# Exercise 1: Implementing the Singleton Pattern

Scenario:  
You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

## Steps:

1. Create a new Java project named SingletonPatternExample.

2. Create a class named Logger that has a private static instance of itself.

3. Ensure the constructor of Logger is private.

4. Provide a public static method to get the instance of the Logger class.

5. Write code to ensure that the Logger class follows the Singleton design pattern.

6. Create a test class to verify that only one instance of Logger is created and used across the application.

## Code Implementation

### Logger.java

public class Logger {  
 private static Logger instance;  
  
 // Private constructor prevents instantiation from other classes  
 private Logger() {  
 System.out.println("Logger Initialized");  
 }  
  
 public static Logger getInstance() {  
 if (instance == null) {  
 instance = new Logger();  
 }  
 return instance;  
 }  
  
 public void log(String message) {  
 System.out.println("LOG: " + message);  
 }  
}

### Main.java

public class Main {  
 public static void main(String[] args) {  
 Logger logger1 = Logger.getInstance();  
 Logger logger2 = Logger.getInstance();  
  
 logger1.log("Application started");  
 logger2.log("Processing data...");  
  
 if (logger1 == logger2) {  
 System.out.println("Same Logger instance used!");  
 } else {  
 System.out.println("Different Logger instances found.");  
 }  
 }  
}

## Output



