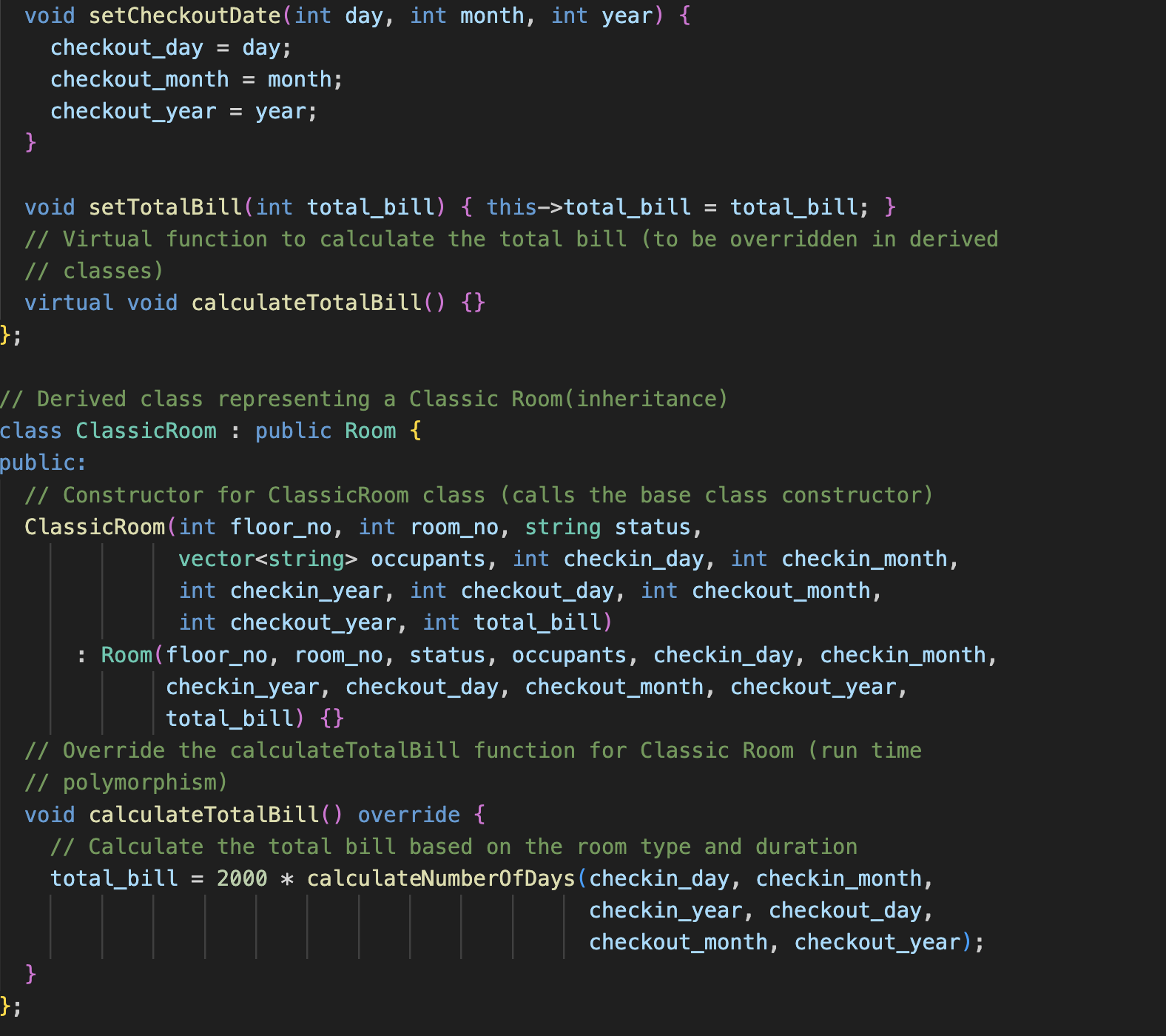
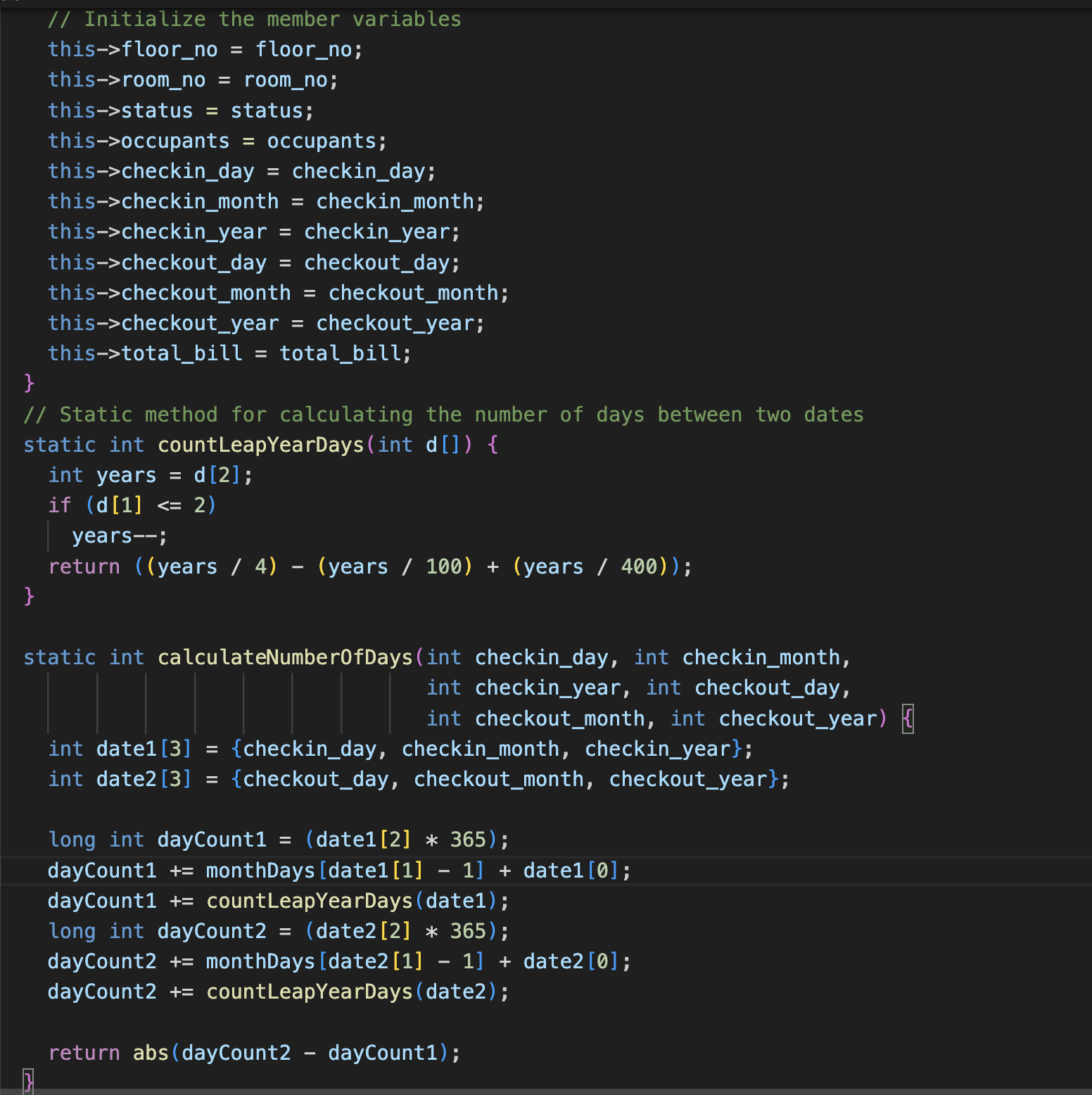
