**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.

Answer(Created on 17/06/2025):

Logger.java

**package** designpatterns;

**public** **class** logger {

**private** **static** logger *instance*;

**private** logger() {

// **TODO** Auto-generated constructor stub

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

}

**public** **static** logger getInstance() {

**if** (*instance*==**null**) {

**synchronized**(logger.**class**) {

**if**(*instance*==**null**) {

*instance*= **new** logger();

}

}

}

**return** *instance*;

}

**public** **void** log(String message) {

System.***out***.println(message);

}

}

test.java

**package** designpatterns;

**public** **class** test {

**public** test() {

// **TODO** Auto-generated constructor stub

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

logger log1=logger.*getInstance*();

log1.log("1");

logger log2=logger.*getInstance*();

log2.log("2");

**if**(log1==log2) {

System.***out***.println("success");

System.***out***.println("1)"+log1.hashCode());

System.***out***.println("2)"+log2.hashCode());

}**else** {

System.***out***.println("fail");

}

}

}

Output Screenshot:

