**Azure Horizon Resort - Hotel Booking**

Azure Horizon Resort is a renowned hotel booking service provider. They are in need of a system to manage hotel reservations and calculate booking costs. As their software consultant, you've been tasked with developing a software system to fulfil their requirements.

Provided a **Reservation** class with the below**private attributes** as a part of the code skeleton.

|  |  |
| --- | --- |
| **Attributes** | **DataType** |
| customerName | String |
| phoneNumber | Long |
| checkInDate | java.util.Date |
| checkOutDate | java.util.Date |
| numOfGuests | int |
| roomType | String |

**Getter and setter** methods for all the above attributes have been provided as part of the code skeleton.

**Reservation** class should be configured using the necessary annotations.

Provided **Hotel** class with the below **private attributes** as a part of code skeleton

|  |  |
| --- | --- |
| **Attributes** | **DataType** |
| hotelName | String |
| location | String |
| roomRates | Map<String, Double> |

**Getter and setter** methods for all the above attributes have been provided as a part of code skeleton.

**Hotel** class should be configured using the necessary annotations.

The values for the **roomRates** attribute should be injected via setter-based injection. The keys represent room types, and the values represent the corresponding room rates per night.

The **hotelName** and **location** values are given in the **details.properties** file.

|  |  |
| --- | --- |
| **Attribute Name** | **Value** |
| hotelName | Azure Horizon Resort HOTELS |
| location | Chennai |

Also the **roomRates** map values are given in the **details.properties** file with the following room types and rates:

|  |  |
| --- | --- |
| **Key - Room Type(String)** | **Value- cost(Double)** |
| Standard | 100.0 |
| Deluxe | 150.0 |
| Suite | 200.0 |

**Note: Only for the above hotel. The keys are case sensitive.**

**Main.java: main(String[] arg):void**

* Using ApplicationContext, get **ReservationService** bean and Hotel bean.
* Invoke the **bookReservation(reservation)** method using the **ReservationService** bean.

**Technical Specifications:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Method Name** | **Input** | **Output** | **Exception** |
| **ReservationService** | bookReservation | Reservation reservation | void | ParseException |
| calculateBookingCost | String roomType, Date checkInDate, Date checkOutDate | double |  |
| checkAvailability | Date checkInDate, Date checkOutDate | boolean |  |

**Business Rules & Validations:**

In **ReservationService** class includes the following private attribute and **inject via field based injection**.

**private Hotel hotel;**

**Getter and setter** methods for the above attribute has been provided as a part of code skeleton

**Service 1**:**bookReservation(reservation): void**

This method should be used to get reservation information from the customer and calculate the booking cost based on the room type and duration of stay.

* Inject the Reservation bean using the application context.
* Retrieve reservation information such as customerName, phoneNumber, checkInDate, checkOutDate, numOfGuests, and roomType.
* Calculate the booking cost based on the selected roomType, duration of stay, and rates obtained from the Hotel bean.

**Service 2:  calculateBookingCost(roomType: String, checkInDate: Date, checkOutDate: Date): void**

This method should calculate the booking cost based on the provided room type, check-in date, and check-out date.

* Invoke **checkAvailability(checkInDate, checkOutDate)**.
* If the room is available, calculate the total cost by multiplying the room rate by the number of nights stayed.
* If the room is not available, display an error message indicating unavailability.
* If the roomType is not available, display an error message indicating invalid room type.

**Service 3**: **checkAvailability(checkInDate: Date, checkOutDate: Date): boolean**

This method should check the availability of rooms for the given check-in and check-out dates.

* Check if the check-in date before the check-out date conflicts with the provided dates.
* If there are no conflicts, return true, indicating availability. Otherwise, return false.

**Limitations and Constraints:**

* The **Reservation** and **Hotel** classes are in the **com.model** package.
* The **ReservationService** class should be in the **com.service** package.
* The **ApplicationConfig** class should be in the **com.hotel** package.
* All classes mentioned above should be configured using the necessary annotations.
* The **Hotel** bean should be declared as an inner bean in the service class and injected using autowiring inside the **ReservationService** class.
* Don’t alter the code given in the **ConversionConfig** class.

**Sample Input and Output 1:**

Enter the Customer Name

**Antony Prakash**

Enter the Phone Number

**9000000**

Enter the Check-In Date <dd-MM-yyyy>

**27-12-2023**

Enter the Check-Out Date <dd-MM-yyyy>

**28-12-2023**

Enter the Total Number of Guests

**2**

Enter the Room Type

**Deluxe**

Total Booking Cost is: $150.0

**Sample Input and Output 2:**

Enter the Customer Name

**Renita Colette**

Enter the Phone Number

**9000000**

Enter the Check-In Date <dd-MM-yyyy>

**28-12-2023**

Enter the Check-Out Date <dd-MM-yyyy>

**27-12-2023**

Enter the Total Number of Guests

**2**

Enter the Room Type

**Deluxe**

Invalid Details

**Sample Input and Output 3:**

Enter the Customer Name

**Renita Colette**

Enter the Phone Number

**9000000**

Enter the Check-In Date <dd-MM-yyyy>

**29-12-2023**

Enter the Check-Out Date <dd-MM-yyyy>

**30-12-2023**

Enter the Total Number of Guests

**2**

Enter the Room Type

**King**

Invalid Details