# Exercise 1: Implementing the Singleton Pattern (Theory Answers)

## 🔸 What is the Singleton Design Pattern?

The Singleton Design Pattern ensures that a class has only one instance throughout the application's lifecycle and provides a global access point to that instance. It is commonly used when exactly one object is needed to coordinate actions across a system.

## 🔸 Why is it important to make the constructor private in a Singleton class?

Making the constructor private prevents external classes from creating new instances of the Singleton class. This enforces the Singleton constraint by ensuring that only the Singleton class itself can create its instance, typically through a static method.

## 🔸 How does the static method help in returning the single instance?

The static method (commonly named getInstance()) checks whether an instance of the class already exists. If not, it creates one and returns it. If the instance already exists, it simply returns the existing one. Since static methods belong to the class rather than any object, they can be called without creating an object.

## 🔸 What is the benefit of using a Singleton for logging purposes?

Using a Singleton for logging ensures that all components of the application use the same logging instance. This allows for consistent log formatting, centralized control over log behavior, and avoids issues like duplicate log files or resource contention. It also simplifies debugging and maintenance.