**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.

## **CODES:**

### ***Logger.java***

package week1.designPatterns.SingletonPattern;

public class Logger {

    private static Logger singleInstance;

    private Logger() {

        System.out.println("Logger initialized...");

    }

    public static Logger getInstance() {

        if (singleInstance == null) {

            singleInstance = new Logger();

        }

        return singleInstance;

    }

    public void log(String message) {

        System.out.println("Log: " + message);

    }

}

***Main.java***

package week1.designPatterns.SingletonPattern;

public class Main {

    public static void main(String[] args) {

        Logger logger1 = Logger.getInstance();

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

        Logger logger2 = Logger.getInstance();

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

        if (logger1 == logger2) {

            System.out.println("Both logger instances are the same. Singleton works!");

        } else {

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

        }

    }

}

* **OUTPUT**

