Week 1-Design Patterns:

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

class Logger {

private static Logger instance;

private Logger() {}

public static synchronized Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

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

}

}

public class Main {

public static void main(String[] args) {

Logger loggerA = Logger.getInstance();

Logger loggerB = Logger.getInstance();

System.out.println("loggerA == loggerB? " + (loggerA == loggerB));

loggerA.log("First message");

loggerB.log("Second message");

}

}

Output:



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

interface Document {

void open();

}

class WordDocument implements Document {

public void open() {

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

}

}

class PdfDocument implements Document {

public void open() {

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

}

}

class ExcelDocument implements Document {

public void open() {

System.out.println("Opening an 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();

}

}

public class Main {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordDocumentFactory();

DocumentFactory pdfFactory = new PdfDocumentFactory();

DocumentFactory excelFactory = new ExcelDocumentFactory();

Document doc1 = wordFactory.createDocument();

doc1.open();

Document doc2 = pdfFactory.createDocument();

doc2.open();

Document doc3 = excelFactory.createDocument();

doc3.open();

}

}

