Hotel Reservation System

Contents

Objective: 1
Problem Description: 1
Key Components: 2
1. Base Class: `Room` 2
2. Derived Classes: 2
3. Reservation Class: 3
4. Payment Class (with Polymorphism): 3
5. Customer Class: 3
6. Hotel Management Class: 4
7. Exception Handling: 4
8. Hotel Reservation System Launcher: 4
Example Usage: 5

Objective:

Design and implement a Hotel Reservation System that allows customers to book rooms, manage reservations, and handle payment, using object-oriented principles such as classes, inheritance, polymorphism, and exception handling.

Problem Description:

The Hotel Reservation System will manage hotel rooms, customer reservations, and payment processing. The system will be designed using object-oriented principles, with a focus on the use of classes and inheritance to represent different types of rooms and reservations, polymorphism to handle different payment methods, and exception handling to manage errors and invalid operations.

Key Components:

1. Base Class: `Room`

- Attributes:
  - `roomNumber`: A unique identifier for the room.
  - `capacity`: The number of people the room can accommodate.
  - `pricePerNight`: The price of the room per night.
  - `isAvailable`: A boolean flag indicating whether the room is available for booking.
- Methods:
  - `bookRoom()`: Marks the room as booked.
  - `releaseRoom()`: Marks the room as available.
  - `displayRoomInfo()`: Displays the room's details.
- Description: The `Room` class serves as a base class representing any room in the hotel. It provides common attributes and methods that can be inherited by more specific room classes.

2. Derived Classes:

- `SingleRoom` Class (inherits from `Room`):
  - Additional Attributes:
    - `bedType`: The type of bed in the room (e.g., single, queen).
  - Additional Methods:
    - `displayRoomInfo()`: Overrides the base method to include specific details about the bed type.
  - Description: The `SingleRoom` class extends `Room` and includes attributes specific to a single room, utilizing polymorphism to override the display method.

- `DoubleRoom` Class (inherits from `Room`):
  - Additional Attributes:
    - `bedType`: The type of beds in the room (e.g., two twins, one king).
  - Additional Methods:
    - `displayRoomInfo()`: Overrides the base method to include specific details about the bed types.
  - Description: The `DoubleRoom` class extends `Room` and includes attributes specific to a double room, demonstrating polymorphism in method overriding.

- `SuiteRoom` Class (inherits from `Room`):
  - Additional Attributes:
    - `hasLivingArea`: A boolean flag indicating if the suite has a separate living area.
    - `numberOfRooms`: The number of rooms in the suite.
  - Additional Methods:
    - `displayRoomInfo()`: Overrides the base method to include details about the living area and the number of rooms.
  - Description: The `SuiteRoom` class extends `Room` and includes attributes specific to a suite room, showcasing polymorphism in method overriding.

3. Reservation Class:

- Attributes:
  - `reservationID`: A unique identifier for the reservation.
  - `room`: A reference to the `Room` object being reserved.
  - `customer`: A reference to the `Customer` making the reservation.
  - `checkInDate`: The check-in date for the reservation.
  - `checkOutDate`: The check-out date for the reservation.
  - `totalAmount`: The total cost of the reservation.
- Methods:
  - `calculateTotalAmount()`: Calculates the total cost based on the room type and duration of stay.
  - `displayReservationDetails()`: Displays the details of the reservation.
- Description: The `Reservation` class manages the details of a reservation, including the room being reserved, the customer, and the duration of stay. It interacts with the `Room` and `Customer` classes.

