**Design Patterns and Principles**

**Exercise 1: Implementing the Singleton Pattern**

File 1: Logger.java (Implementing the singleton pattern)

public class Logger {

    // Private static instance of Logger (singleton)

    private static Logger instance;

    // Private constructor to prevent instantiation from outside

    private Logger() {

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

    }

    // Public method to provide access to the single instance

    public static Logger getInstance() {

        if (instance == null) {

            instance = new Logger(); // Lazy initialization

        }

        return instance;

    }

    // Method to log a message

    public void log(String message) {

        System.out.println("[LOG] " + message);

    }

}

File 2: Main.java (Testing the singleton behaviour)

public class Main {

    public static void main(String[] args) {

        // Get the first instance of Logger

        Logger logger1 = Logger.getInstance();

        logger1.log("Application started.");

        // Get the second instance of Logger

        Logger logger2 = Logger.getInstance();

        logger2.log("Performing some operation...");

        // Get the third instance of Logger

        Logger logger3 = Logger.getInstance();

        logger3.log("Operation completed.");

        // Check if all instances are the same

        if (logger1 == logger2 && logger2 == logger3) {

            System.out.println("All logger instances are the same. Singleton works!");

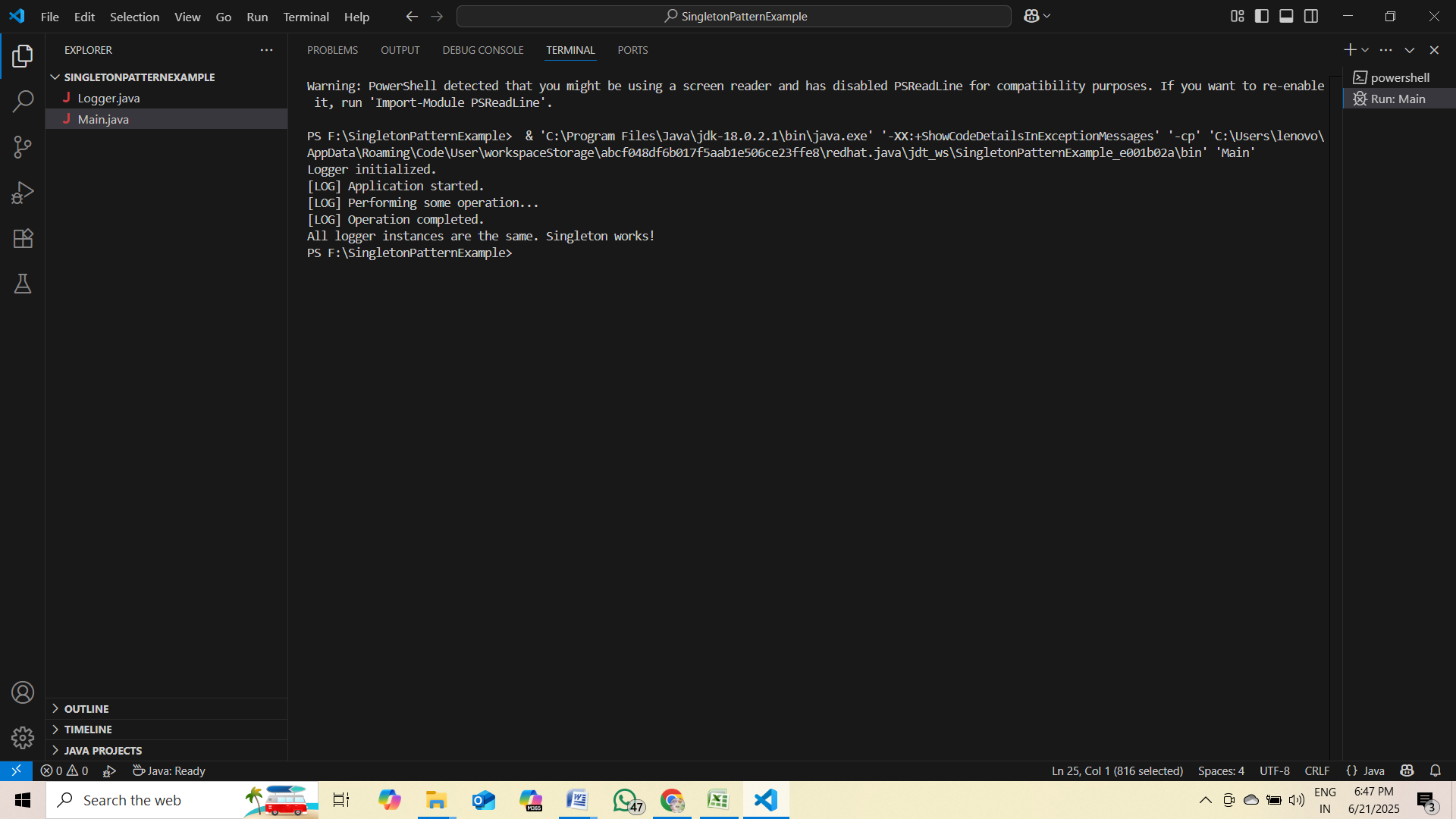
        } else {

            System.out.println("Different instances exist. Singleton failed!");

