# **Singleton Design Pattern: Java Implementation**

### **Overview**

### **This paper provides a Java example of the Singleton design pattern. The purpose of the Singleton pattern is to make sure that a class may have only a single instance and to give a global access point to it. This is illustrated with a Logger utility class, which should be created only once during the lifetime of the application.**

### **1. Logger.java (Singleton Class)**

This class is implemented as a Singleton. It features a private constructor to prevent direct instantiation and a public static method, getInstance(), to provide access to the single, shared instance.

public class Logger{

private static final Logger instance = new Logger();

private Logger() {

System.out.println("Logger instance created.");

}

public static Logger getInstance() {

return instance;

}

public void log(String var1) {

System.out.println("LOG: " + var1);

}

### **2. SingletonPatternTest.java (Test Class)**

This class serves to verify that the Logger class correctly implements the Singleton pattern. It requests the Logger instance twice and then compares the object references and their hash codes to confirm that both variables point to the exact same instance.

public class SingletonPatternTest {

public SingletonPatternTest() {

}

public static void main(String[] var0) {

System.out.println("--- Singleton Pattern Test ---");

System.out.println("Requesting Logger instance 1...");

Loggerer var1 = Loggerer.getInstance();

System.out.println("Requesting Logger instance 2...");

Loggerer var2 = Loggerer.getInstance();

var1.log("This is the first log message.");

var2.log("This is the second log message.");

System.out.println("\nVerifying if both instances are the same...");

if (var1 == var2) {

System.out.println("Success: Logger1 and Logger2 are the same instance.");

} else {

System.out.println("Failure: Logger1 and Logger2 are different instances.");

}

System.out.println("Hash code for Logger1: " + var1.hashCode());

System.out.println("Hash code for Logger2: " + var2.hashCode());

}

}

### **3. Execution and Output**

When the SingletonPatternTest class is executed, it produces the following output. This confirms that the Logger constructor is called only once ("Logger instance created."), and both Logger1 and Logger2 are references to the same object, as shown by the success message and identical hash codes.

