**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.

The **Singleton Design Pattern** is a **creational pattern** that ensures **"A class has only one instance in the entire application, and provides a global access point to that instance."**

**Implementation:**

**//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 message");

        logger2.log("Second message");

        if (logger1 == logger2) {

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

        } else {

            System.out.println("Different instances exist.");

        }

    }

}

**Output:**