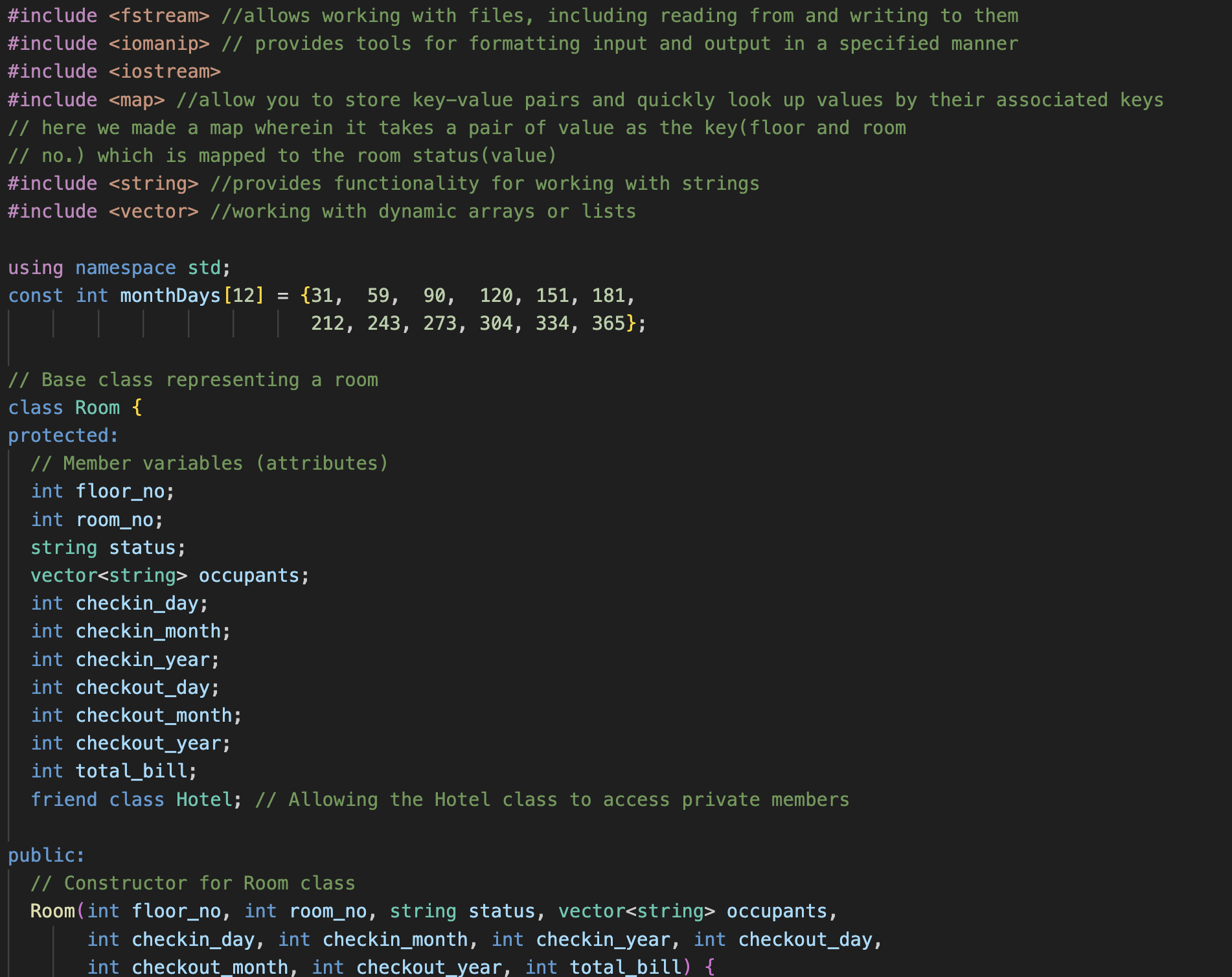
**Constructors:**

