**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.

1. Create a new Java Project:

* Create a new Java Project named SingletonPatternExample.
* A Java project named SingletonPatternExample is created in eclipse IDE.

2. Define a singleton class:

* Create a class named Logger that has a private static instance of itself.

public class Logger {

private static Logger instance;

* The Logger class ensures that only one instance of the class is created. The private static variable instance holds the single instance of the Logger class
* Ensure the constructor of Logger is private.

private Logger() {

}

* The constructor is private to prevent instantiation from outside the class
* Provide a public static method to get the instance of the Logger class.

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

* The getInstance() method checks if the instance is null and creates a new one if necessary. It returns the existing instance otherwise.

3. Implement the Singleton Pattern:

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

public class Logger {

private static Logger instance;

private Logger() {

}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

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

}

}

* The Logger class follows the Singleton design pattern as it:
* Has a private static instance variable.
* Provides a private constructor.
* Offers a public static method to return the single instance.

4. Test the Singleton Implementation:

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

import java.util.Scanner;

public class TestLogger {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

if (logger1 == logger2) {

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

} else {

System.out.println("Different instances exist.");

}

System.out.println("Enter a log message (type 'exit' to quit):");

while (true) {

String message = scanner.nextLine();

if (message.equalsIgnoreCase("exit")) {

break;

}

logger1.log(message);

}

scanner.close();

}

}

* In TestLogger, we obtain two references to the Logger instance using getInstance().
* We check if both references point to the same object using the == operator.
* If both references are the same, it confirms that the Logger class is following the Singleton pattern.
* We then use the logger to log messages based on user input. The user can type messages to be logged, and they can type 'exit' to quit the logging process.
* Output:

