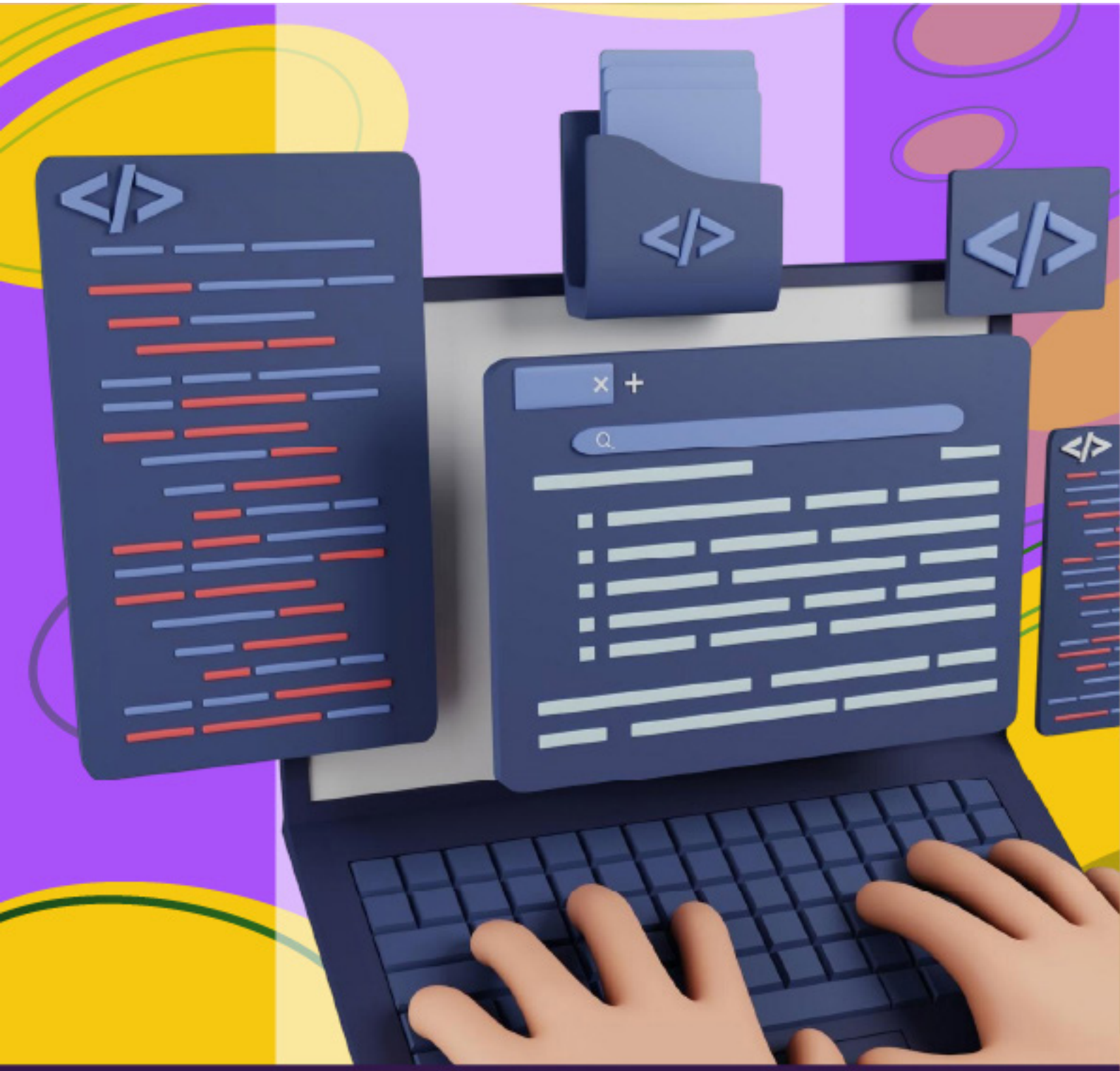
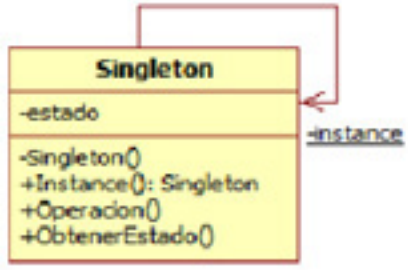
TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR



ANALISIS Y DISEÑO DE SOFTWARE

PATRÓN SINGLETON

PATRÓN SINGLETON: CONCEPTO Y FUNDAMENTOS



El patrón Singleton es una solución arquitectónica perteneciente al grupo de los patrones creacionales, cuyo propósito principal es asegurar que una clase solo tenga una única instancia en todo el sistema y que exista un punto de acceso global a dicha instancia (Sampedro Hernández, 2011). Este enfoque resulta especialmente útil cuando se necesita coordinar acciones a través de un único recurso compartido, como un archivo de configuración, un logger, una conexión a base de datos o un manejador de impresión.

Desde una perspectiva técnica, el patrón Singleton impide la creación múltiple de objetos de una clase determinada mediante el uso de un constructor privado. De esta manera, ninguna otra clase puede instanciar el objeto de forma directa. En su lugar, se define un método estático que verifica si la instancia ya existe. Si no existe, la crea; de lo contrario, devuelve la instancia previamente generada. Esta técnica garantiza consistencia en el acceso y uso de ciertos componentes críticos dentro de una aplicación.

Estructura básica del patrón Singleton

```
public class ConfiguracionSistema {
    private static ConfiguracionSistema instancia;

    // Constructor privado
    private ConfiguracionSistema() {
        // Cargar configuración del sistema
    }

    // Método de acceso global
    public static ConfiguracionSistema getInstancia() {
        if (instancia == null) {
            instancia = new ConfiguracionSistema();
        }
        return instancia;
    }
}
```

En este ejemplo, “ConfiguracionSistema” solo puede ser instanciada una vez. Cada vez que se llama a getInstancia(), se devuelve la misma instancia, garantizando así un comportamiento uniforme en toda la aplicación.

Aplicaciones comunes del patrón Singleton

- **Registro de logs.** Un sistema de logging debe registrar eventos globales desde distintos puntos del programa. Utilizar un Singleton asegura que todos los módulos escriben sobre el mismo archivo o canal.
- **Manejador de base de datos.** Para evitar múltiples conexiones abiertas innecesariamente, se puede emplear un Singleton que administre una única conexión activa.
- **Controladores de configuración.** Los parámetros globales que afectan el comportamiento de una aplicación se centralizan en un objeto Singleton, evitando duplicidad o incoherencias.

Ventajas

- Control de acceso a una única instancia.
- Reducción de consumo de memoria, porque no se crean objetos repetidos.
- Facilita la coordinación de procesos cuando se trabaja con recursos compartidos.

Desventajas

- Puede dificultar las pruebas unitarias si no se implementa adecuadamente.
- Introduce acoplamiento global, lo que puede ir en contra de ciertos principios como la inversión de dependencias si no se maneja con cuidado.

En conclusión, el patrón Singleton representa una estrategia poderosa y efectiva para garantizar que ciertas clases tengan un comportamiento único y controlado en una aplicación. Su correcta implementación mejora la eficiencia, la coherencia del sistema y facilita la gestión de recursos compartidos, siempre que se utilice de forma consciente y en los contextos apropiados.