**Design Pattern And Principles**

**EXERCISE 1: Implementing the Singleton Pattern**

**Source Code**

// Logger.java

class Logger {

// Step 1: Create a private static instance

private static Logger instance;

// Step 2: Make the constructor private

private Logger() {

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

}

// Step 3: Provide a public method to get the instance

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

// Optional: A method to demonstrate logging

public void log(String message) {

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

}

}

// Main.java

public class Main {

public static void main(String[] args) {

// Step 4: Test the Singleton Implementation

Logger logger1 = Logger.getInstance();

logger1.log("First message");

Logger logger2 = Logger.getInstance();

logger2.log("Second message");

// Check if both references point to the same instance

if (logger1 == logger2) {

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

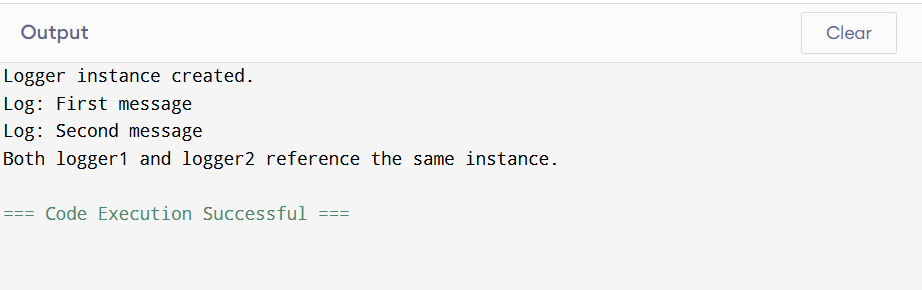
} else {

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

}

}

}

**Output**