viernes, 10 de mayo de 2024

Practica

Patrones de diseño

Código	Descripción
	Se crea una clase llamada *Database* contiene un constructor donde mediante el if checa si es que ya existe una instancia de Database y si no le da un valor, en caso de que si haya regresa su valor. Tengo entendido que en *query* lo único que hace es que imprime el mensaje "*Ejecutando consulta:*", pero aún no termino de comprender que función hace el sql. Fuera de la clase *Database* se crean dos instancias de la clase *Database* las cuales después se usan para compararse e indicar si son la misma instancia. Yo creo que se esta usando el patrón **Singleton** porque este nos habla de que se puede limitar una clase para que solo se instancié una vez y es lo que se hace en el if del constructor. Algo que también ayuda a verlo es que al comparar db1 y db2 la salida es true, entonces es la misma instancia de la clase.

Código	Descripción
	Se crea una clase llamada *Logger* que contiene su constructor donde se declara un arreglo llamado *logs* que esta vacío. Existe una subclase llamada *log* donde entran mensajes, aquí lo que se hace es que mediante el push el mensaje que entra se agrega al arreglo *logs* y al hacerlo imprime “*Log registrado:*” . En *getInstance* lo que sucede es que de nuevo revisa si es que hay una instancia en Logger y en caso de que no se instancia, en caso de que sí se devuelve el valor. Fuera de la clase se declaran *logger1* y *logger2* donde para ambos se llama a la función *getInstance.* Se comparan para verificar si son el mismo valor y después se llama al método log en *logger1* que registra un mensaje en el array logs. Creo que también usa el patrón **Singleton** en el método *getInstance*.

Código	Descripción
	Se crea una clase llamada *User* y en su constructor se guarda el nombre de usuario, en el método *greet* lo imprime junto con el mensaje "*Hola, soy*". Hay una segunda clase *UserFactory* con un método (*createUser*) para crear instancias de *User* con el nombre que se da. Aquí es donde se crean los objetos. Se agrega una instancia en la linea 17 de *UserFactory* bajo el nombre *factory.* Se crean los usuarios usando el método *createUser.* Por último se llama al método *greet* con cada usuario, lo que imprime el saludo. En este código no pude identificar ningún patrón, pero creo que lo que sí contiene es el principio de **Single Responsability** ya que *User* solo tiene una responsabilidad que es guarda el nombre del usuario y su método para saludar, así como *UserFactory* solo crea instancias de *User*.