TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR

ANALISIS Y DISEÑO DE SOFTWARE

IMPLEMENTACIÓN DEL PATRÓN SINGLETON

IMPLEMENTACIÓN DEL PATRÓN SINGLETON

La implementación del patrón Singleton representa un paso técnico crucial dentro del diseño de software cuando se desea restringir la creación de múltiples instancias de una clase. Este patrón asegura que solo se genere una instancia de un objeto, proporcionando un punto de acceso global a dicha instancia. La clave de su implementación reside en el uso de modificadores de acceso, almacenamiento estático y control de concurrencia cuando es necesario (Sampedro Hernández, 2011).

Estructura general del Singleton

Para implementar un Singleton en un lenguaje orientado a objetos, como Java o C#, se siguen tres pasos fundamentales:

1. **Constructor privado.** Se impide que otras clases puedan crear instancias directamente.
2. **Variable estática privada.** Contiene la única instancia que se devolverá.
3. **Método de acceso público (generalmente estático).** Devuelve la instancia única, creándose si aún no existe.

Ejemplo en Java: Singleton clásico

En este ejemplo, la clase GestorConfiguracion controla que solo se cree una instancia y que dicha instancia sea accesible mediante el método estático getInstancia(). Esta implementación es suficiente para aplicaciones monohilo.

Singleton con seguridad en entornos multihilo

Cuando se trabaja en un entorno con múltiples hilos, existe el riesgo de que dos hilos creen la instancia al mismo tiempo. Para evitar esta situación, se utiliza una técnica llamada doble verificación con sincronización.

Aquí, el uso de volatile y synchronized garantiza que solo un hilo podrá crear la instancia y que cualquier cambio se refleje adecuadamente en los demás hilos.

Variaciones comunes en la implementación:

- **Singleton Eager.** La instancia se crea al cargar la clase, ideal cuando se sabe que siempre se utilizará.

- **Singleton con inicialización estática.** Usar bloques static para controlar la lógica de inicialización más compleja.

Aplicaciones prácticas del patrón

- En sistemas de videojuegos, el motor de audio o la gestión de eventos se implementan como Singletons para garantizar una instancia coherente en todo el sistema.
- En sistemas empresariales, los controladores de auditoría o de seguridad global suelen centralizarse con Singleton para asegurar un punto de control único.

Buenas prácticas en su implementación

- Evitar el uso excesivo del patrón, debido a que puede introducir un alto acoplamiento.
- Asegurarse de que sea realmente necesario que exista una sola instancia.
- Implementar interfaces para facilitar las pruebas y desacoplar su uso.