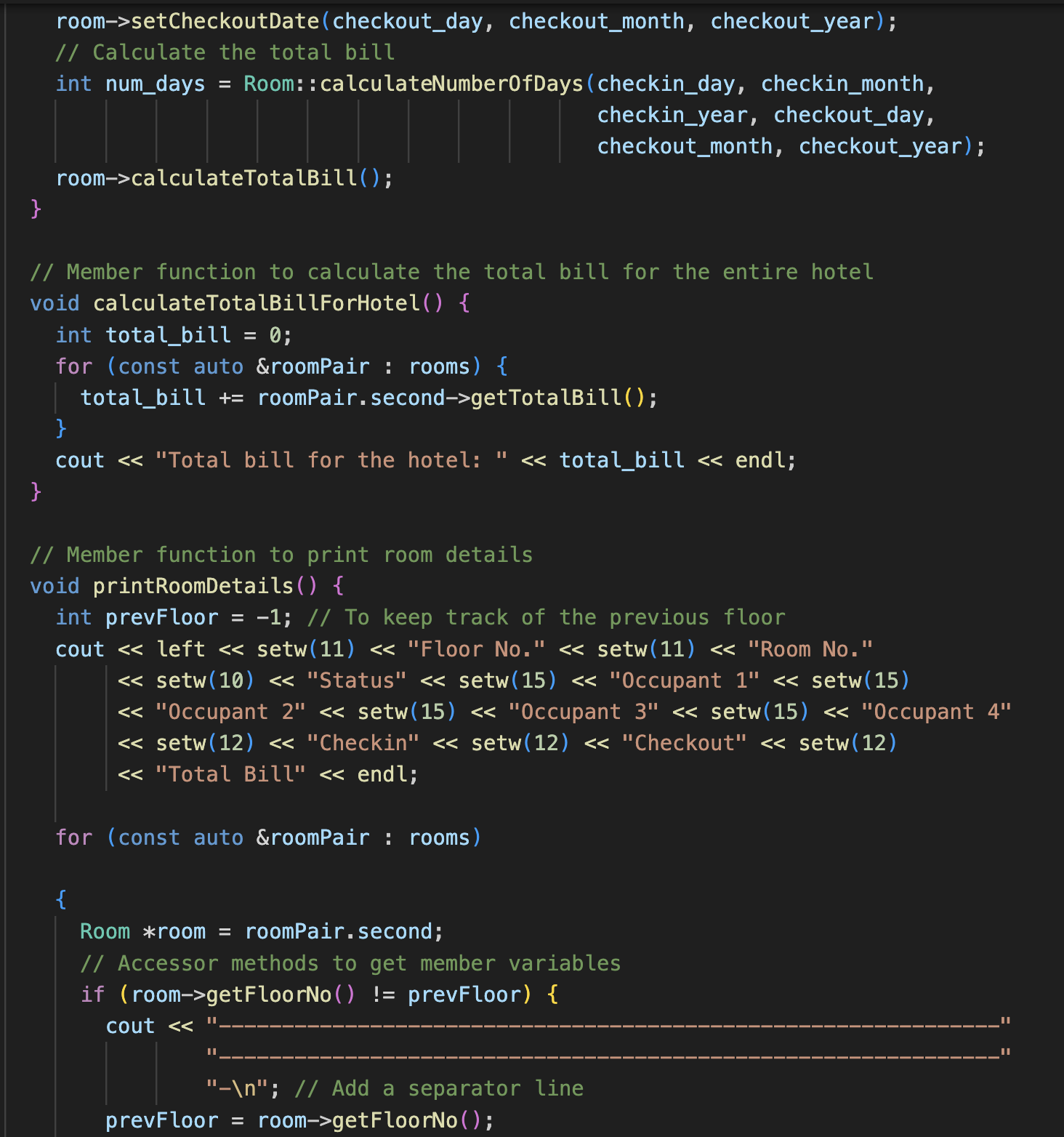
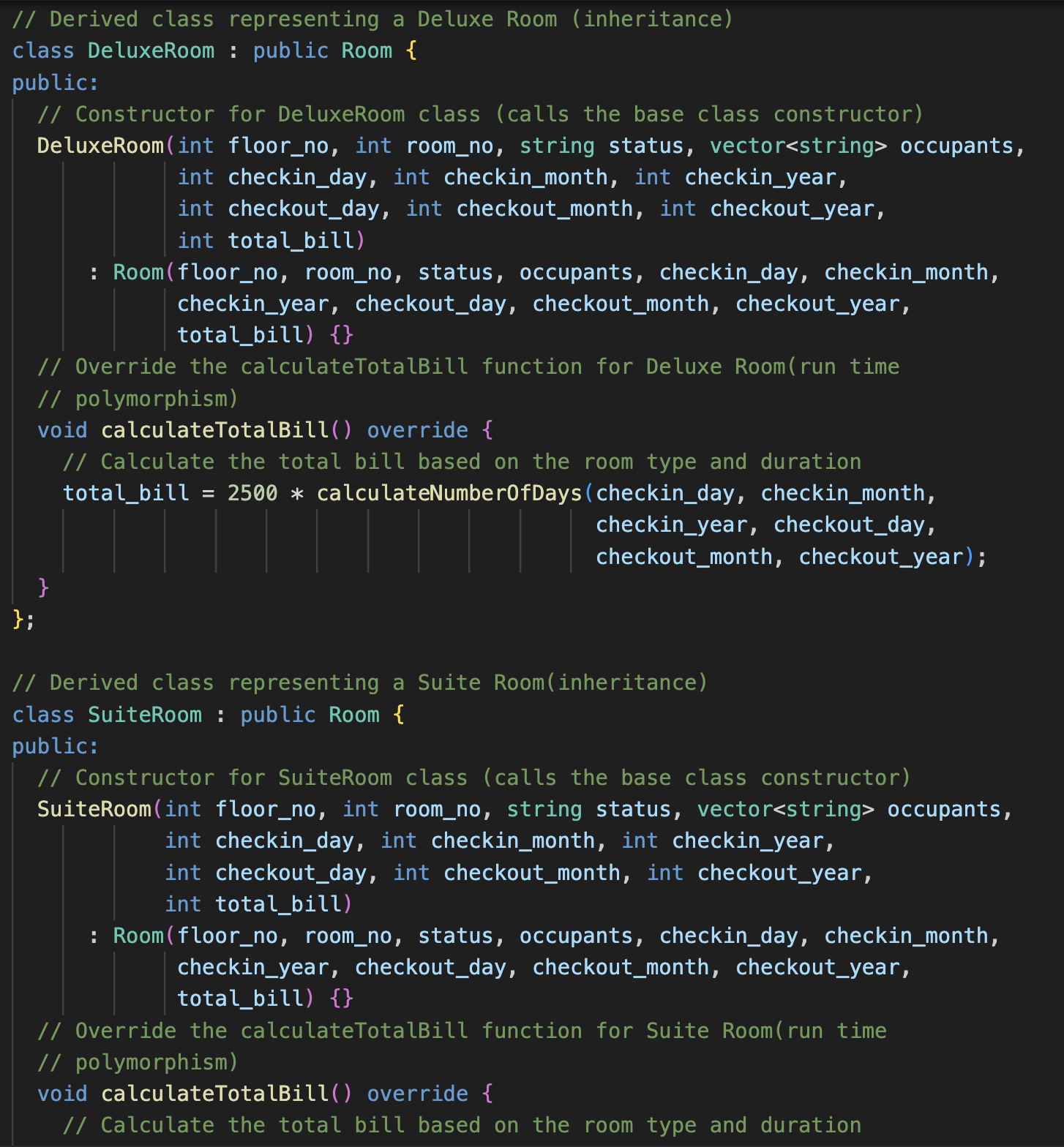
- Constructors are used in the ‘*Room*’ class to initialize room objects and in derived classes (‘*ClassicRoom’, ‘DeluxeRoom’, ‘SuiteRoom*’) to set their initial attributes. Additionally, constructors are indirectly employed during dynamic memory allocation when initializing room data in the ‘*initializeCSVFile*’ method.

