# DESIGN PATTERN AND PRINCIPLE - SINGLETON PATTERN IMPLEMENTATION

## Project Structure

Create a Java project named SingletonPatternExample with the following structure:  
SingletonPatternExample/  
├── src/  
│ ├── Logger.java  
│ └── LoggerTest.java

## 1. Logger.java (Singleton Class)

This class ensures only one instance is created using lazy initialization.

public class Logger {  
   
 private static Logger instance;  
  
 private Logger() {  
 System.out.println("Logger instance created.");  
 }  
  
 public static Logger getInstance() {  
 if (instance == null) {  
 instance = new Logger(); // Lazy initialization  
 }  
 return instance;  
 }  
  
 public void log(String message) {  
 System.out.println("LOG: " + message);  
 }  
}

Key Features:  
- Private constructor prevents direct instantiation  
- Static getInstance() method controls access to the single instance  
- Lazy initialization creates instance only when first needed  
- Thread-safe for single-threaded applications

## 2. LoggerTest.java (Test Class)

public class LoggerTest {  
 public static void main(String[] args) {  
   
 Logger logger1 = Logger.getInstance();  
 Logger logger2 = Logger.getInstance();  
  
 logger1.log("First log message");  
 logger2.log("Second log message");  
 System.out.println("\n=== Singleton Verification ===");  
 System.out.println("Logger1 hash: " + System.identityHashCode(logger1));  
 System.out.println("Logger2 hash: " + System.identityHashCode(logger2));  
 System.out.println("Same instance? " + (logger1 == logger2));  
  
 System.out.println("\n=== Constructor Access Test ===");  
 try {  
 Logger logger3 = new Logger();  
 } catch (Exception e) {  
 System.out.println("Constructor access failed: " + e.getMessage());  
 }  
 }  
}

## 3. Expected Output

Logger instance created.  
LOG: First log message  
LOG: Second log message  
  
=== Singleton Verification ===  
Logger1 hash: 366712642  
Logger2 hash: 366712642  
Same instance? true  
  
=== Constructor Access Test ===  
Constructor access failed: Logger() has private access in Logger

## 4. Explanation

The constructor message appears only once, proving single instantiation.  
Both logger1 and logger2 output messages successfully.  
Identity hash codes match, confirming same object instance.  
Direct constructor access fails due to private constructor.

## 5. How to Run

1. Create a Java project named SingletonPatternExample.  
2. Add Logger.java and LoggerTest.java with the provided code.  
3. Compile and run LoggerTest.java.  
4. Verify that the output matches expected results.

## 6. Thread-Safe Alternative (Bill Pugh Method)

public class Logger {  
 private Logger() {  
 System.out.println("Logger instance created.");  
 }  
  
 private static class SingletonHelper {  
 private static final Logger INSTANCE = new Logger();  
 }  
  
 public static Logger getInstance() {  
 return SingletonHelper.INSTANCE;  
 }  
  
 public void log(String message) {  
 System.out.println("LOG: " + message);  
 }  
}