**Exercise 1 – Implementation of Singleton Pattern**

**What is Singleton Pattern?**

* The Singleton Pattern is a **Creational Pattern.**
* The Singleton Pattern restricts the instantiation of a class to one single object. It provides a way to access the object globally.

**Key Principles of Singleton Pattern:**

1. private constructor - prevents outside code from creating instances.
2. static instance – holds only one instance
3. public accessor (i.e. getInstance() method) – create instance once and returns the same object in all the future calls.

**Objective**

Implement the Singleton Design Pattern for a logging utility in java ensuring only one instance existence during the entire application process.

**Steps for Implementing Singleton Design Pattern:**

**Step 1:** Create a new Java Project

* Name of the project: **SingletonPatternExample**
* The project folder consists of 2 files

SingletonPatternExample/

├─ Logger.java // Singleton class

└── LoggerTest.java // Main method

**Step 2:**

* Develop a Singleton Class
* The class name is Logger and it follows the key principles of singleton Design Pattern.
* Responsibilities:
* Ensure only one object of the class exists.
* Make the instance accessible from anywhere in the application.
* Restrict instantiation by making the constructor private.

**Step 3:** Logger.java

|  |
| --- |
| package exercise;  public class Logger {  // private static instance of Logger  private static Logger instance;  // private constructor to prevent instantiation from outside  private Logger () {  System.out.println("Logger instance created…");  }  // public static method to provide access to the instance  public static Logger getInstance () {  if (instance == null)  instance = new Logger (); // lazy initialization  return instance;  }  Public void log () {  System.out.println("This message is from Logger...");  }  } |

**Step 4:** LoggerTest.java

|  |
| --- |
| package exercise;  public class LoggerTest {  public static void main (String[] args) {  // Getting the Log instance  Logger log1 = Logger.getInstance();  Logger log2 = Logger.getInstance();  // Confirming singleton behavior  log1.log();  System.out.println("Are both logger instances same? " + (log1 == log2));  }  } |

**Expected Output:**

