**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.

**SOLUTION**

**Step-1:**

Create Java Project

* Create a new Java project: SingletonPatternExample
* Inside the src folder, create two files:
  + Logger.java
  + Main.java

**Step-2:**

**Logger.java**

public class Logger {

    private static Logger instance;

    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);

    }

}

**Step-3:**

**Main.java**

public class Main {

    public static void main(String[] args) {

        Logger logger1 = Logger.getInstance();

        logger1.log("First log message");

        Logger logger2 = Logger.getInstance();

        logger2.log("Second log message");

        if (logger1 == logger2) {

            System.out.println("Both logger1 and logger2 refer to the same instance.");

        } else {

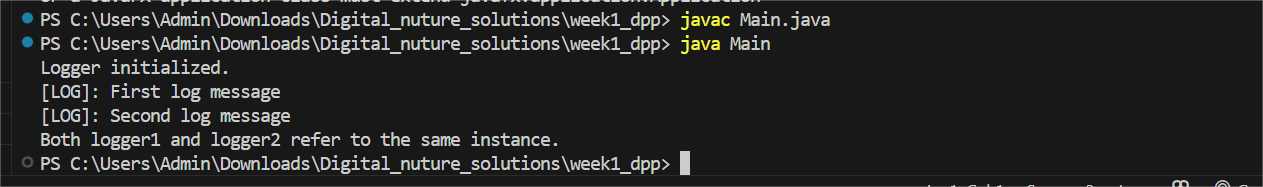
            System.out.println("Different instances exist! Singleton failed.");

        }

    }

}

**OUTPUT:**



**Step-4 :**

**Objective:**  
To verify that the Logger class truly follows the **Singleton Design Pattern**, we need to test whether only **one instance** of the class is created and shared across the application.

public class SingletonTest {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

logger1.log("This is the first log message.");

logger2.log("This is the second log message.");

if (logger1 == logger2) {

System.out.println("Both logger1 and logger2 are the same instance.");

} else {

System.out.println("Different instances were created (not a singleton).");

}

}

}

**OUTPUT:**

This is the first log message.

This is the second log message.

Both logger1 and logger2 are the same instance.