Patrón Singleton

Ejemplo: archivo de logs

https://www.youtube.com/watch?v=GGq6s7xhHzY&list=PLJkcleqxxobUJlz1Cm8WYd-F_kckkDvc8&index=5

- **El objetivo del patrón singleton** es de asegurar que existe tan una sola instancia de una clase concreta, por lo que se consigue que sea imposible instanciar más de una vez la clase de un objeto.
- Singleton provee una accesibilidad global de esta instancia, por lo que podemos acceder desde cualquier parte del programa.

Implementación

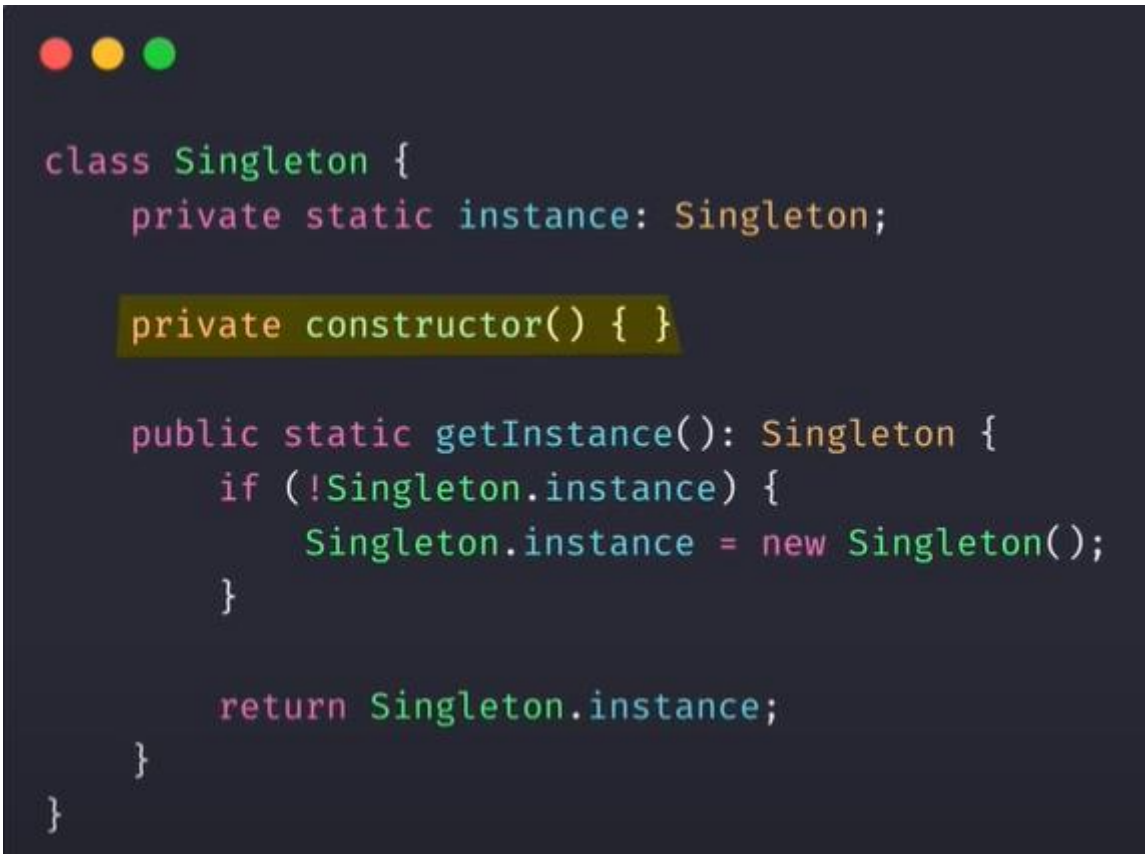
Veamos un caso de uso donde se puede aplicar, gestionar recursos de forma individual que se aplica a toda la aplicación: Un archivo de Logs.

En este caso de uso se necesita acceder al mismo archivo centralizado donde no se debería tener varias instancias escribiendo el mismo.

¿Pero cómo se hace?

Para ello se debe crear un **constructor que sea privado**. Por lo que conseguimos:

- Que no sea accesible desde fuera, por lo tanto, ganamos el control sobre la instanciación de la misma.
- La propia clase quien gana el control de como se instancian sus propios objetos
- Bloqueamos que cualquiera desde fuera de la propia clase pueda instancias de esa clase
- Evitamos escribir el new nombreClaseSingleton() en cualquier lado del programa



```
class Singleton {
    private static instance: Singleton;

    private constructor() { }

    public static getInstance(): Singleton {
        if (!Singleton.instance) {
            Singleton.instance = new Singleton();
        }

        return Singleton.instance;
    }
}
```

La forma entonces para instanciar esa clase concreta es con otra función, la que normalmente se suele llamar **getInstance**() o similar (algo que indique que estás obteniendo la instancia del singleton)

```
public static getInstance(): Singleton {
    if (!Singleton.instance) {
        Singleton.instance = new Singleton();
    }

    return Singleton.instance;
}
```

Esta función estática getInstance() se encargará de construir a nivel interno de construir esta instancia. Puede llamar al constructor en forma privada sin problemas por ser parte de la esta clase y por lo tanto puede comprobar

Si ya existe una instancia creada anteriormente **?** La función devuelve la instancia **:** Crea por primera vez.

```
if (!Singleton.instance) {
    Singleton.instance = new Singleton();
}
```

"Conseguimos entonces que exista únicamente una única instancia o copia creada de esta clase"

Representación en UML