**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.

**Solution:**

**FileName:Logger.class**

package com.cognizant.SingletonPatternExample;

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;

    }

}

**FileName:Main.java**

package com.cognizant.SingletonPatternExample;

public class Main {

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

*Logger* logger1 = Logger.getInstance();

*Logger* logger2 = Logger.getInstance();

        if(logger1==logger2){

            System.out.println("Both Logger1 and Logger2 are the same instance of Logger class and follows Singleton Design Pattern." );

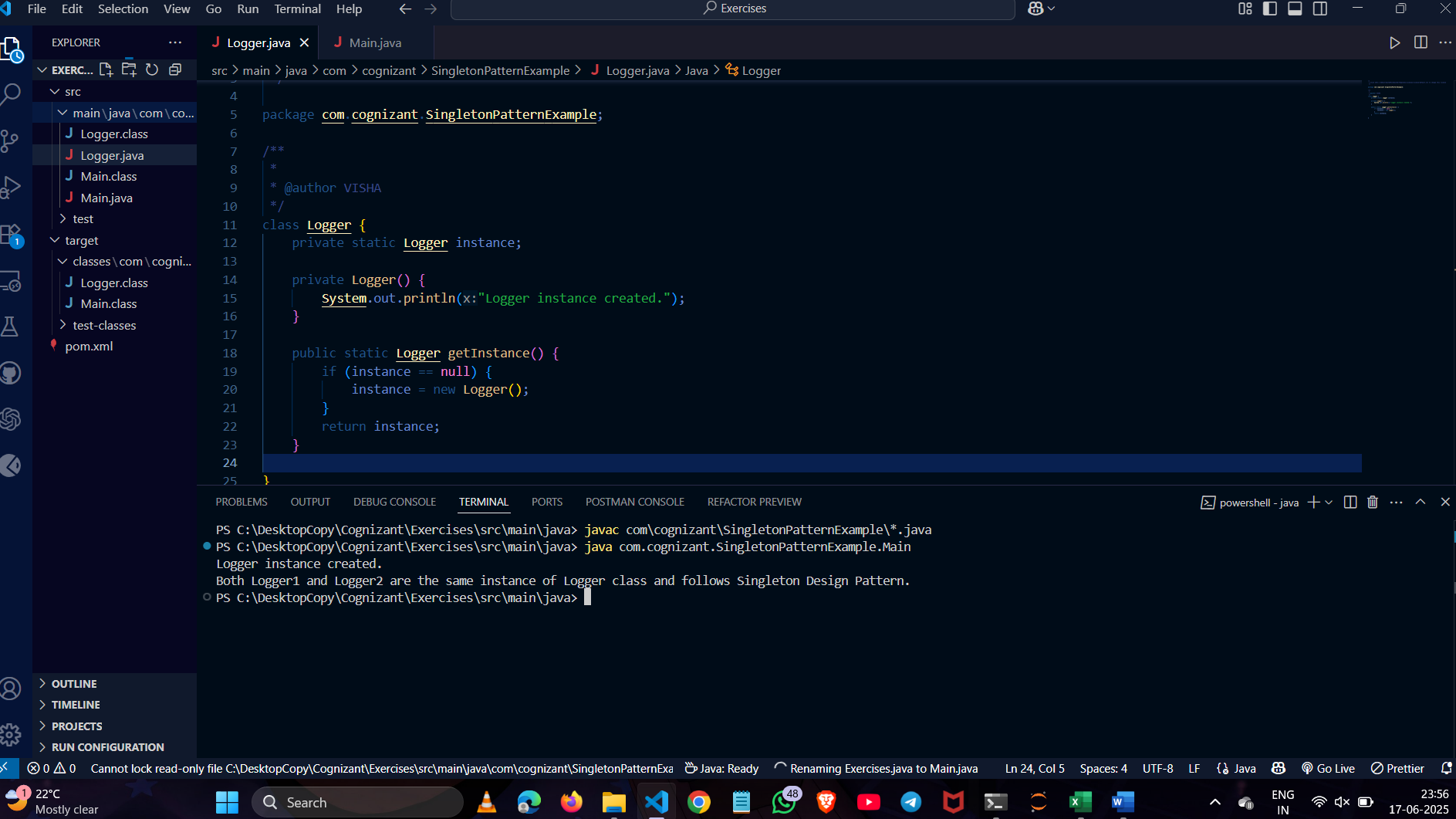
        }else{

            System.out.println("Logger1 and Logger2 are different instances of Logger class and does not follow Singleton Design Pattern." );

        }

    }}

**Output:**



**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.**

**Solution:**

**FileName:WordDocument.java**

package document;

public interface WordDocument {

void openword();

}

**FileName:ExcelDocument.java**

package document;

public interface ExcelDocument{

void openExcel();

}

**FileName:PdfDocument.java**

package document;

public interface PdfDocument{

void openpdf();

}

**FileName:MyWordDocument.java**

package document;

public class MyWordDocument implements WordDocument {

public void openword(){

System.out.println("Opening Word Document");

}

}

**FileName:MyPdfDocument.java**

package document;

public class MyPdfDocument implements PdfDocument {

public void openpdf(){

System.out.println("Opening Pdf Document");

}

}

**FileName:MyExcelDocument.java**

package document;

public class MyExcelDocument implements ExcelDocument {

public void openExcel(){

System.out.println("Opening Excel Document");

}

}

**FileName: DocumentFactory.java**

package factories;

public abstract class DocumentFactory {

public abstract Object createDocument();

}

**FileName :ExcelFactory.java**

package factories;

import document.\*;

public class ExcelFactory extends DocumentFactory {

public MyExcelDocument createDocument() {

return new MyExcelDocument();

}

}

**FileName:PdfFactory.java**

package factories;

import document.\*;

public class PdfFactory extends DocumentFactory {

public MyPdfDocument createDocument() {

return new MyPdfDocument();

}

}

**FileName:WordFactory**

package factories;

import document.\*;

public class WordFactory extends DocumentFactory {

public MyWordDocument createDocument() {

return new MyWordDocument();

}

}

**FileName:Main.java**

import factories.\*;

import document.\*;

public class Main {

public static void main(String[] args) {

WordFactory wordFactory = new WordFactory();

WordDocument wordDoc = wordFactory.createDocument();

wordDoc.openword();

PdfFactory pdfFactory = new PdfFactory();

PdfDocument pdfDoc = pdfFactory.createDocument();

pdfDoc.openpdf();

ExcelFactory excelFactory = new ExcelFactory();

ExcelDocument excelDoc = excelFactory.createDocument();

excelDoc.openExcel();

}

}

**Output :**

