**Exercise 1: Implementing the Singleton Pattern**

### Step 1: Create a New Java Project

1. Open your IDE (e.g., Eclipse, IntelliJ IDEA).
2. Create a new Java project and name it SingletonPatternExample.

### Step 2: Define a Singleton Class

1. In the src folder of your project, create a new package (e.g., com.singleton).
2. Inside this package, create a new Java class named Logger.

### Step 3: Implement the Singleton Pattern

1. Open the Logger class and implement the Singleton design pattern as follows:

**Code:**

package com.singleton;

public class Logger {

// Private static instance of the Logger class

private static Logger instance;

// Private constructor to prevent instantiation from other classes

private Logger() {

// Initialize any resources here if needed

}

// Public method to provide access to the instance

public static Logger getInstance() {

if (instance == null) {

// Synchronized block to remove overhead

synchronized (Logger.class) {

if (instance == null) {

// Create instance if it does not exist

instance = new Logger();

}

}

}

return instance;

}

// Method to log messages

public void log(String message) {

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

}

}

### Step 4: Test the Singleton Implementation

1. Create another class named SingletonTest in the same package to test the Singleton implementation.

package com.singleton;

public class SingletonTest {

public static void main(String[] args) {

// Get the instance of Logger

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

// Log messages using both instances

logger1.log("First message");

logger2.log("Second message");

// Check if both instances are the same

if (logger1 == logger2) {

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

} else {

System.out.println("logger1 and logger2 are different instances.");

}

}

}

### Explanation

**Logger Class**:

* 1. The Logger class has a private static instance variable named instance.
  2. The constructor is private to prevent instantiation from outside the class.
  3. The getInstance() method checks if the instance is null and if so, synchronizes on the class and creates a new instance if it is still null. This ensures thread safety.
  4. The log(String message) method is a simple example of a logging method.

**SingletonTest Class**:

* 1. The SingletonTest class gets the instance of the Logger class twice and logs messages using both instances.
  2. It checks if both instances are the same by comparing them with ==.