**DESIGN PATTERNS & PRINCIPLES**

**Exercise : 1 - Implementing the Singleton Pattern**

**Code1: Logger.java**

public class Logger {

private static volatile Logger instance;

private int logCount;

private Logger() {

logCount = 0;

System.out.println("New Logger instance created. Total instances: 1");

}

public static Logger getInstance() {

if (instance == null) {

synchronized (Logger.class) {

if (instance == null) {

instance = new Logger();

}

}

}

return instance;

}

public void log(String message) {

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

}

public int getLogCount() {

return logCount;

}

}

**Code2: LoggerTest.java**

public class LoggerTest {

public static void main(String[] args) {

Logger loggerA = Logger.getInstance();

Logger loggerB = Logger.getInstance();

System.out.println("Are both logger references the same? " + (loggerA == loggerB));

loggerA.log("Application started");

loggerB.log("Loading configuration");

new Thread(() -> {

Logger threadLogger = Logger.getInstance();

threadLogger.log("Background task running");

}).start();

// More logging

loggerA.log("User logged in");

loggerB.log("Data loaded");

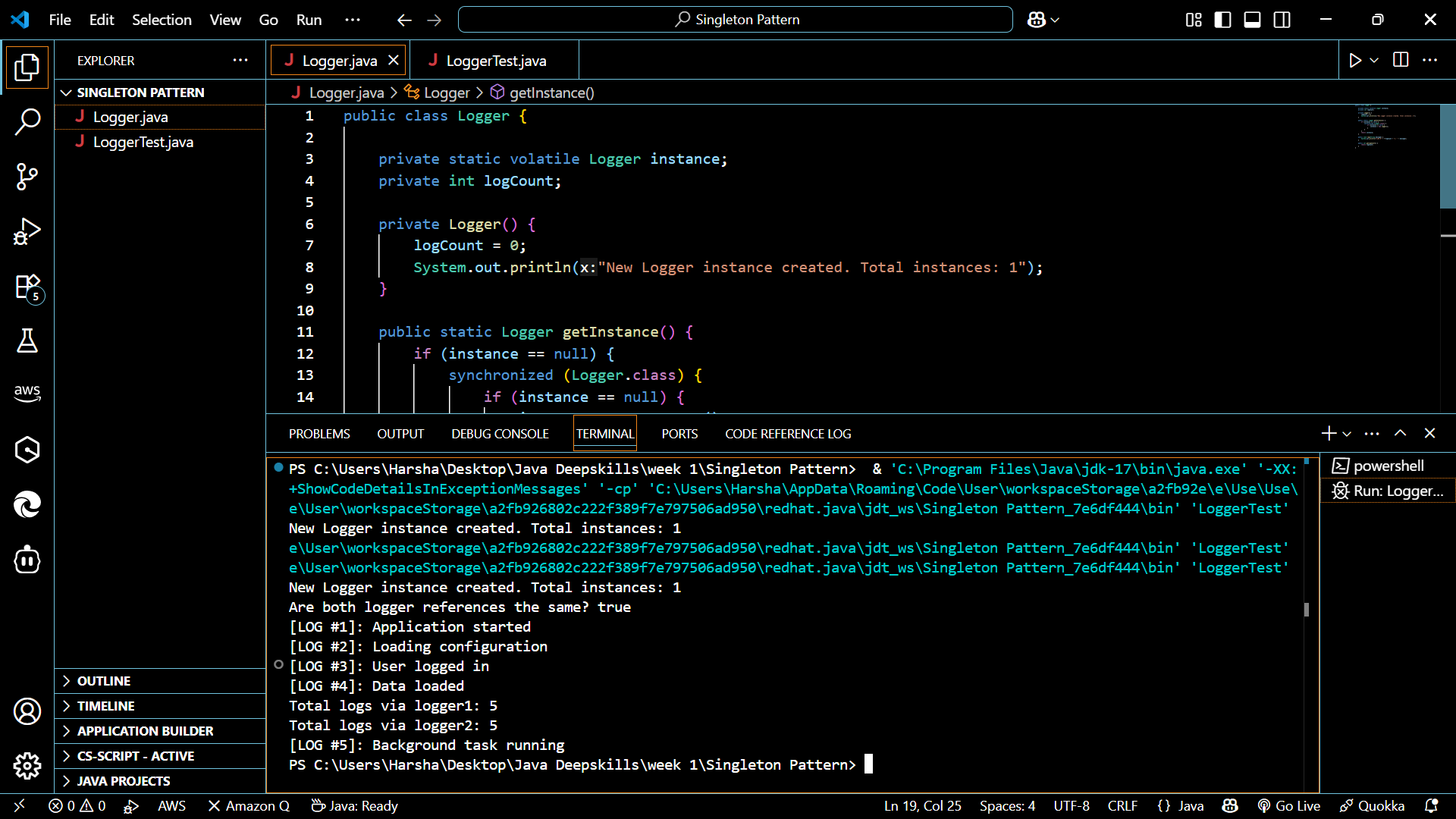
System.out.println("Total logs via logger1: " + loggerA.getLogCount());

System.out.println("Total logs via logger2: " + loggerB.getLogCount());

}

}

**Output:**

****

**EXERCISE : 2 - Implementing the Factory Method Pattern**

**Code1: Document.java**  
public interface Document {

void open();

void save();

void close();

