**SINGLETON FACTORY PATTERN**

**QUESTION :**

**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 :**

**Logger.java :**

public class Logger {

private static Logger logger;

private Logger(){

System.out.println("Logger Constructor called");

}

public static Logger getInstance(){

if(logger == null){

logger = new Logger();

}

return logger;

}

public void log(String msg){

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

}

}

**Main.java :**

public class Main {

public static void main(String[] args) {

Logger log1 = Logger.getInstance();

Logger log2 = Logger.getInstance();

log1.log("message from log 1");

log2.log("message from log 2");

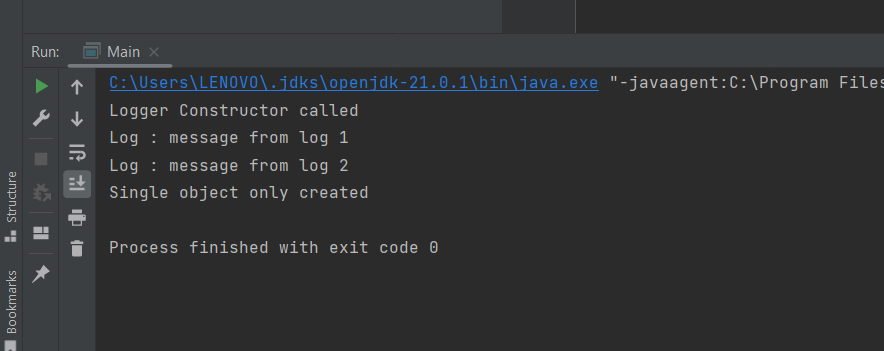
if(log1 == log2)

System.out.println("Single object only created");

}

}

**OUTPUT :**

****