**WEEK - 1**

**Module 1 - Design Patterns and Principles:**

**Exercise 1: Implementing the Singleton Pattern**

Step 1: Create a New Java Project

Create a new Java project named SingletonPatternExample in your preferred IDE (like IntelliJ IDEA, Eclipse, etc.).

Step 2: Define a Singleton Class

Create a class named Logger that implements the Singleton design pattern.

Step 3: Implement the Singleton Pattern

The implementation above ensures that the Logger class follows the Singleton design pattern by:

• Having a private constructor to prevent instantiation from outside the class.

• Providing a static method getInstance() that returns the single instance of the class.

Step 4: Test the Singleton Implementation

Create a test class to verify that only one instance of Logger is created and used across the application.

**CODE:**

**public class Main {**

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

**Logger logger1 = Logger.getInstance();**

**logger1.log("First message");**

**Logger logger2 = Logger.getInstance();**

**logger2.log("Second message");**

**System.out.println("Same instance? " + (logger1 == logger2));**

**}**

**}**

**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;**

**}**

**public void log(String message) {**

**System.out.println("LOG: " + message);**

**}**

**}**

**OUTPUT:**

****

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

1. Project Setup
   * Single-file Java implementation suitable for online compilers
   * Contains all required components in one executable class
2. Document Interface
   * Created Document interface with open(), save(), close() methods
   * Serves as common contract for all document types
3. Concrete Document Classes
   * WordDocument: Implements PDF operations
   * PdfDocument: Implements PDF operations
   * ExcelDocument: Implements Excel operations
   * Each provides specific implementation of interface methods
4. Factory Implementation
   * Abstract DocumentFactory class with createDocument() method
   * Concrete factories:
     + WordDocumentFactory: Creates Word documents
     + PdfDocumentFactory: Creates PDF documents
     + ExcelDocumentFactory: Creates Excel documents
5. Testing
   * Main class demonstrates:
     + Factory instantiation for each document type
     + Document creation through factories
     + Method execution on each document
   * Output shows proper type-specific behavior

**CODE:**

**public class FactoryMethodPatternExample {**

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

**DocumentFactory wordFactory = new WordDocumentFactory();**

**Document wordDoc = wordFactory.createDocument();**

**wordDoc.open();**

**wordDoc.save();**

**wordDoc.close();**

**DocumentFactory pdfFactory = new PdfDocumentFactory();**

**Document pdfDoc = pdfFactory.createDocument();**

**pdfDoc.open();**

**pdfDoc.save();**

**pdfDoc.close();**

**DocumentFactory excelFactory = new ExcelDocumentFactory();**

**Document excelDoc = excelFactory.createDocument();**

**excelDoc.open();**

**excelDoc.save();**

**excelDoc.close();**

**}**

**}**

**interface Document {**

**void open();**

**void save();**

**void close();**

**}**

**class WordDocument implements Document {**

**public void open() { System.out.println("Word Document: Opening..."); }**

**public void save() { System.out.println("Word Document: Saving..."); }**

**public void close() { System.out.println("Word Document: Closing..."); }**

**}**

**class PdfDocument implements Document {**

**public void open() { System.out.println("PDF Document: Opening..."); }**

**public void save() { System.out.println("PDF Document: Saving..."); }**

**public void close() { System.out.println("PDF Document: Closing..."); }**

**}**

**class ExcelDocument implements Document {**

**public void open() { System.out.println("Excel Document: Opening..."); }**

**public void save() { System.out.println("Excel Document: Saving..."); }**

**public void close() { System.out.println("Excel Document: Closing..."); }**

**}**

**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(); }**

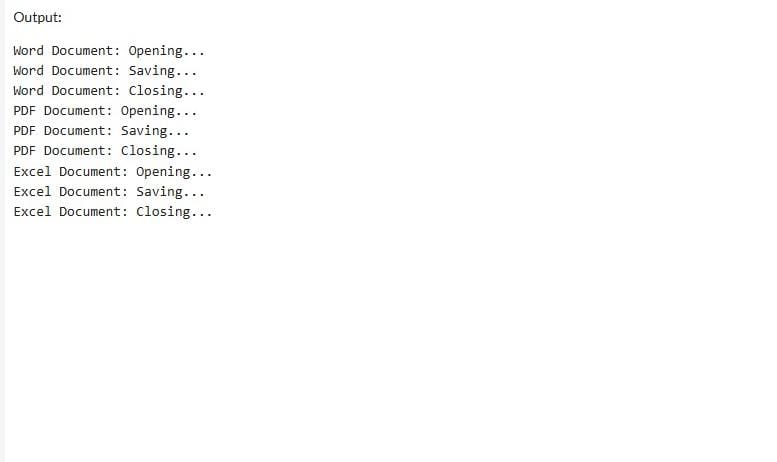
**}**

**class ExcelDocumentFactory extends DocumentFactory {**

**public Document createDocument() { return new ExcelDocument(); }**

**}**

**OUTPUT:**

****