|  |  |
| --- | --- |
| **Cognizant Digital Nurture 4.0: Deep Skilling Hands-on** | |
| Name: Rajatesh Paul | Superset ID: 6365356 |
| Week: 01 | **Design Patterns and Principles** |

**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.

1. 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.

1. Implement the Singleton Pattern:

* Write code to ensure that the Logger class follows the Singleton design pattern.

1. Test the Singleton Implementation:

* Create a test class to verify that only one instance of Logger is created and used across the application.

#Code-

**Logger.java**-

package Singleton;  
public class Logger {  
 private static Logger *instance*;  
 private Logger(){  
 }  
 public static Logger getInstance(){  
 if (*instance*==null){  
 *instance* = new Logger();  
 }  
 return *instance*;  
 }  
}

**SingletonTest.java-**

package Singleton;  
public class SingletonTest{  
 public static void main(String[] args){  
 Logger L1=Logger.*getInstance*();  
 Logger L2=Logger.*getInstance*();  
 if(L1==L2) {  
 System.*out*.println("Both Logger1 and Logger2 are the same instance!");  
 }  
 else{  
 System.*out*.println("Both are Different");  
 }  
 }  
}

**Output**-

