1.Problema: Clonación de Personajes en un Videojuego de Aventuras

Imagina que estás desarrollando un videojuego de aventuras en el que los jugadores pueden controlar a un grupo de héroes con habilidades únicas. Necesitas implementar un sistema de clonación de personajes. Los jugadores deben poder clonar y personalizar héroes existentes para formar equipos estratégicos.

Requerimientos del sistema:

1. Los jugadores deben poder clonar a los héroes existentes en el juego para crear nuevos personajes con habilidades idénticas.

2. Cada héroe tiene un conjunto único de habilidades especiales que incluyen ataques, defensas y habilidades mágicas.

3. Los héroes pueden pertenecer a diferentes clases, como guerreros, magos o arqueros, y cada clase tiene su propio conjunto de habilidades base.

4. Los jugadores deben poder personalizar los nombres y aspectos visuales de los héroes clonados después de la clonación.

5. Los héroes deben mantener un registro de su experiencia, nivel y puntos de habilidad, y esta información debe copiarse correctamente al clonar un héroe.

6. El sistema debe ser eficiente y permitir la creación de múltiples clones de un mismo héroe sin duplicar innecesariamente los datos.

2. La empresa está desarrollando una aplicación de mensajería en tiempo real que permite a los usuarios comunicarse desde múltiples dispositivos. Cada vez que un usuario recibe un nuevo mensaje, todos los dispositivos del usuario deben ser notificados para que el mensaje se muestre en cada uno de ellos. Para mantener la flexibilidad y desacoplar la lógica de notificación de la aplicación principal.

El patrón que escoja deberá permitir

1. **Notificación en Tiempo Real:** Cada dispositivo del usuario debe recibir una notificación cuando llegue un nuevo mensaje.
2. **Desacoplamiento:** La aplicación de mensajería debe poder notificar a cualquier número de dispositivos sin saber detalles específicos sobre cada uno de ellos.
3. **Flexibilidad:** Los dispositivos pueden ser añadidos o eliminados en tiempo de ejecución sin necesidad de modificar la lógica principal de la aplicación.

3. Integración de un Sistema de Pago Externo

Imagina que estás desarrollando una plataforma de comercio electrónico y deseas ofrecer a tus clientes múltiples opciones de pago, incluyendo tarjetas de crédito, PayPal y un nuevo sistema de pago digital que acaba de ser lanzado. Sin embargo, este nuevo sistema de pago tiene una interfaz incompatible con tu plataforma existente, lo que dificulta su integración.

Tu solución debe permitir lograr una integración suave y coherente del nuevo sistema de pago en tu plataforma de comercio electrónico, sin tener que modificar la lógica interna de tu sistema existente. Para facilitar la adopción de nuevas tecnologías y sistemas en tu aplicación sin interrumpir su funcionalidad principal.

4. Sistema de Gestión de Tareas

Contexto:
Imagina un sistema de gestión de tareas en el que los usuarios pueden crear, editar, eliminar y completar tareas. Cada acción realizada por el usuario corresponde a una acción que debe ser ejecutada. Además, es importante mantener un registro de todas las acciones realizadas para permitir la reversión de las mismas si es necesario.

Aplicación del Patrón:
En este escenario, el patrón será aplicado para encapsular cada una de las acciones que el usuario puede realizar sobre una tarea.
El patrón que seleccione debe tener los siguientes beneficios:

- Desacopla el invocador de los objetos que realizan las acciones.
- Permite la extensión de nuevas operaciones sin modificar el código existente.
- Facilita el registro de acciones para realizar operaciones de reversión.

5. Decoración de Habitaciones en un Hotel
Imagina que estamos desarrollando un sistema para gestionar la decoración de habitaciones en un hotel de lujo. Cada habitación puede tener una decoración básica, pero los huéspedes pueden solicitar mejoras y adiciones para personalizar su experiencia. Estas mejoras pueden incluir servicios adicionales, como flores frescas, chocolate gourmet, vino de alta calidad, etc.

Aplicación del Patrón:

En este escenario, el patrón que se seleccione se utilizará para agregar características adicionales y personalizadas a las habitaciones del hotel de manera dinámica.

El patrón que seleccione debe tener los siguientes beneficios:

- Permite agregar nuevas funcionalidades a objetos existentes de manera dinámica.

- Proporciona una alternativa flexible a la subclase para extender funcionalidades.
- Mejora la legibilidad y el mantenimiento del código al separar las preocupaciones.