Patrones de Diseño
Nombre: Manuel Valdes
Lenguaje: Java

Objetivo del Proyecto

Simular un sistema de reservas en línea para hoteles, vuelos y alquiler de autos, aplicando 9 patrones de diseño fundamentales (3 de cada categoría: creacionales, estructurales y de comportamiento). La solución es modular, extensible y sigue buenas prácticas de diseño orientado a objetos.

PATRONES CREACIONALES

- **1. Singleton – SistemaReservas**

Garantiza una única instancia del sistema que coordina las reservas.

Método getInstancia() asegura que solo se cree una instancia y que sea accesible globalmente.

Útil para centralizar la lógica del sistema sin permitir múltiples instancias conflictivas.

- **2. Factory Method – ServicioFactory**

Encapsula la lógica de creación de distintos tipos de servicios (Hotel, Vuelo, Auto).

Permite crear objetos sin acoplar el código cliente a clases concretas.

Ejemplo: ServicioFactory.crearServicio("Hotel").

- **3. Builder – Reserva.Builder**

Facilita la construcción de objetos Reserva complejos paso a paso.

Soporta opciones como añadir seguro o servicios extra sin usar múltiples constructores.

PATRONES ESTRUCTURALES

- **4. Adapter – AdapterReservadorHotel**

Convierte la interfaz de un proveedor externo (ReservadorHotelExterno) en una interfaz compatible (ReservadorExterno) con el sistema.

Permite integrar servicios de terceros sin modificar su código original.

- **5. Facade – FachadaReservas**

Proporciona una interfaz simple para gestionar una reserva completa con todos los pasos internos: creación, configuración y notificación.

Oculta la complejidad del subsistema.

- **6. Decorator – ReservaDecorator, SeguroDecorator, ExtraDecorator**

Permite agregar funcionalidades opcionales a una reserva básica de forma dinámica.

PATRONES DE COMPORTAMIENTO

- **7. Strategy – MetodoPago, PagoTarjeta, PagoPaypal**

Define algoritmos intercambiables para métodos de pago.

El sistema puede cambiar la estrategia de pago en tiempo de ejecución.

- **8. Observer – Notificador, Observador**

Notifica automáticamente a los clientes cuando su reserva ha sido procesada.

Se puede ampliar fácilmente para múltiples canales de notificación (correo, SMS, etc.).

- **9. Command – Comando, ComandoReserva**

Encapsula la acción de procesar una reserva como un objeto.

Facilita funciones como deshacer/reintentar y registrar un historial de operaciones.

INTERFAZ DE USUARIO

Tipo: Consola

1. Simula pasos reales:
2. Crear y personalizar reservas.
3. Procesar mediante la fachada.
4. Aplicar decoradores.
5. Elegir estrategia de pago.
6. Ejecutar comandos.

El main() muestra cómo interactuar con el sistema mediante una serie de llamadas claras.

PRUEBAS Y FUNCIONALIDAD

Se probaron combinaciones de servicios, personalizaciones, notificaciones y pagos.

Cada patrón fue verificado individualmente y como parte del flujo completo.

CONCLUSIÓN
Este proyecto demuestra:

Comprensión profunda de patrones de diseño.

Aplicación modular y extensible.

Integración limpia entre lógica de negocio y arquitectura orientada a objetos.