**Design Principles and Patterns**

**Exercise 1: Implementing the Singleton Pattern**

**Class 1 : Logger.java**

public class Logger {

    // Private static instance for Singleton implementation

    private static Logger uniqueInstance

    // Private constructor to restrict instantiation

    private Logger() {

        System.out.println("Logger instance created.");

    }

    public static Logger getInstance() {

        if (uniqueInstance == null) {

            uniqueInstance = new Logger();

        }

        return uniqueInstance;

    }

    public void log(String message) {

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

    }

}

**Class 2 : LoggerTest.java**

public class LoggerTest {

    public static void main(String[] args) {

        Logger firstLogger = Logger.getInstance();

        firstLogger.log("First logger message.");

        Logger secondLogger = Logger.getInstance();

        secondLogger.log("Second logger message.");

        // Verify Singleton behavior

        if (firstLogger == secondLogger) {

            System.out.println("Same logger instance confirmed.");

            System.out.println("Singleton pattern successfully implemented.");

        } else {

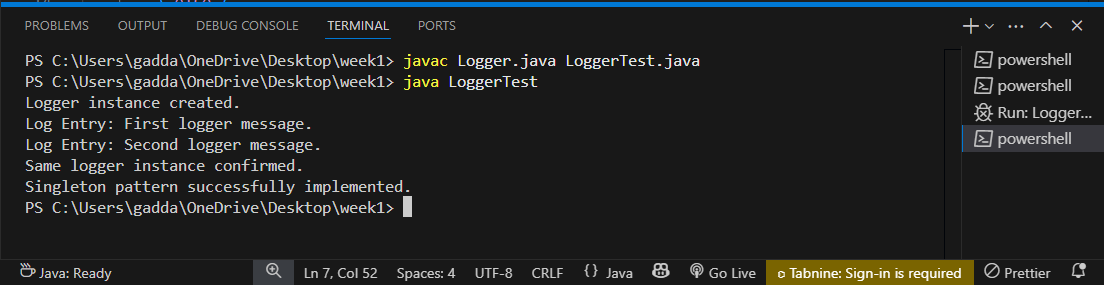
            System.out.println("Singleton implementation failed.");

        }

    }

}

**Output:**

****

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

**Class 1 : Document.java**

public interface Document {

    void open();

}

**Class 2 : WordDocument.java**

public class WordDocument implements Document {

    @Override

    public void open() {

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

    }

}

**Class 3 : PdfDocument.java**

public class PdfDocument implements Document {

    @Override

    public void open() {

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

    }

}

**Class 4 : ExcelDocument.java**

public class ExcelDocument implements Document {

    @Override

    public void open() {

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

    }

}

**Class 5 : DocumentFactory.java**

public abstract class DocumentFactory {

    public abstract Document createDocument();

}

**Class 6 : WordDocumentFactory.java**

public class WordDocumentFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new WordDocument();

    }

}

**Class 7 : PdfDocumentFactory.java**

public class PdfDocumentFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new PdfDocument();

    }

}

**Class 8 : ExcelDocumentFactory.java**

public class ExcelDocumentFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new ExcelDocument();

    }

}

**Class 9 : Main.java**

public class Main {

    public static void main(String[] args) {

        // Create Word Document

        DocumentFactory wordFactory = new WordDocumentFactory();

        Document wordDoc = wordFactory.createDocument();

        wordDoc.open();

        // Create PDF Document

        DocumentFactory pdfFactory = new PdfDocumentFactory();

        Document pdfDoc = pdfFactory.createDocument();

        pdfDoc.open();

        // Create Excel Document

        DocumentFactory excelFactory = new ExcelDocumentFactory();

        Document excelDoc = excelFactory.createDocument();

        excelDoc.open();

    }

}

**Output :**