String getDocumentType();

void addContent(String content);

void displayMetadata();

}

**Code2: ExcelDocument.java**

import java.util.HashMap;

import java.util.Map;

public class ExcelDocument implements Document {

private Map<String, String[]> sheets = new HashMap<>();

private String currentSheet = "Sheet\_1";

private int rowCount = 1;

public ExcelDocument() {

sheets.put(currentSheet, new String[10000]);

}

@Override

public void open() {

System.out.println("Opening Excel\_doc - Enabling macros");

}

@Override

public void save() {

System.out.println("Saving Excel\_doc - Calculating formulas");

}

@Override

public void close() {

System.out.println("Closing Excel\_doc - Saving workbook");

}

@Override

public String getDocumentType() {

return "Microsoft Excel (.xlsx)";

}

@Override

public void addContent(String content) {

String[] rows = sheets.get(currentSheet);

if (rows != null && rowCount < rows.length) {

rows[rowCount++] = content;

}

}

@Override

public void displayMetadata() {

System.out.println("Excel Document Metadata:");

System.out.println("Sheets: " + sheets.size());

System.out.println("Active\_Sheet: " + currentSheet);

System.out.println("Rows in current sheet: " + rowCount);

}

public void addSheet(String sheetName) {

sheets.put(sheetName, new String[10000]);

}

}

**Code3: PdfDocument.java**

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

public class PdfDocument implements Document {

private String title;

private LocalDateTime creationDate;

private boolean isSecured;

public PdfDocument() {

this.title = "Untitled PDF";

this.creationDate = LocalDateTime.now();

this.isSecured = false;

}

@Override

public void open() {

System.out.println("Opening PDF\_doc in read-only mode");

}

@Override

public void save() {

System.out.println("Saving PDF\_doc - Finalizing formatting");

}

@Override

public void close() {

System.out.println("Closing PDF\_doc - Preserving annotations");

}

@Override

public String getDocumentType() {

return "Portable Document (.pdf)";

}

@Override

public void addContent(String content) {

System.out.println("Adding content to PDF: " + content);

}

@Override

public void displayMetadata() {

System.out.println("PDF Document Metadata:");

System.out.println("Title: " + title);

System.out.println("Created: " +

creationDate.format(DateTimeFormatter.ISO\_LOCAL\_DATE\_TIME));

System.out.println("Secured: " + (isSecured ? "Yes" : "No"));

}

public void setSecured(boolean secured) {

isSecured = secured;

}

}

**Code4: WordDocument.java**

import java.util.ArrayList;

import java.util.List;

public class WordDocument implements Document {

private List<String> content = new ArrayList<>();

private String author;

private int pageCount;

public WordDocument() {

this.author = System.getProperty("user.name");

this.pageCount = 1;

}

@Override

public void open() {

System.out.println("Opening Word\_doc - Compatibility Mode");

}

@Override

public void save() {

System.out.println("Saving Word\_doc with " + pageCount + " pages");

}

@Override

public void close() {

System.out.println("Closing Word\_doc - Remember to save changes");

}

@Override

public String getDocumentType() {

return "Microsoft Word\_doc (.docx)";

}

@Override

public void addContent(String content) {

this.content.add(content);

pageCount = (int) Math.ceil(this.content.size() / 20.0);

}

@Override

public void displayMetadata() {

System.out.println("Word Document Metadata:");

System.out.println("Author: " + author);

System.out.println("Pages: " + pageCount);

System.out.println("Lines: " + content.size());

}

}

**Code5: DocumentFactory.java**

public abstract class DocumentFactory {

public abstract Document createDocument();

public void prepareDocument() {

Document doc = createDocument();

doc.open();

doc.addContent("Content for doc");

System.out.println("Created: " + doc.getDocumentType());

}

}

**Code6: ExcelDocumentFactory.java**

public class ExcelDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

ExcelDocument excel = new ExcelDocument();

excel.addSheet("Data");

excel.addSheet("Analytics");

return excel;

}

}

**Code7: PdfDocumentFactory.java**

public class PdfDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

PdfDocument pdf = new PdfDocument();

pdf.setSecured(true);

return pdf;

}

}

**Code8: WordDocumentFactory.java**

public class WordDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new WordDocument();

}

}

**Code9: DocumentManagementSystem.java**

public class DocumentManagementSystem {

public static void main(String[] args) {

System.out.println("Document\_Management\_System starting..\n");

DocumentFactory WordFactory = new WordDocumentFactory();

DocumentFactory PdfFactory = new PdfDocumentFactory();

DocumentFactory ExcelFactory = new ExcelDocumentFactory();

Document wordDoc = WordFactory.createDocument();

Document pdfDoc = PdfFactory.createDocument();

Document excelDoc = ExcelFactory.createDocument();

System.out.println("\nTesting Word\_document:");

wordDoc.addContent("We should write content for Word document.");

wordDoc.displayMetadata();

System.out.println("\nTesting PDF\_document:");

pdfDoc.addContent("Highly confidential content for PDF.");

pdfDoc.displayMetadata();

System.out.println("\nTesting Excel\_document:");

excelDoc.addContent("A1=Profit, B1=60000");

excelDoc.displayMetadata();

System.out.println("\nPreparing documents using Factory\_Methods:");

WordFactory.prepareDocument();

PdfFactory.prepareDocument();

ExcelFactory.prepareDocument();

}

}

**Output:**

