**H**

**otel Reservation System**

**Documentation**

* **Project Supervisors:**

Dr. Mahmoud Bassioni

Eng. Mohamed Hussein Kamel

* **Project Members:**

|  |  |
| --- | --- |
| Name | ID |
| Mohamed Osman Younes | 2101189 |
| Salma Abdelhameed Mohamed | 2101541 |
| Shahd Mahmoud Ramzy | 2101599 |
| Sara Ezzat Shaker | 2101572 |
| Shahd Hossam Elden Hussein | 2101611 |

* **Project Idea:**

Hotel Reservation System

**Introduction:**

The Hotel Reservation System is a desktop application designed to manage hotel room bookings, reservations, and customer services. This system employs several design patterns, including Singleton, Factory, Prototype, Builder, and Decorator, to ensure a scalable, maintainable, and modular architecture. These patterns are integrated seamlessly into the system to enhance its functionality and maintainability, catering to the dynamic needs of users and administrators.

**1. Singleton Pattern**

**Purpose:**

Ensures only one instance of specific classes exists throughout the application.

**Classes:**

* **ReservationManager:** Manages all reservation activities centrally, such as checking availability, confirming bookings, and maintaining a database of active reservations.

**Code:**

* **PaymentProcessor:** Handles payment transactions to guarantee consistency and manage payment operations, including refunds and receipts.

**Code:**

**2. Factory Pattern**

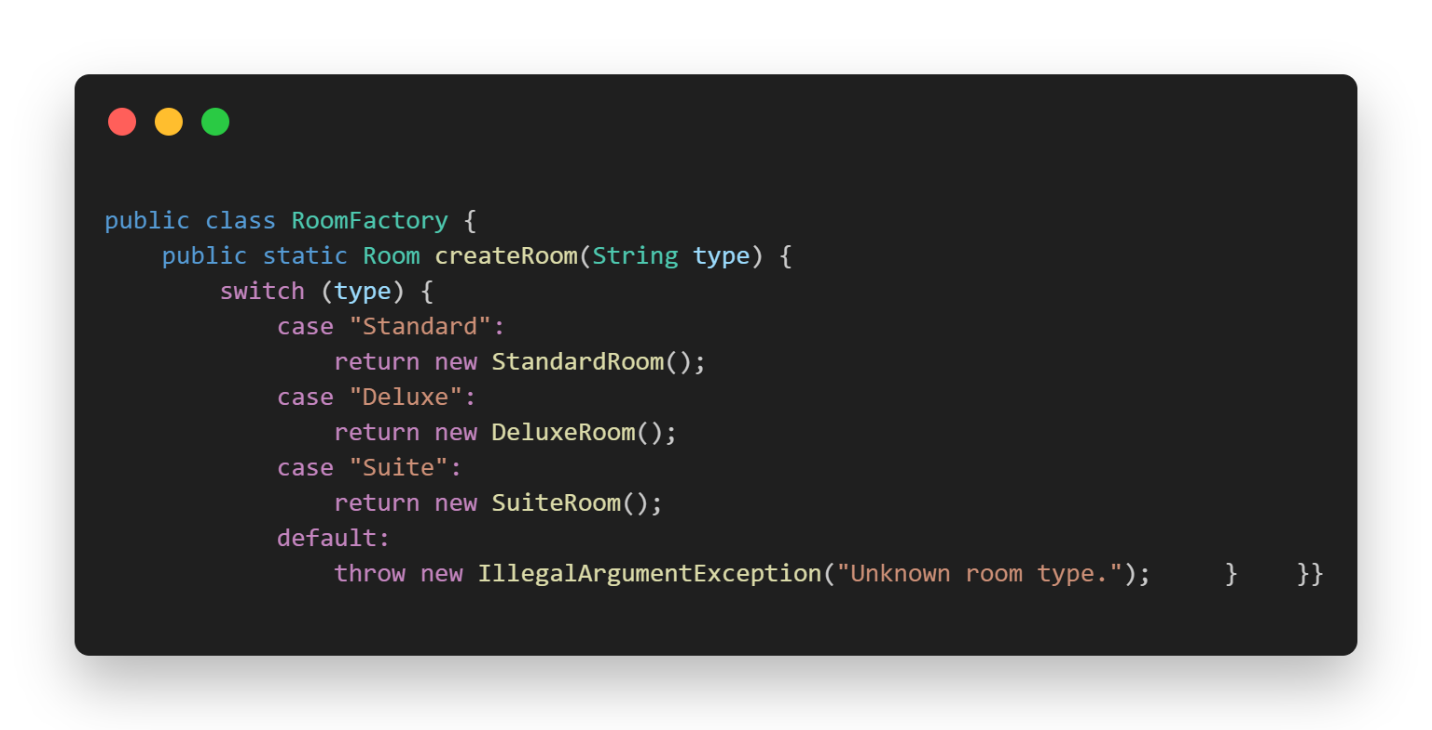
**Purpose:**

Creates objects based on specified criteria, promoting reusability and separation of instantiation logic.

