Week 1: Design Pattern & Principles

**Exercise 1: Singleton Pattern**

**Scenario:**

You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

# Create a New Java Project

# Create a new Java project named SingletonPatternExample.

# Define a Singleton Class

# public class SingletonTest {

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

# Logger l1 = Logger.getInstance();

# Logger l2 = Logger.getInstance();

# l1.log("First log message");

# l2.log("Second log message");

# if (l1 == l2) {

# System.out.println("Only one instance exists. Singleton is working.");

# } else {

# System.out.println("Multiple instances created. Singleton failed.");

# }

# }

# }

**Implement the Singleton Pattern & Test the implementation**

# public class Logger {

# private static Logger instance;

# private Logger() { //Ensure the constructor of Logger class is private.

# System.out.println("Logger Initialized");

# }

# public static Logger getInstance() {

# if (instance == null) {

# instance = new Logger();

# }

# return instance;

# }

# public void log(String msg) {

# System.out.println("Log: " + msg);

# }

# }

A computer screen with text

AI-generated content may be incorrect.**Output**

Week 1: Design Pattern & Principles

**Exercise 2: Factory Method Pattern**

**Scenario:**

You are developing a document management system that needs to create different types of documents (e.g., Word, PDF, Excel). Use the Factory Method Pattern to achieve this.

# Create a New Java Project

# Create a new Java project named FactoryMethodPattern.

# Define the Document classes

# public interface Document {

# void open();

# }

# public class PdfDocument implements Document {

# public void open() {

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

# }

# }

# public class ExcelDocument implements Document {

# public void open() {

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

# }

# }

# public class WordDocument implements Document {

# public void open() {

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

# }

# }

**Create Concrete Document Classes**

# public class PdfDocumentFactory extends DocumentFactory {

# public Document createDocument() {

# return new PdfDocument();

# }

# }

# 

# public class ExcelDocumentFactory extends DocumentFactory {

# public Document createDocument() {

# return new ExcelDocument();

# }

# }

# public class WordDocumentFactory extends DocumentFactory {

# public Document createDocument() {

# return new WordDocument();

# }

# }

**Implement the Factory Method**

public abstract class DocumentFactory {

public abstract Document createDocument();

}

**Test the Factory Method Implementation (Main Class)**

public class TestFactoryPattern {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordDocumentFactory();

Document wordDoc = wordFactory.createDocument();

System.out.println(">>> Creating a Word document...");

wordDoc.open();

System.out.println(">>> Closing Word document.\n");

DocumentFactory pdfFactory = new PdfDocumentFactory();

Document pdfDoc = pdfFactory.createDocument();

System.out.println(">>> Creating a PDF document...");

pdfDoc.open();

System.out.println(">>> Closing PDF document.\n");

DocumentFactory excelFactory = new ExcelDocumentFactory();

Document excelDoc = excelFactory.createDocument();

System.out.println(">>> Creating an Excel document...");

excelDoc.open();

System.out.println(">>> Closing Excel document.");

}

}

**Output**

A screen shot of a computer

AI-generated content may be incorrect.