**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**.
2. **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.
3. **Implement the Singleton Pattern:**
   * Write code to ensure that the Logger class follows the Singleton design pattern.
4. **Test the Singleton Implementation:**
   * Create a test class to verify that only one instance of Logger is created and used across the application.

## Answer:-

Logger.java

public class Logger {

    private static Logger log = null;

    private Logger(){

    };

    public static Logger getInstance(){

        if(log == null){

            log = new Logger();

        }

        return log;

    }

    public void logDetails(String str){

        System.out.println(str);

    }

}

Main.java

public class Main {

    public static void main(String[] args) {

        Logger.getInstance().logDetails("Application has started.");

        Logger log1 = Logger.getInstance();

        Logger log2 = Logger.getInstance();

        log1.logDetails("Log 1 initiated.");

        log2.logDetails("Log 2 initiated.");

        if(log1==log2){

            System.out.println("Only one instance of Logger is created and used across the application");

        } else {

            System.out.println("More than one instance of Logger is being used.");

        }

    }

}

Output:-