**Classes:**

* **RoomFactory:** Dynamically generates room types, such as Standard, Deluxe, and Suite, based on user input and specific requirements.

**Code:**

****

* **CustomerProfileFactory**: Creates customer profiles (e.g., Regular, VIP, Corporate), enabling tailored services and pricing strategies.

**Code:**

**3. Builder Pattern**

**Purpose:**

Constructs complex objects step by step, enabling clarity and customization.

**Classes:**

* **ReservationBuilder**: Constructs reservation objects with optional attributes, such as meal plans, airport transfers, and flexible check-in times.

**Code:**

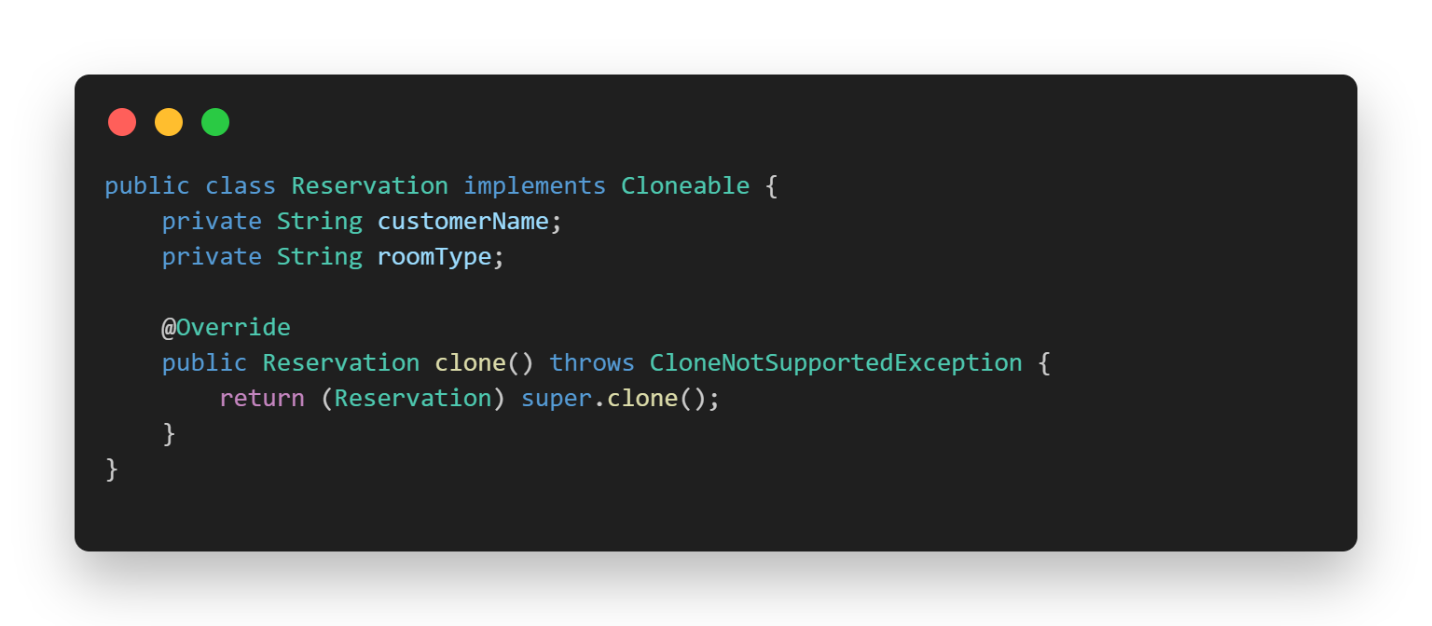
**4. Prototype Pattern**

**Purpose:**

Enables cloning of objects to create new instances with the same properties, useful for repeat bookings or similar reservations.

**Classes:**

* **Reservation**: Implements cloning for duplicating reservations, ensuring quick and efficient creation of similar bookings.

**Code:**

**5. Decorator Pattern**

**Purpose:**

Dynamically adds new functionalities to objects without altering their structure.

**Classes:**

* **RoomServiceDecorator**: Adds additional features like spa access, premium internet, or balcony access, allowing for highly customizable room offerings.

**Code:**

**Integration of Patterns:**

**Singleton:** Ensures consistency for shared resources like ReservationManager and PaymentProcessor.

**Factory:** Simplifies object creation for room types and customer profiles.

**Builder:** Manages the creation of complex reservation objects with multiple optional features.

**Prototype:** Facilitates duplication of reservations for repeat customers or group bookings.

**Decorator:** Adds dynamic enhancements to room features, improving user satisfaction and service flexibility.

**Summary:**

The Hotel Reservation System demonstrates the effective application of design patterns to create a robust, flexible, and maintainable architecture. Each pattern addresses specific design challenges, ensuring scalability and adaptability for future requirements. With these patterns, the system is equipped to handle the evolving demands of the hospitality industry, providing a seamless and enriching experience for users and administrators alike.