**Design Principles and Patterns**

**Exercise 1: Singleton Pattern**

**Objective:**  
Ensure that a logging utility class in the application has only one instance throughout the application lifecycle, providing consistent and centralized logging.

**Step-by-Step Implementation:**

**1. Project Setup:**

* File Name: SingletonLoggerDemo.java

**2. Code:**

public class SingletonLoggerDemo {

// Static nested Singleton Logger class

static 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);

}

}

// Main method to test singleton behavior

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

logger1.log("This is the first log message.");

logger2.log("This is the second log message.");

if (logger1 == logger2) {

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

} else {

System.out.println("Logger instances are different (singleton failed!).");

}

}

}

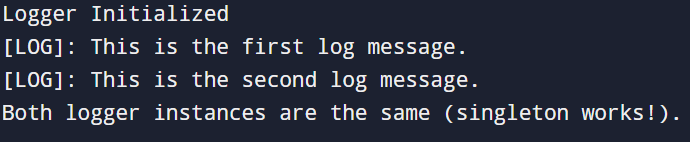
**Expected Output:**

Logger Initialized

[LOG]: This is the first log message.

[LOG]: This is the second log message.

Both logger instances are the same (singleton works!).

**Screenshots:**  


**Exercise 2: Singleton Pattern**

**Objective:** To create different types of documents (Word, PDF, Excel) using the Factory Method Pattern, allowing for flexible and scalable object creation.

**Step-by-Step Implementation:**

**1. Project Setup:**

* Project Name: FactoryMethodPatternExample

**2. Define Document Interface:**

interface Document {

void open();

}

**3. Create Concrete Document Classes:**

class WordDocument implements Document {

public void open() {

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

}

}

class PdfDocument implements Document {

public void open() {

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

}

}

class ExcelDocument implements Document {

public void open() {

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

}

}

**4. Implement the Factory Method:**

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();

}

}

**5. Testing the Factory Method Implementation:**

public class Main {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordDocumentFactory();

Document word = wordFactory.createDocument();

word.open();

DocumentFactory pdfFactory = new PdfDocumentFactory();

Document pdf = pdfFactory.createDocument();

pdf.open();

DocumentFactory excelFactory = new ExcelDocumentFactory();

Document excel = excelFactory.createDocument();

excel.open();

}

}

**Expected Output:**

Opening Word document...

Opening PDF document...

Opening Excel document...

**Screenshots:**