4. Payment Class (with Polymorphism):

- Attributes:
  - `amount`: The amount to be paid.
- Methods:
  - `processPayment()`: Abstract method to be implemented by derived classes.
- Derived Classes:
  - `CreditCardPayment` Class (inherits from `Payment`):
    - Additional Attributes:
      - `cardNumber`: The credit card number.
      - `cardHolderName`: The name on the credit card.
      - `expiryDate`: The expiry date of the credit card.
    - Methods:
      - `processPayment()`: Implements the payment processing logic for credit cards.
    - Description: The `CreditCardPayment` class extends `Payment` and implements the specific logic for processing credit card payments.
  - `PayPalPayment` Class (inherits from `Payment`):
    - Additional Attributes:
      - `email`: The email address associated with the PayPal account.
    - Methods:
      - `processPayment()`: Implements the payment processing logic for PayPal.
    - Description: The `PayPalPayment` class extends `Payment` and implements the specific logic for processing payments through PayPal.
- Description: The `Payment` class and its derived classes use polymorphism to handle different payment methods, providing a flexible way to process payments.

5. Customer Class:

- Attributes:
  - `customerID`: A unique identifier for the customer.

- `name`: The name of the customer.
- `email`: The email address of the customer.
- `reservations`: A list of reservations made by the customer.

- Methods:
- `makeReservation(Room room, String checkInDate, String checkOutDate)`: Creates a new reservation for the customer.
- `cancelReservation(Reservation reservation)`: Cancels an existing reservation.
- `displayCustomerInfo()`: Displays the customer's details and their reservations.

- Description: The `Customer` class represents a customer in the hotel system who can make and cancel reservations. It interacts with the `Reservation` and `Room` classes.

6. Hotel Management Class:

- Attributes:
- `rooms`: A collection of all rooms in the hotel.
- `customers`: A collection of all registered customers.
- `reservations`: A collection of all reservations made.

- Methods:
- `addRoom(Room room)`: Adds a new room to the hotel's inventory.
- `registerCustomer(Customer customer)`: Registers a new customer.
- `findRoomByNumber(int roomNumber)`: Searches for a room by its number and returns it if found.
- `makeReservation(Customer customer, Room room, String checkInDate, String checkOutDate)`: Creates a reservation for a customer.

- Description: The `HotelManagement` class manages the overall operations of the hotel, including room inventory, customer registration, and reservation handling.

7. Exception Handling:

- Custom Exceptions:
- `RoomNotAvailableException` Class:
- Description: Thrown when a customer attempts to book a room that is not available.
- `InvalidPaymentException` Class:
- Description: Thrown when a payment method is invalid or cannot be processed.
- `ReservationNotFoundException` Class:
- Description: Thrown when attempting to cancel or modify a reservation that does not exist.

- Usage: Implement exception handling to manage errors, such as trying to book an unavailable room, making an invalid payment, or attempting to modify a non-existent reservation. Use try-catch blocks to gracefully handle these exceptions and provide meaningful feedback to the user.

8. Hotel Reservation System Launcher:

- Purpose: Implement a `HotelReservationLauncher` class with a `main()` method to simulate the hotel reservation process. This class will:
- Create instances of different room types (e.g., `SingleRoom`, `DoubleRoom`, `SuiteRoom`).
- Register customers and add rooms to the hotel.
- Allow customers to make, view, and cancel reservations.

- Handle payments using different payment methods, showcasing polymorphism.
- Demonstrate exception handling by simulating common errors, such as booking an unavailable room or making an invalid payment.

Example Usage:

- Create a `Room` base class with common attributes and methods, and extend it with specific room types like `SingleRoom`, `DoubleRoom`, and `SuiteRoom`, demonstrating inheritance and polymorphism.
- Develop a `Reservation` class to manage the process of booking rooms, including calculating total costs and storing reservation details.
- Implement polymorphic behavior in the `Payment` class and its derived classes (`CreditCardPayment`, `PayPalPayment`) to handle different types of payments.
- Simulate error scenarios by implementing custom exceptions and using try-catch blocks to manage issues like unavailable rooms and invalid payments.
- Launch the hotel reservation system using a launcher class to integrate all components and simulate a complete reservation process, including handling edge cases with exception handling.

This problem statement emphasizes the use of inheritance and polymorphism to manage different types of rooms and payments, while also incorporating exception handling to create a robust and user-friendly hotel reservation system in Java. The system design demonstrates the effective application of object-oriented principles and exception handling in a real-world scenario.