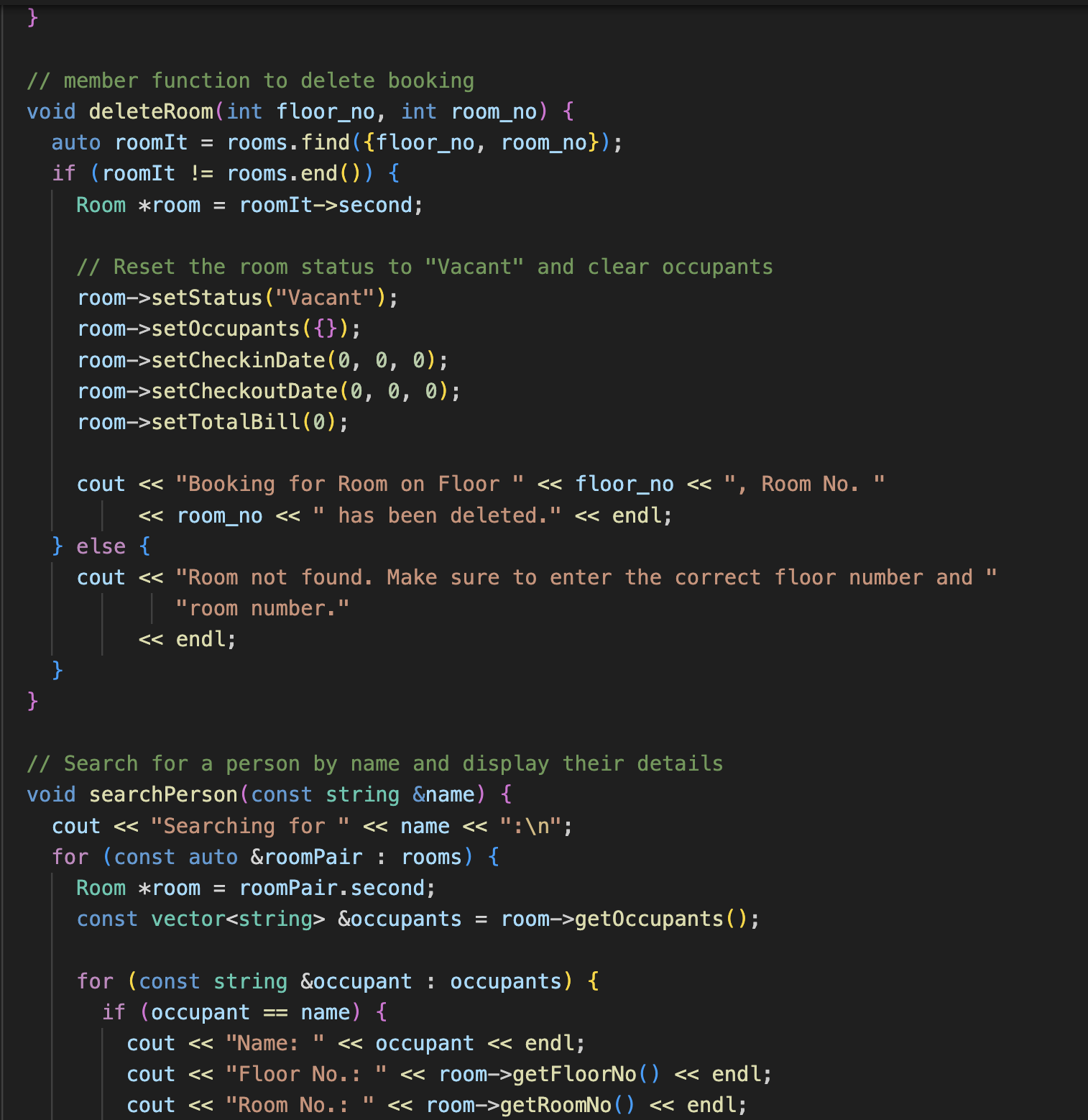
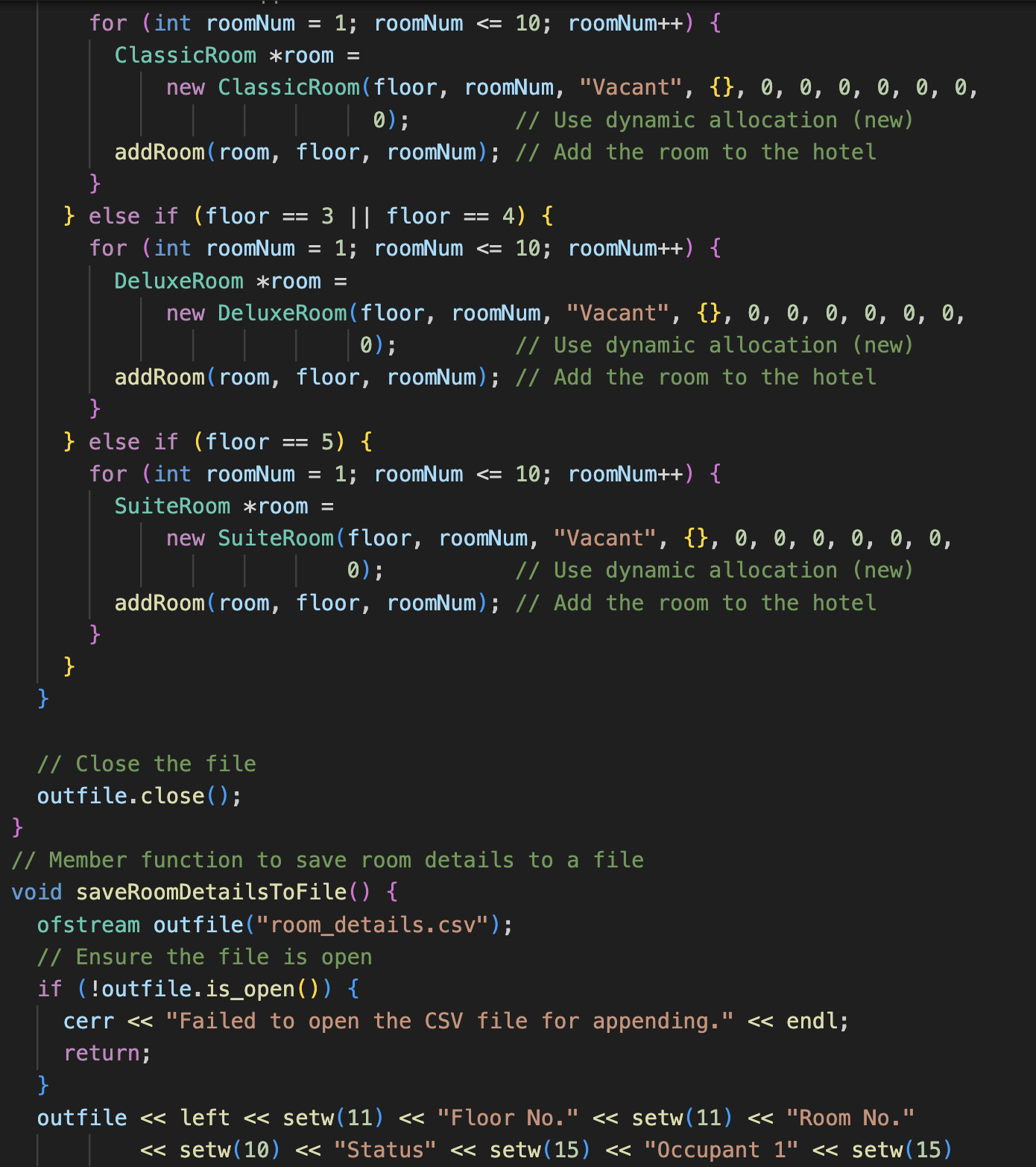
**Class Diagram:**

