WEEK-1 TASKS

# DESIGN PATTERNS AND PRINCIPLES

**EXERCISE 1: IMPLEMENTING THE SINGLETON PATTERN**

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

# Steps:

1. create a new java project
2. define a singleton class

# implement the singleton pattern

# test the singleton implementation

**LOGGER:**

package singleton;

public class Logger {

private static Logger instance;

private Logger() {

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

}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger(); // Lazy initialization

}

return instance;

}

public void log(String message) {

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

}

**MAINLOGGERTEST.JAVA:**

package singleton;

public class MainLoggerTest {

public static void main(String[] args) {

Logger loggerA = Logger.getInstance();

Logger loggerB = Logger.getInstance();

loggerA.log("First log from loggerA");

loggerB.log("Second log from loggerB");

if (loggerA == loggerB) {

System.out.println("Success: Both references point to the same Logger instance.");

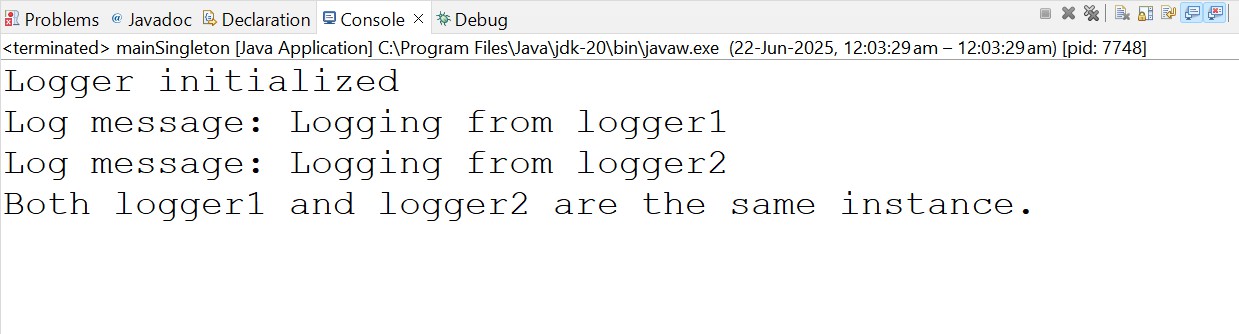
} else {

System.out.println("Failure: Different Logger instances were created.");

}

}

}



**EXERCISE 2: IMPLEMENTING THE FACTORY METHOD PATTERN**

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.

# Steps:

1. Create a New Java Project
2. Define Document Classes

# Create Concrete Document Classes

# Implement the Factory Method

# Test the Factory Method Implementation

# 

#### Document.java:

java

CopyEdit

package factory;

public interface Document {

void open();

}

#### DocumentFactory.java:

java

CopyEdit

package factory;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

#### WordDocument.java:

java

CopyEdit

package factory;

public class WordDocument implements Document {

@Override

public void open() {

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

}

}

#### PdfDocument.java:

java

CopyEdit

package factory;

public class PdfDocument implements Document {

@Override

public void open() {

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

}

}

#### ExcelDocument.java:

java

CopyEdit

package factory;

public class ExcelDocument implements Document {

@Override

public void open() {

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

}

}

#### WordDocumentFactory.java:

java

CopyEdit

package factory;

public class WordDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new WordDocument();

}

}

#### PdfDocumentFactory.java:

java

CopyEdit

package factory;

public class PdfDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new PdfDocument();

}

}

#### ExcelDocumentFactory.java:

java

CopyEdit

package factory;

public class ExcelDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new ExcelDocument();

}

}

#### MainFactoryDemo.java:

java

CopyEdit

package factory;

public class MainFactoryDemo {

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();

}

}

