WEEK-1

**SKILL:DESIGN PRINCIPLES & PATTERNS**

**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:**

* + Create a new Java project named **SingletonPatternExample**.

**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.Implement the Singleton Pattern:**

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

**4.Test the Singleton Implementation:**

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

**SINGLETONPATTERNEXAMPLE**

**Logger.java**

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

public class Logger {

private static Logger singleInstance;

private int logCount = 0;

private Logger() {

System.out.println("[Logger] Initialized at: " + currentTime());

}

public static Logger getInstance() {

if (singleInstance == null) {

singleInstance = new Logger();

}

return singleInstance;

}

public void log(String message) {

logCount++;

String output = String.format("[Log-%02d @ %s] %s", logCount, currentTime(), message);

System.out.println(output);

}

private String currentTime() {

DateTimeFormatter dtf = DateTimeFormatter.ofPattern("HH:mm:ss");

return LocalDateTime.now().format(dtf);

}

}

**Main.java**

public class Main {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

logger1.log("System booting up.");

Logger logger2 = Logger.getInstance();

logger2.log("Configuration loaded.");

logger2.log("User login successful.");

if (logger1 == logger2) {

System.out.println("[Info] Singleton confirmed: Only one logger instance used.");

} else {

System.out.println("[Warning] Singleton failed: Different instances created.");

}

}

}

**OUTPUT:**

