**Exercise 1: Implementing the Singleton Pattern**

**Program:**

**1. Logger.java**

class Logger{

    private static Logger instance;

    private Logger(){

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

    }

    public static Logger getInstance(){

        if(instance==null)

            instance=new Logger();

        return instance;

    }

    public void log(String message){

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

    }

}

**2. Singleton\_Pattern.java**

class Singleton\_Pattern {

    public static void main(String[] args) {

        Logger L1=Logger.getInstance();

        Logger L2=Logger.getInstance();

        L1.log("This is the Message from Logger 1");

        L2.log("This is the Message from Logger 2");

        if(L1==L2)

            System.out.println("Both the Logger 1 and Logger 2 are of Same instance");

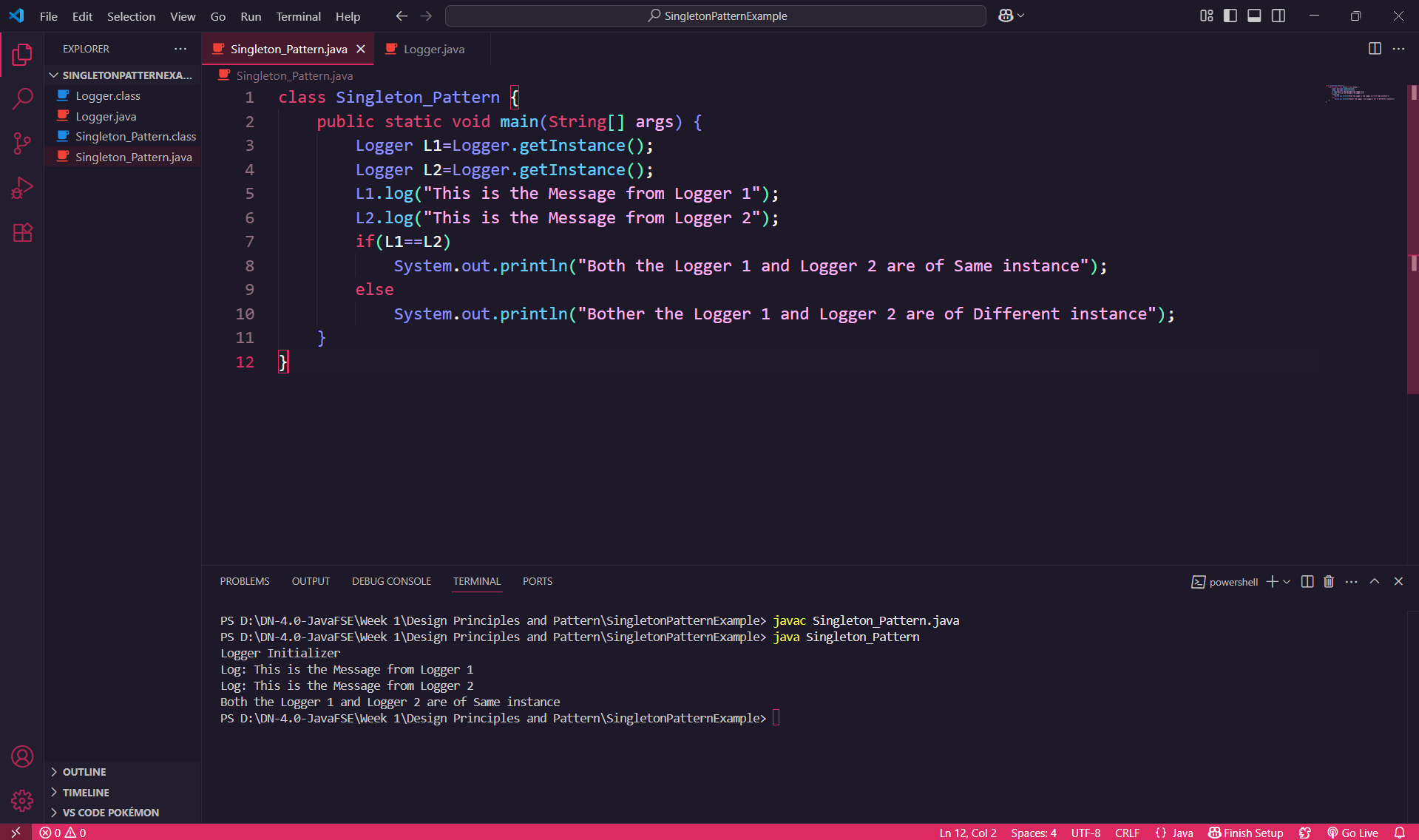
        else

            System.out.println("Bother the Logger 1 and Logger 2 are of Different instance");

    }

}

**Output:**



**Explanation:**

This program is Singleton Design Pattern. I created a Logger class where only one object can be created using the getInstance() method. In the main() method, I called getInstance() twice and stored it in L1 and L2. Both refer to the same Logger object. I used it to log two messages, and at the end, I checked whether both instances are the same. This ensures that only one instance is used throughout the program."