CANCÚN

Materia: Desarrollo de Software

Profesor: José Dimas Lujan Castillo

Alumno: Miguel Nicolau dos Santos Allar

Fecha: 16/03/2022

Creacionales

Singleton

Explicación:

El patrón Singleton es utilizado cuando queremos crear una sola instancia de la clase.

Elementos:

1.- Un constructor tipo private.

2.- Un Getter Static que verifique que no se haya creado una instancia de su clase y manda a llamar al constructor privado.

3.- Una variable Static que contenga esa instancia.

Caso de uso:

Crear una conexión a una base de datos.

Builder

Explicación:

El patrón Builder permite personalizar nuestro objeto y agregar sus valores uno a uno en cualquier orden y en cualquier momento; o no agregarlos.

Elementos:

1. Constructor del objeto base.
2. Setters que ayuden a personalizar y construir el objeto.

Caso de uso:

En la configuración de objetos que tienen diferentes opciones con diferentes combinaciones. Como lo sería una pizza en domino’s en la opción de “arma tu pizza” o un NFT.

Factory

Explicación:

Se utiliza para crear objetos de un mismo tipo, pero con algunas características o atributos diferentes.

Elementos:

1. Una interfaz del objeto base y fabrica.
2. Clases concretas de la variación del objeto y fábrica.

Caso de uso:

Cuando la creación de nuestro objeto es muy compleja y se requieren regresar objetos diferentes basados en factores dinámicos. Como la creación de enemigos en un videojuego que los genere aleatoriamente y/o basados en una configuración previa del usuario.

Prototype

Explicación:

Permite clonar objetos y copiarlos a otras variables.

Elementos:

1. Una función cuyo único propósito sea clonar al objeto.

Caso de uso:

Cualquier lugar donde una funcionalidad sea tener duplicados del objeto. Como copiar y pegar un archivo.

Abstract Factory

Explicación:

Nos permite crear una familia de objetos con un atributo específico que todos tengan y tener una fábrica para cada tipo de ese atributo.

Elementos:

1. Familia de objetos que se relacionan
2. Fabricas para cada tipo de diferencia.

Caso de uso:

Dark mode en cualquier dispositivo.

Estructurales

Proxy

Explicación:

Manipular un objeto mediante otro que tenga permitido controlar ese objeto.

Elementos:

1. Objeto a manipular.
2. Objeto/s que lo controlen.

Caso de uso:

Google docs. Nosotros no tenemos acceso directo a nuestros archivos sin embargo los manipulamos como si lo tuviéramos, pero eso es gracias al proxy o Google.

Decorator

Explicación:

Te permite agregar funcionalidad al objeto al envolver dicho objeto en un envoltorio especial que contenga esa funcionalidad.

Elementos:

1. Una interfaz base de los decorators.
2. Implementación de cada decorator.

Caso de uso:

Responder mediante una notificación, dependiendo de la aplicación la forma en la que se envía el mensaje es diferente y al mensaje le agregar el wrapper de cómo se va a enviar.

Bridge

Explicación:

Permite separar una clase muy grande en 2 clases diferentes.

Elementos:

1. Se requiere de una interfaz
2. La implementación de la interfaz
3. Implementaciones concretas
4. Hacer la conexión entre ambas abstracciones

Caso de uso:

Se podría utilizar para separar clases muy grandes que tienen lógicas diferentes. Como una figura y su color, en vez de tener una clase por figura por color solo tienes una clase por figura y una clase color, y las unes.

Facade

Explicación:

Simplifica el uso de una clase muy compleja mediante el uso de una interfaz.

Elementos:

1. La interfaz que contendrá el uso de la clase compleja

Caso de uso:

Crear una framework o una API

Composite

Explicación:

Te permite componer objetos en una estructura tipo árbol. Y trabajar individualmente con estas estructuras.

Elementos:

1. Una interfaz que describa comportamientos comunes en el árbol.
2. La hoja quien se encargará de hacer la mayoría del trabajo
3. Una clase contenedor quien no conoce la clase de sus objetos y trabaja con ellos mediante la interfaz.

Caso de uso:

Se podría utilizar cuando necesitemos tratar a un grupo de objetos como si fueran uno solo.

Flyweight

Explicación:

Este objeto se utiliza para disminuir el uso de la RAM al hacer que todos los objetos compartan el mismo atributo pesado que está llenando la memoria. Como una textura.

Elementos:

1. Una clase que contenga los objetos similares y pesados a la cual referenciar.

Caso de uso:

Particles en videojuegos. En vez de que millones de partículas contengan su textura, que todas comparten, solo contienen una referencia a la textura original así reduciendo el uso de RAM a solo un par de variables muy probablemente acerca de la posición de dicha partícula.

Comportamiento

Chain of Responsability

Explicación:

Permite crear una cadena de responsabilidad de menor a mayor donde cada objeto delega el trabajo que no puede manejar a su superior.

Elementos:

1. Una interfaz que pida implementar la forma en la que manejara los request.
2. Las clases concretas que implementaran la interfaz y su forma de responder al request.

Caso de uso:

Cuando hayan handlers diferentes a los cuales se les pueda pasar un request si es que uno no lo puede manejar.

Memento

Explicación:

Te permite guardar el estado de un objeto y restaurarlo

Elementos:

1. Clase Memento quien representa el que guarda el estado
2. Clase Caretaker quien guarda los mementos
3. Clase Originator quien manejara el guardado y restauracion del estado

Caso de uso:

Save files de un videojuego. El memento podria ser cada archivo que contenga el save file, Caretaker el folder y Originator la interfaz en la que eliges que save usar y quien se encarga de cargar dicho save.

Iterator

Explicación:

Te permite definir una forma de recorrer y manejar una colección.

Elementos:

1. Una interfaz que defina la funcionalidad basica de las colecciones
2. Una clase concreta que la implemente

Caso de uso:

Una lista en cualquier lenguaje. O incluso el mismo for i to array.length que hacemos para recorrer la colección.

Mediator

Explicación:

Funciona de mediador y te permite controlar la comunicación entre dependencias a que solo utilicen al mediador.

Elementos:

1. Una interfaz mediador quien declara la forma comunicación entre las clases
2. El mediador concreto quien forma las conexiones para la comunicación.

Caso de uso:

Implementar una comunicación clara entre 2 objetos.

Observer

Explicación:

Permite suscribirse a los objetos para así notificar de cambios al objeto que se observa.

Elementos:

1. Una clase que tenga un funcionamiento dependiente del cambio de un objeto
2. La clase observer que se suscribe al objeto y notifica a la clase interesada

Caso de uso:

Eventos en videojuegos.