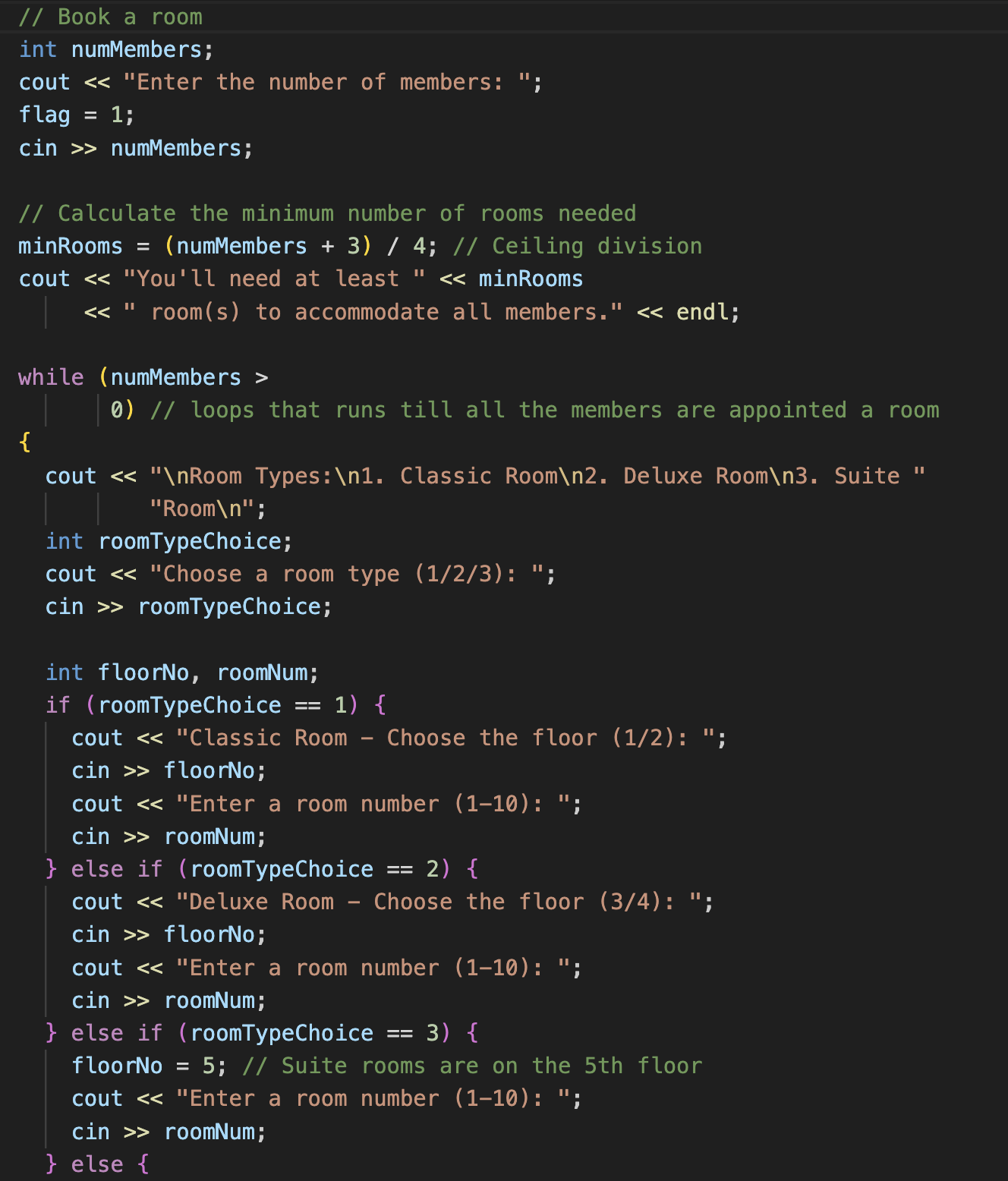


**Code snippets:**

