Ejercicios Desarrollo Software

03/04/24

1 Identifica el Patrón de Diseño Software

1. Patrón que especifique un mecanismo para construir un complejo objeto paso a paso, donde se necesitan diferentes representaciones del objeto para diferentes procesos de pedido en un restaurante online. BUILDER
2. Desarrolle un marco para juegos que permita extender fácilmente el tipo de personajes disponibles, sin cambiar el código que añade nuevos personajes al juego. FACTORY METHOD
3. Construya un sistema de edición de gráficos que debe permitir al usuario crear objetos personalizados y luego duplicarlos sin conocer los detalles de cómo se crean o estructuran. PROTOTYPE
4. Implemente un gestor de configuración para su aplicación que asegure que solo haya una instancia de la configuración en toda la aplicación y que sea accesible globalmente. SINGLETON
5. Integre una librería de procesamiento de imágenes de terceros en su aplicación de galería, pero la interfaz de la librería no es compatible con el sistema actual de su aplicación. ADAPTER
6. Desarrolle un sistema de archivos donde archivos y carpetas puedan ser tratados de manera uniforme. COMPOSITE
7. Implemente un sistema de pedidos de café donde se pueda añadir dinámicamente nuevos complementos y variantes a los cafés sin alterar las clases existentes. DECORATOR
8. Cree una interfaz simple para el complejo sistema de subcomponentes de un home theater que incluye el proyector, amplificador, reproductor de DVD y las luces de la habitación. FACHADA
9. Desarrolle un sistema de alertas meteorológicas donde los usuarios puedan recibir actualizaciones en tiempo real cuando se emitan nuevas previsiones. OBSERVER
10. Diseñe un sistema de navegación que pueda calcular la ruta entre dos puntos utilizando diferentes algoritmos de ruta como el más rápido, el más corto y el más económico. ESTRATEGIA
11. Imagine que es un ingeniero de software encargado de simplificar la interacción de los usuarios con un sistema de reservación de viajes, que incluye múltiples componentes como reservación de vuelos, hoteles y alquiler de autos. Cada uno de estos componentes tiene su propio sistema de interfaz complicado. FACHADA

2 Identificar Patrón de diseño, realizar diagrama UML, código de las clases y un main()

COMPOSITE

1. Necesitas diseñar un sistema de gestión de tareas para un equipo de proyecto. Las tareas pueden ser simples o complejas, las últimas consisten en sub-tareas. Debes permitir que las tareas simples y las tareas compuestas sean tratadas de la misma manera.

ESTRATEGIA

2. Estás creando un simulador de tráfico que debe poder a diferentes distancias (euclidea, manhatan y distancia coseno). El usuario podrá elegir qué distancia ejecutar en tiempo de ejecución para ir desde un punto A a otro punto B.

DECORATOR

3. Una empresa de café desea una aplicación que pueda añadir ingredientes adicionales a una orden de café. Cada adición (leche, azúcar, crema, etc.) agrega un costo adicional y la descripción del café debe actualizarse en consecuencia. La solución debe permitir añadir nuevos ingredientes sin cambiar el código existente de las clases de café.

DECORATOR

4. Desarrolle una aplicación para personalizar vehículos. Cree la clase base Vehiculo con un método costo y descripcion. Extienda esta funcionalidad con clases adicionales como ConGPS, ConPinturaEspecial y ConAsientosDeLujo que agreguen comportamientos y costos adicionales. Estas clases deben poder envolver objetos de tipo Vehiculo y modificar su comportamiento sin cambiar la clase Vehiculo original.

FACTORY METHOD

5. Una aplicación de blog necesita generar diferentes tipos de publicaciones: Artículo, Noticia, y Entrevista. Cada tipo de publicación tiene un método publicar que difiere en su comportamiento. Diseña un sistema que permita la creación de estas publicaciones sin determinar específicamente la clase de publicación en el código cliente.

   - Implementar las clases Publicacion, Articulo, Noticia, y Entrevista, todas con el método publicar.
   - Crear un método o clase Creador que, dependiendo de un parámetro o contexto dado, instancie y retorne el tipo de publicación adecuado.

PROTOTYPE

6. Para un juego de estrategia, se requiere un sistema para crear copias exactas de unidades de combate sin conocer sus clases específicas. Las unidades tienen estados y comportamientos que pueden ser duplicados. Implementa este sistema asegurando que el proceso de duplicación sea eficiente y seguro.

   - Implementar la interfaz correspondiente.
   - Crear clases concretas Infanteria, Arquero, y Caballeria que permitan crear otras copias a partir de ellas.

ADAPTER

7. Un sistema de análisis de datos trabaja con diferentes proveedores de datos. Uno de los proveedores suministra los datos en un formato incompatible con el sistema actual. Implementa una solución que permita al sistema utilizar los datos de este proveedor sin cambiar el código fuente del sistema de análisis.

   - Crear una interfaz ProveedorDatos que el sistema de análisis utiliza para leer datos.
   - Implementar una clase AdaptadorProveedorDatos que convierte los datos del proveedor incompatible a un formato que el sistema de análisis puede utilizar.