        }

    }

}

OUTPUT

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

public class FactoryMethodPatternExample {

    public static void main(String[] args) {

        DocumentFactory wordFactory = new WordDocumentFactory();

        Document word = wordFactory.createDocument();

        word.open();

        DocumentFactory pdfFactory = new PdfDocumentFactory();

        Document pdf = pdfFactory.createDocument();

        pdf.open();

        DocumentFactory excelFactory = new ExcelDocumentFactory();

        Document excel = excelFactory.createDocument();

        excel.open();

    }

}

interface Document {

    void open();

}

class WordDocument implements Document {

    public void open() {

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

    }

}

class PdfDocument implements Document {

    public void open() {

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

    }

}

class ExcelDocument implements Document {

    public void open() {

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

    }

}

abstract class DocumentFactory {

    public abstract Document createDocument();

}

class WordDocumentFactory extends DocumentFactory {

    public Document createDocument() {

        return new WordDocument();

    }

}

class PdfDocumentFactory extends DocumentFactory {

    public Document createDocument() {

        return new PdfDocument();

    }

}

class ExcelDocumentFactory extends DocumentFactory {

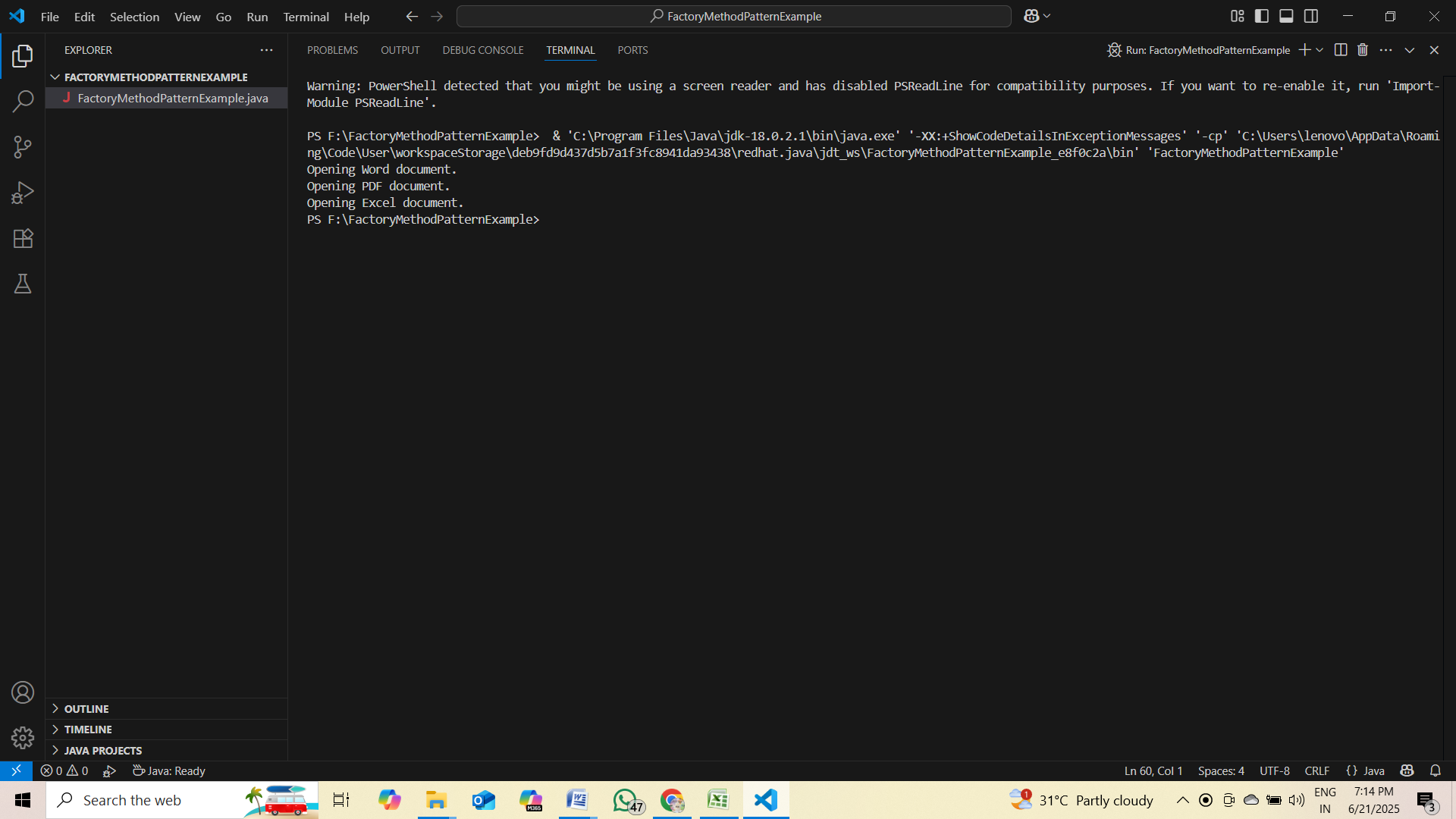
    public Document createDocument() {

        return new ExcelDocument();

    }

}

**OUTPUT**

****