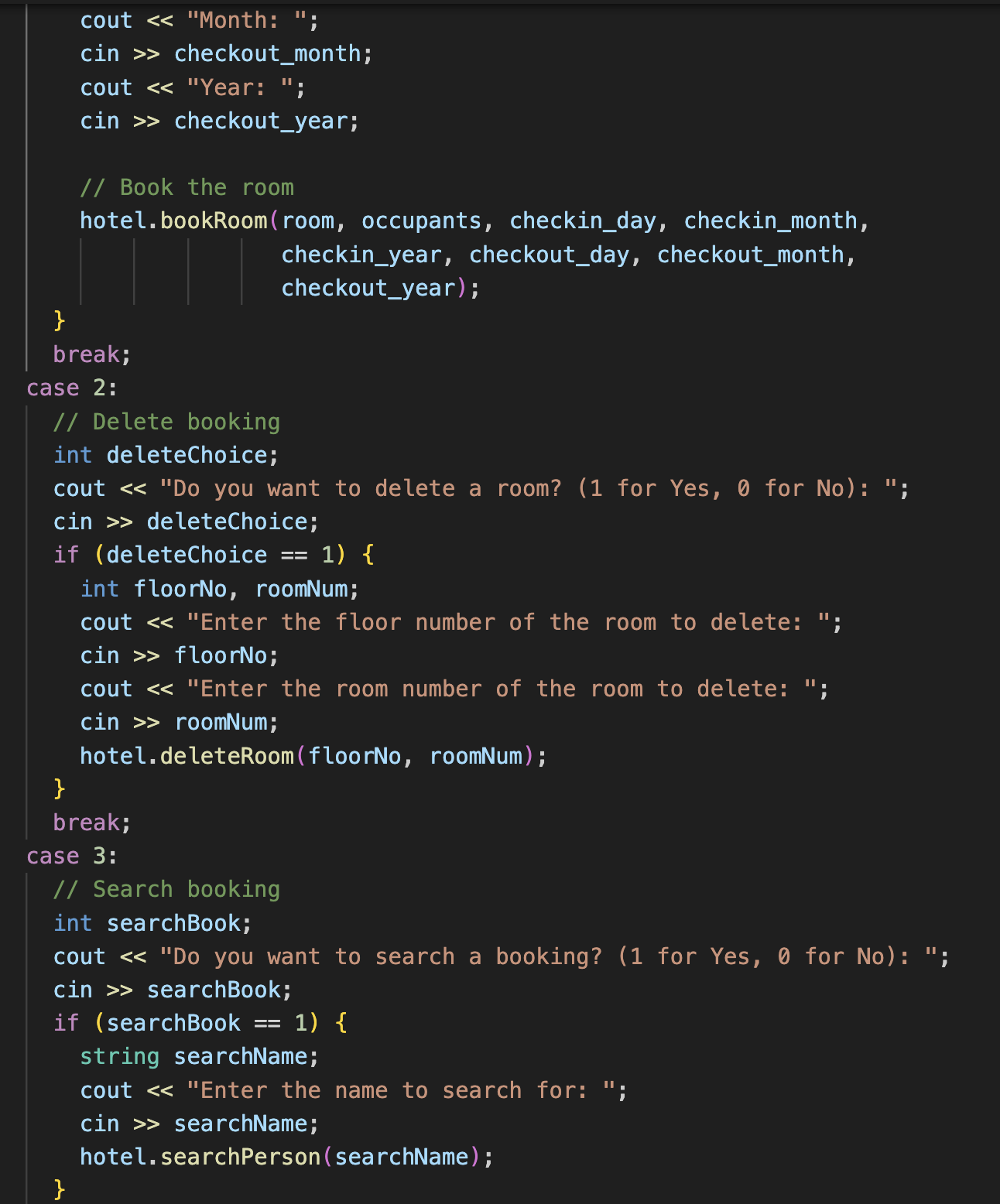


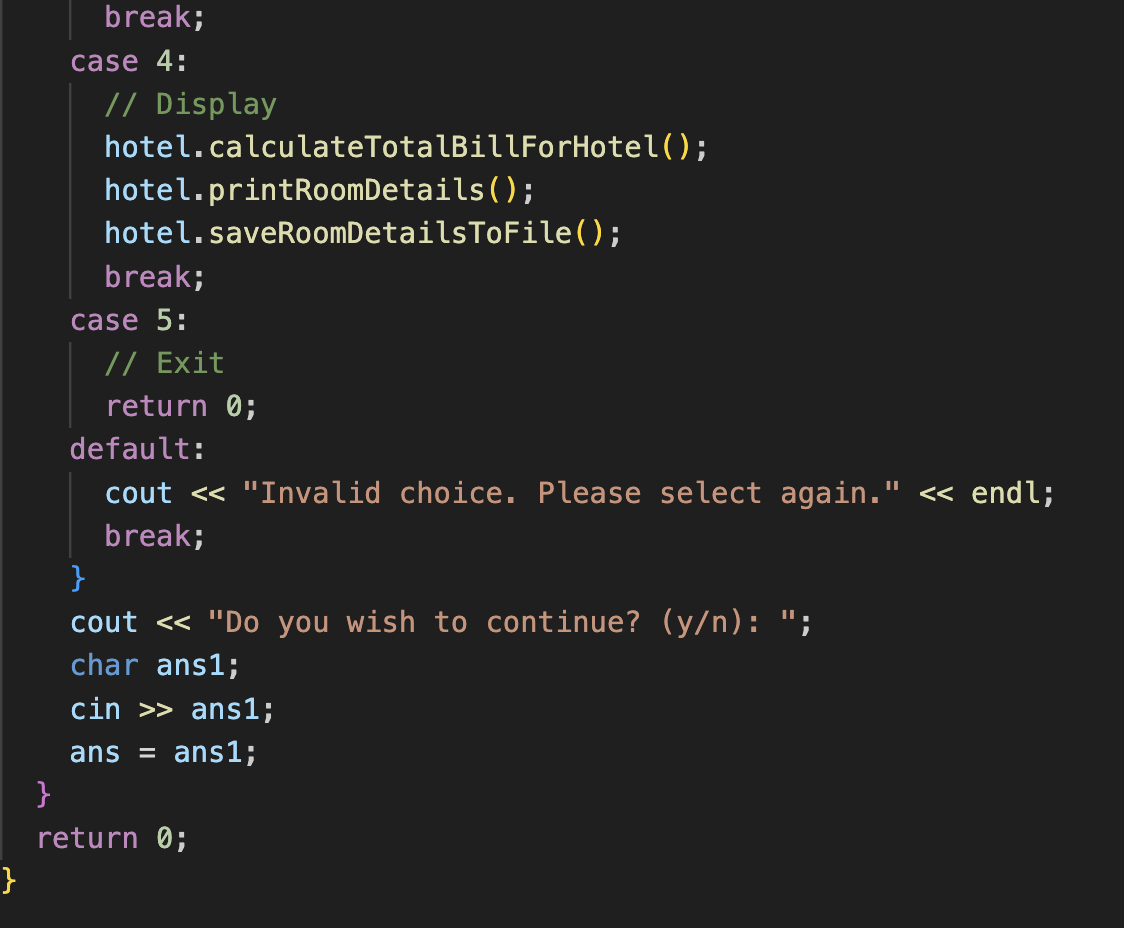






