**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:**

* **Project Name**: SingletonPatternExample
* Use your preferred IDE (like IntelliJ IDEA, Eclipse, or NetBeans) to create a new Java project.

**2. Define a Singleton Class:**

**Logger Class Implementation**:

* The Logger class should be designed to ensure that only one instance is created and used throughout the application.

public class Logger {

// Private static instance of the Logger class

private static Logger instance;

// Private constructor to prevent instantiation from outside

private Logger() {

// Optional: Initialize any resources if necessary

System.out.println("Logger initialized");

}

// Public static method to provide access to the instance

public static Logger getInstance() {

if (instance == null) {

// If instance is null, create a new one

instance = new Logger();

}

// Return the single instance

return instance;

}

// Method to log messages

public void log(String message) {

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

}

}

**Key Points**:

* **Private Constructor**: The constructor is private, so the class cannot be instantiated from outside.
* **Static Instance**: A static instance variable holds the single instance of the class.
* **Public Static Method**: The getInstance() method checks if the instance is null, creates it if necessary, and returns it.

**3. Implement the Singleton Pattern:**

* The code in the Logger class ensures that only one instance of the Logger is created and shared across the application.

**4. Test the Singleton Implementation:**

**Test Singleton in a Separate Class**:

* Create a TestSingleton class to verify that only one instance of the Logger is used.

public class TestSingleton {

public static void main(String[] args) {

// Get the first instance of Logger

Logger logger1 = Logger.getInstance();

logger1.log("First log message");

// Get another instance of Logger

Logger logger2 = Logger.getInstance();

logger2.log("Second log message");

// Check if both instances are the same

if (logger1 == logger2) {

System.out.println("Both logger instances are the same.");

} else {

System.out.println("Logger instances are different.");

}

}

}