Week 1 :DESIGN PATTERN AND PRINCIPLES

Question :

**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.

**Code:**

logger :

public class logger{

    private static logger instance;

    private logger () {

        System.out.println("logger() constructor called");

    }

    public static synchronized logger getInstance() {

        if (instance == null) {

            instance = new logger();

        }

        return instance;

    }

    public void log(String message) {

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

    }

}

Testclass:

public class Testclass {

    public static void main(String[] args) {

        logger log1 = logger.getInstance();

        log1.log("This is a log message.");

        logger log2 = logger.getInstance();

        log2.log("This is another log message.");

        if (log1 == log2) {

            System.out.println("log1 and log2 are the same instance.");

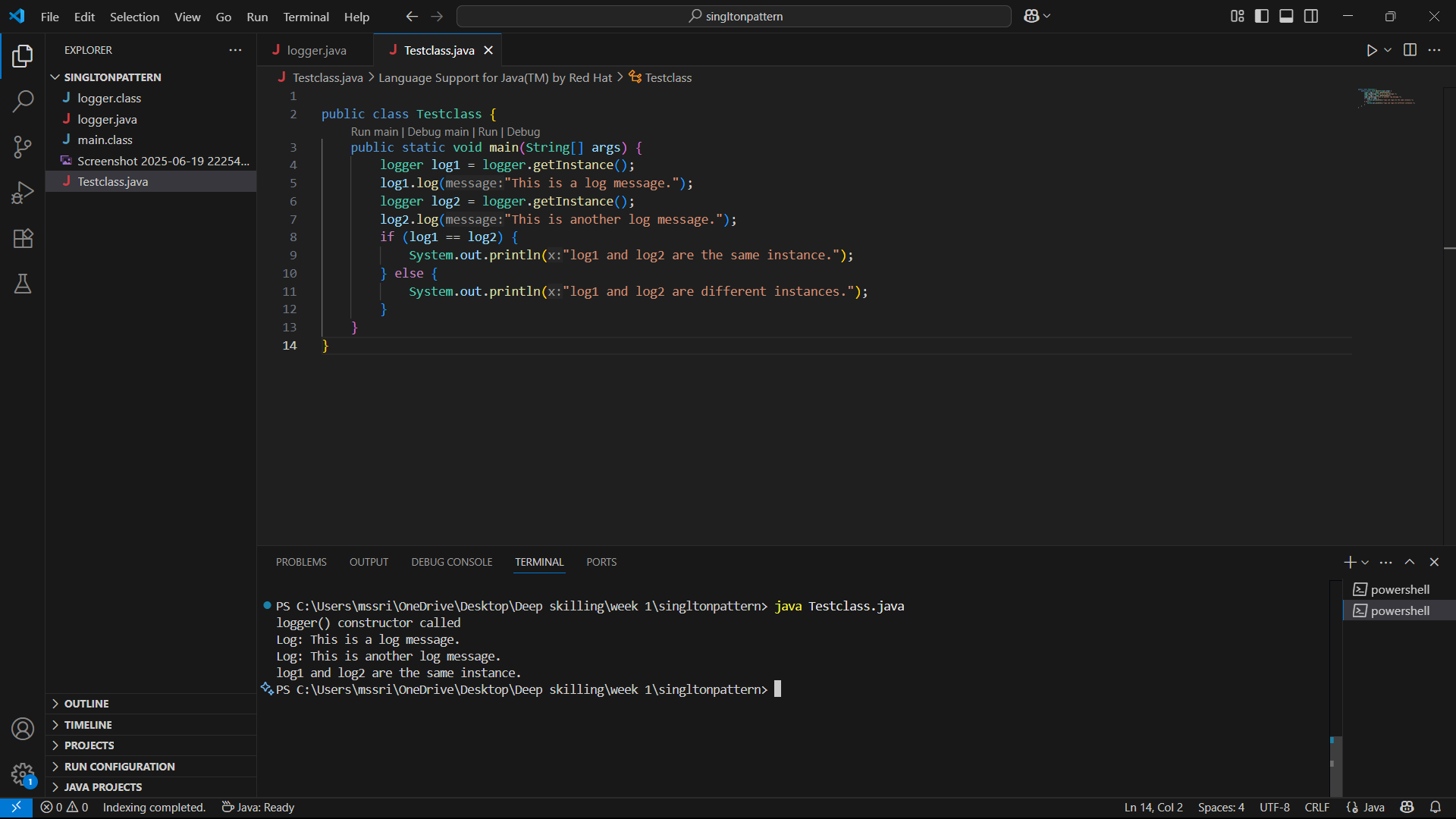
        } else {

            System.out.println("log1 and log2 are different instances.");

        }

    }

}

Output:

Question:

**Exercise 2: Implementing the Factory Method Pattern**

**Scenario:**

You are developing a document management system that needs to create different types of documents (e.g., Word, PDF, Excel). Use the Factory Method Pattern to achieve this.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **FactoryMethodPatternExample**.
2. **Define Document Classes:**
   * Create interfaces or abstract classes for different document types such as **WordDocument**, **PdfDocument**, and **ExcelDocument**.
3. **Create Concrete Document Classes:**
   * Implement concrete classes for each document type that implements or extends the above interfaces or abstract classes.
4. **Implement the Factory Method:**
   * Create an abstract class **DocumentFactory** with a method **createDocument()**.
   * Create concrete factory classes for each document type that extends DocumentFactory and implements the **createDocument()** method.
5. **Test the Factory Method Implementation:**
   * Create a test class to demonstrate the creation of different document types using the factory method.

**Code:**

**Interface :**

public interface Document {

void open();

}

**Concrete Classes:**

**Worddocument:**

**public class wordocument implements Document {**

**public void open() {**

**System.out.println("Opening Word document");**

**}**

**}**

**Pdfdocument:**

**public class pdfdocument implements Document {**

**public void open(){**

**System.out.println("Opening PDF document");**

**}**

**}**

**Exceldocument:**

**public class exceldocument implements Document {**

**public void open(){**

**System.out.println("Opening Excel document");**

**}**

**}**

**Abstract Factory Classes:**

**Documentfactory:**

**public abstract class Documentfactory {**

**public abstract Document createDocument( );**

**}**

**Concrete Factory Classes:**

**Worddocumentfactory:**

**public class wordocumentfactory extends Documentfactory {**

**public Document createDocument() {**

**return new wordocument();**

**}**

**}**

**Pdfdocumentfactory:**

**public class pdfdocumentfactory extends Documentfactory {**

**public Document createDocument() {**

**return new pdfdocument();**

**}**

**}**

**Exceldocumentfactory:**

**public class exceldocumentfactory extends Documentfactory {**

**public Document createDocument(){**

**return new exceldocument();**

**}**

**}**

**Testclass:**

**public class testclass {**

**public static void main(String[] args) {**

**Documentfactory wordfactory= new wordocumentfactory();**

**Document worddoc = wordfactory.createDocument();**

**worddoc.open();**

**Documentfactory pdffactory = new pdfdocumentfactory();**

**Document pdfdoc = pdffactory.createDocument();**

**pdfdoc.open();**

**Documentfactory excelfactory= new exceldocumentfactory();**

**Document exceldoc=excelfactory.createDocument();**

**exceldoc.open();**

**}**

**}**

**Output:**

