**Week 1 - Design Patterns - Hands-On Exercise**

**Student Name: Jitesh Sen  
Email:** [**jiteshsen2002@gmail.com**](mailto:jiteshsen2002@gmail.com) **Superset ID: 6387362**

**Exercise 1: Implementing the Singleton Pattern**

**1. Understanding Singleton Pattern**

**What is Singleton Pattern?**

Singleton pattern ensures only one object of a class can be created in the whole program. It uses private constructor and static method to control object creation. This pattern is useful for logging, database connections, and shared resources.

**2. Setup:** I created a Logger class using Singleton pattern to ensure consistent logging throughout the application. The class has private constructor, static variable, and getObject() method.

**3. Implementation**

I implemented Singleton pattern with three main parts:

* Private static variable to store single instance
* Private constructor to prevent external object creation
* Public static getObject() method that returns same instance always

**4. Analysis**

**Time Complexity:** O(1) for getting instance after first creation.

**Optimization:** Works efficiently for single-threaded applications. For multi-threading, we need synchronization.

**Conclusion:** Singleton pattern is perfect for logging utilities where we need one consistent logger throughout the application.

**Program Output:**

