Project: Implementing the Singleton Pattern in Java

Objective:

To ensure that a logging utility (Logger class) has only one instance throughout the application lifecycle using the Singleton Design Pattern.

Steps Implemented:

1. Java Project Setup: Created a new project 'SingletonPatternExample'.
2. Logger Singleton Class:
   * Constructor is private.
   * A static instance variable holds the single object.
   * A public static method getInstance() returns that object.
3. Singleton Logic: Ensures lazy instantiation - only when getInstance() is called first time.
4. Test Class: Verifies that two calls to getInstance() return the same instance.

# Logger.java

public class Logger {

private static Logger instance;

private Logger() {

System.out.println("Logger instance created.");

}

public static Logger getInstance() { if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) { System.out.println("Log: " + message);

}

}

# TestLogger.java

public class TestLogger {

public static void main(String[] args) { Logger logger1 = Logger.getInstance(); Logger logger2 = Logger.getInstance();

logger1.log("First log"); logger2.log("Second log");

System.out.println(logger1 == logger2); // true

}

}

**Output Screenshot:**

****