**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.

**Source Code :**

**Logger.java**

package singleton;

public class Logger {

private static Logger instance;

private Logger() {

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

}

public static synchronized Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

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

}

}

**TestSingleton.java**

package singleton;

public class TestSingleton {

public static void main(String[] args) {

Logger first\_logger = Logger.getInstance();

Logger second\_logger = Logger.getInstance();

first\_logger.log("This is first log message.");

second\_logger.log("This is second log message.");

if (first\_logger == second\_logger) {

System.out.println("Only one Logger instance exists. Singleton confirmed.");

} else {

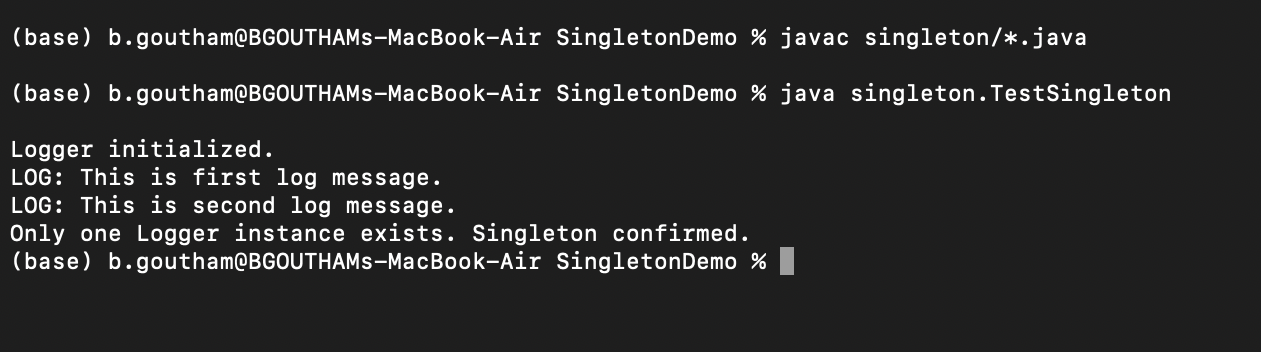
System.out.println("More than one Logger instances exist. Singleton violated.");

}

}

}

**Output :**

****