Harshitha M

GSSSIETW

**Week : 1**

**1)Design Patterns and Principles**

**Exercise 1: Implementing the Singleton Pattern**

**CODE :**

**1)Create new java class : Logger (Define a Singleton Class)**

public class Logger {

private static Logger instance;

private Logger() {

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

}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

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

}

}

2) **Create new java class : Main (Test the Singleton Implementation)**

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");

if (logger1 == logger2) {

System.out.println("Both logger instances are the same (Singleton works).");

} else {

System.out.println("Different instances (Singleton failed).");

}

}

}

**Output :**

A screenshot of a computer program

AI-generated content may be incorrect.

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

**CODE:**

1. Define Document Interface

public interface Document {

void open();

}

1. Create Concrete Document Classes

**WordDocument.java**

public class WordDocument implements Document {

@Override

public void open() {

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

}

}

**PdfDocument.java**

public class PdfDocument implements Document {

@Override

public void open() {

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

}

}

**ExcelDocument.java**

public class ExcelDocument implements Document {

@Override

public void open() {

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

}

}

1. Implement the Factory Method

Create an abstract factory class:

**DocumentFactory.java**

public abstract class DocumentFactory {

public abstract Document createDocument();

}

create concrete factories for each document type:

**WordDocumentFactory.java**

public class WordDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new WordDocument();

}

}

**PdfDocumentFactory.java**

public class PdfDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new PdfDocument();

}

}

**ExcelDocumentFactory.java**

public class ExcelDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new ExcelDocument();

}

}

1. Test the Factory Method Implementation

**Main.java**

public class Main {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordDocumentFactory();

DocumentFactory pdfFactory = new PdfDocumentFactory();

DocumentFactory excelFactory = new ExcelDocumentFactory();

Document wordDoc = wordFactory.createDocument();

Document pdfDoc = pdfFactory.createDocument();

Document excelDoc = excelFactory.createDocument();

wordDoc.open();

pdfDoc.open();

excelDoc.open();

}

}

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**