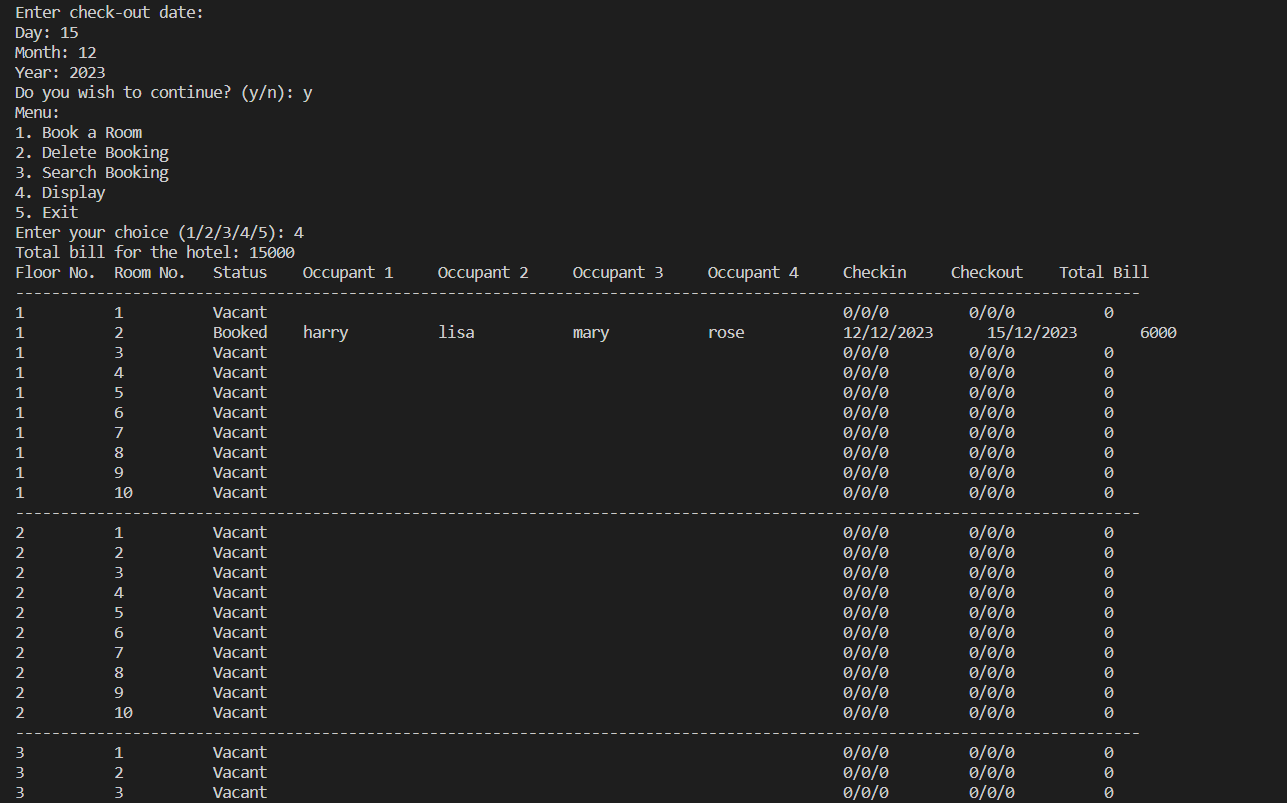
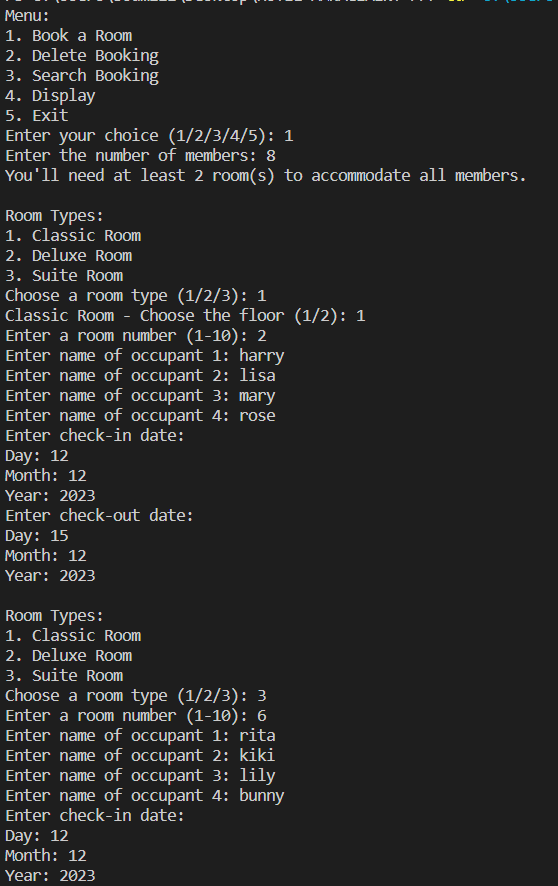




