Informe Método Singleton

Joshua Abril

Patrones de Diseño

Luis Alberto Reyes

Uninpahu

2025

**¿Qué es?** El Singleton es un patrón de diseño utilizado cuando se requiere controlar el acceso a recursos compartidos, como conexiones a bases de datos, configuraciones globales o manejo de impresoras, entre otros. Asegura que no haya más de una instancia de la clase a lo largo de la ejecución de un programa.

**Funcionalidad:** El **método Singleton** es un patrón de diseño creacional que asegura que una clase tenga **una sola instancia** y proporciona un **punto de acceso global** a esa instancia. En otras palabras, este patrón controla la creación de objetos de una clase, garantizando que solo se cree una instancia, incluso si se hace múltiples solicitudes para crear nuevos objetos de esa clase.

**Ejemplo breve:** Tenemos una clase BaseDeDatos, y que que solo exista una conexión a la base de datos durante todo el ciclo de vida de la aplicación. Si utilizamos el patrón Singleton, cada vez que tratemos de crear una nueva instancia de BaseDeDatos, el sistema devolverá la misma instancia previamente creada, evitando la creación de múltiples objetos innecesarios.

**Código de Ejemplo:**

public class BaseDeDatos {

private static BaseDeDatos instancia;

private BaseDeDatos() {

}

public static BaseDeDatos getInstancia() {

if (instancia == null) {

instancia = new BaseDeDatos();

}

return instancia;

}

public void conectar() {

System.out.println("Conectado a la base de datos");

}

}

public class Main {

public static void main(String[] args) {

BaseDeDatos db1 = BaseDeDatos.getInstancia();

BaseDeDatos db2 = BaseDeDatos.getInstancia();

System.out.println(db1 == db2);

db1.conectar();

}

}

**¿Qué se consigue con este patrón?**

Este patrón asegura que la clase BaseDeDatos solo tenga una única instancia a lo largo de toda la ejecución del programa, lo cual es útil para controlar recursos compartidos como una conexión a base de datos.

**Explicación del Ejemplo del Repositorio:**

El código implementa el **patrón Singleton** en la clase Conexión para asegurar que solo haya **una única instancia** de conexión durante la ejecución del programa. La clase Conexión tiene un **constructor privado** y un método estático getInstancia() que crea la instancia solo si no existe, garantizando un acceso controlado. En App.java, se obtiene esta única instancia, se ejecutan los métodos conectar () y desconectar (), y se verifica que el objeto creado pertenece a la clase Conexión. Esto permite gestionar la conexión de manera centralizada, evitando múltiples instancias innecesarias.