# Singleton Pattern Example in Java

## 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. 1. Create a New Java Project named SingletonPatternExample.
2. 2. Define a Singleton Class:  
    - Create a class named Logger that has a private static instance of itself.  
    - Ensure the constructor of Logger is private.  
    - Provide a public static method to get the instance of the Logger class.
3. 3. Implement the Singleton Pattern:  
    - Write code to ensure that the Logger class follows the Singleton design pattern.
4. 4. Test the Singleton Implementation:  
    - Create a test class to verify that only one instance of Logger is created and used across the application.

## Logger.java

public class Logger {  
 // Private static instance  
 private static Logger instance;  
  
 // Private constructor  
 private Logger() {  
 System.out.println("Logger initialized.");  
 }  
  
 // Public method to provide access to the instance  
 public static Logger getInstance() {  
 if (instance == null) {  
 instance = new Logger();  
 }  
 return instance;  
 }  
  
 // Method to simulate logging  
 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();  
 logger1.log("First message");  
  
 Logger logger2 = Logger.getInstance();  
 logger2.log("Second message");  
  
 // Test if both references point to the same object  
 if (logger1 == logger2) {  
 System.out.println("Both logger1 and logger2 are the same instance.");  
 } else {  
 System.out.println("Different instances!");  
 }  
 }  
}