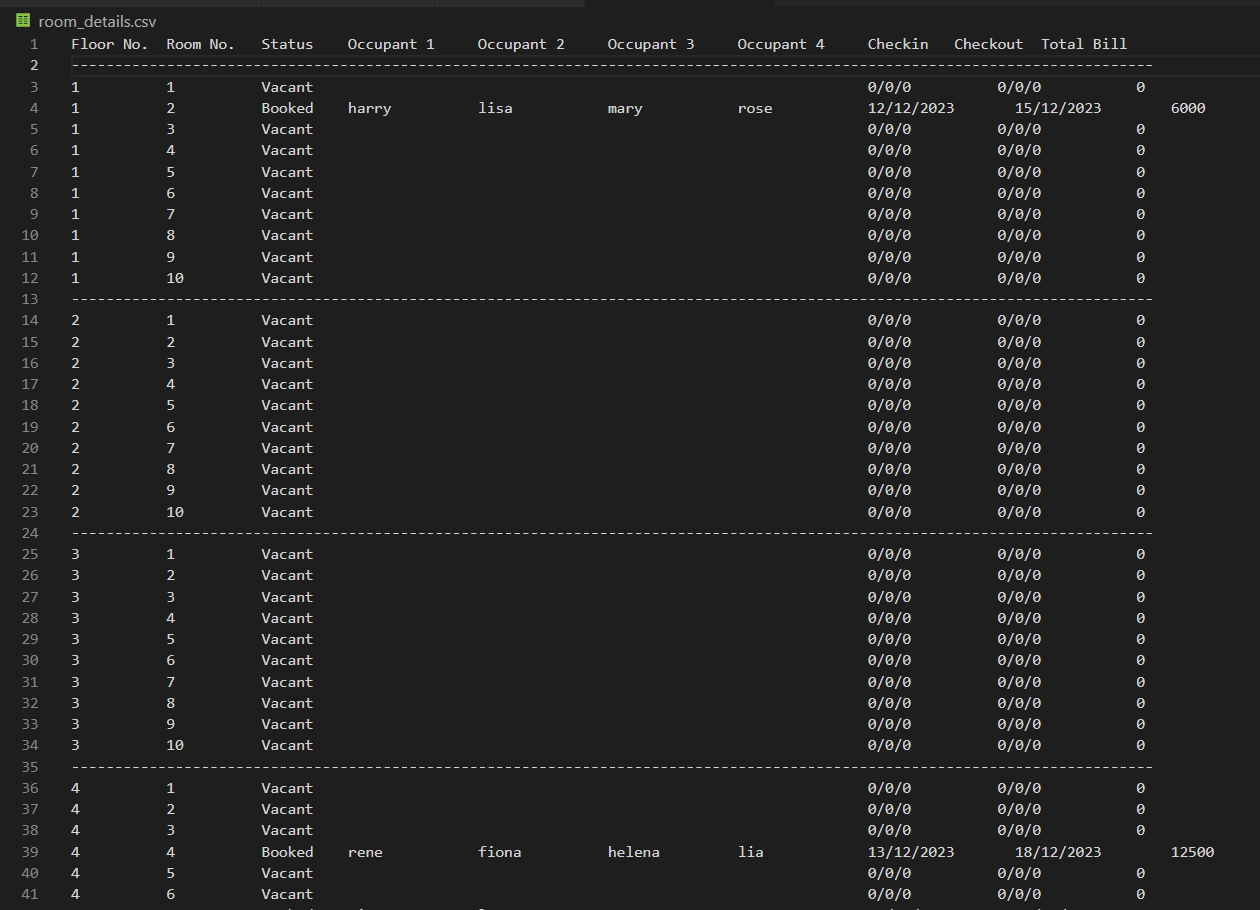


**Input/Output:**

The menu is displayed initially and the user books a room for eight people. Therefore, a minimum of 2 rooms is needed.

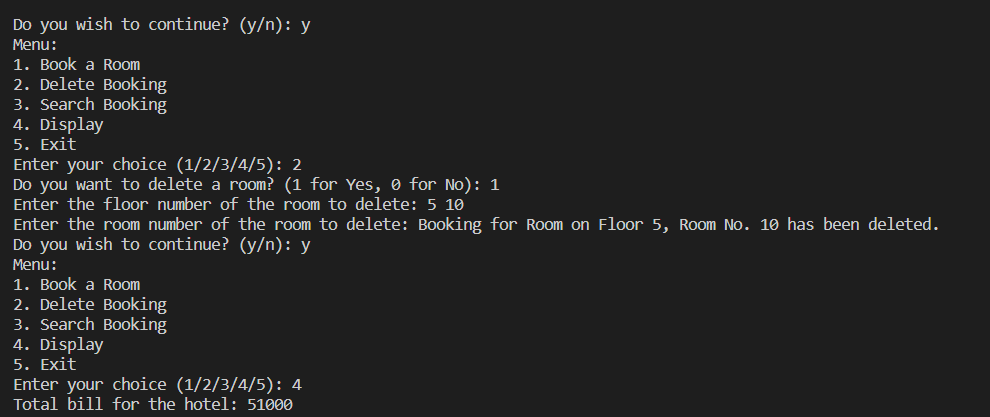


A database is created and an output is displayed in the CSV file.

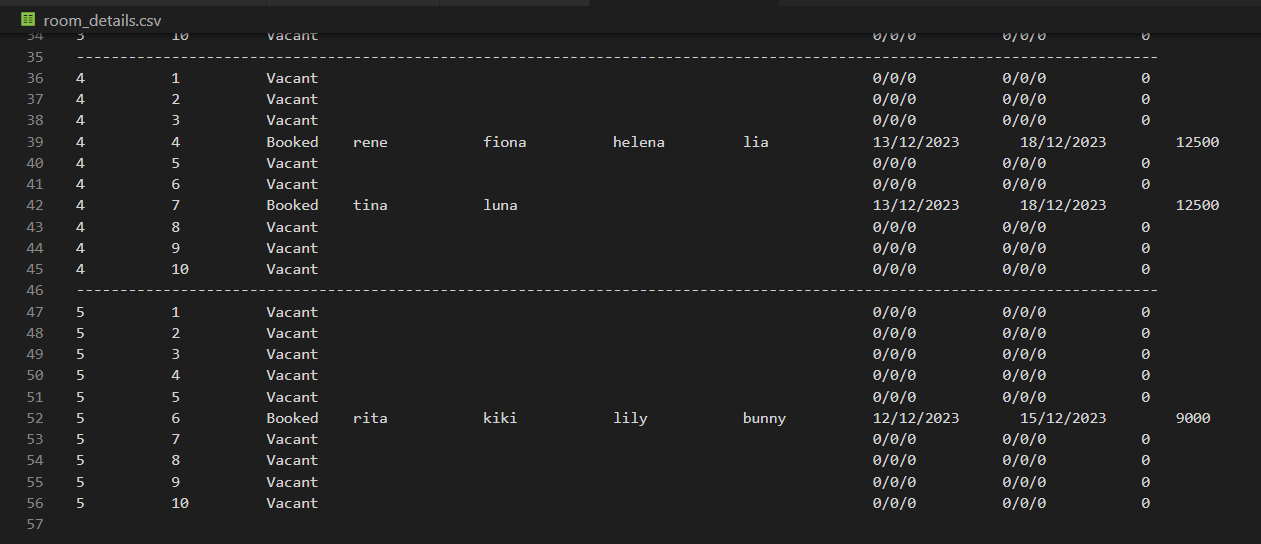


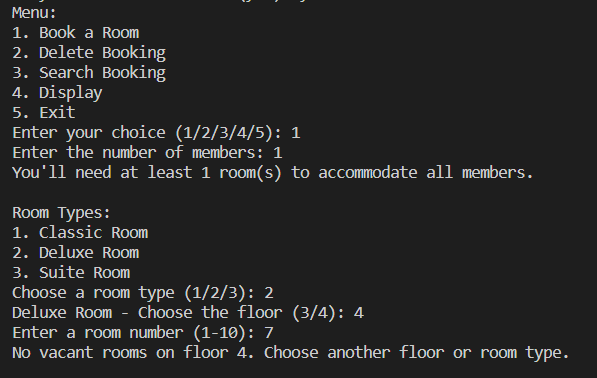
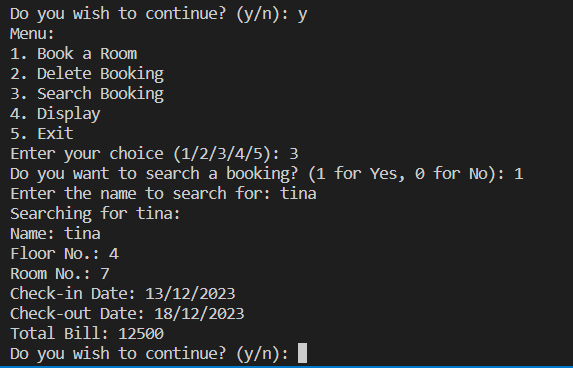
If the user wishes to delete a room booking ( here room number 10 on the fifth floor) :





Thus, the 10th room on the 5th floor gets deleted.



Searching for a booked room: Showing no vacancy for an already booked room: 

Github repository link:

<https://github.com/soumili-03/HOTEL-MANAGEMENT-SYSTEM>