WEEK-1 TASKS

# DESIGN PATTERNS AND PRINCIPLES

**Exercise 1: Implementing the 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.

# Steps:

1. **Create a New Java Project:**
   * Create a new Java project named

# SingletonPatternExample.

1. **Define a Singleton Class:**
   * Create a class named Logger that has a private static instance of itself.
   * Ensure the constructor of Logger is private.
   * Provide a public static method to get the instance of the Logger class.

# Implement the Singleton Pattern:

* + Write code to ensure that the Logger class follows the Singleton design pattern.

# Test the Singleton Implementation:

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

**Logger.java**

package Singlton;

public class Logger {

private static Logger *insta*;

private Logger() {

System.***out***.println("Logger initailized");

}

public static Logger getInstance() {

if (*insta*==null) {

*insta*=new Logger();

}

return *insta*;

}

public void log(String message) {

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

}

}

**Main.java**

package Singlton;

public class Main {

public static void main(String[] args) {

Logger log1=Logger.*getInstance*();

log1.log("this is first log message");

Logger log2=Logger.*getInstance*();

log2.log("this is second log message");

if(log1==log2) {

System.***out***.println("Both log instance are same.. So singleton works");

}

else {

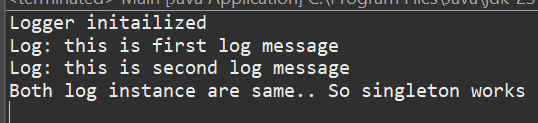
System.***out***.println("Logger instance are different. Singleton failed!!!");

}

}

}

**Output:**



**Exercise 2: Implementing the 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.

# Steps:

1. **Create a New Java Project:**
   * Create a new Java project named

# FactoryMethodPatternExample.

1. **Define Document Classes:**
   * Create interfaces or abstract classes for different document types such as **WordDocument**, **PdfDocument**, and **ExcelDocument**.

# Create Concrete Document Classes:

* + Implement concrete classes for each document type that implements or extends the above interfaces or abstract classes.

# Implement the Factory Method:

* + Create an abstract class **DocumentFactory** with a method **createDocument()**.
  + Create concrete factory classes for each document type that extends DocumentFactory and implements the **createDocument()** method.

# Test the Factory Method Implementation:

* + Create a test class to demonstrate the creation of different document types using the factory method.

#### Document.java ****(Interface)****

package FactoryMethods;

public interface Document {

void open();

}

##### ****WordDocument.java****

package FactoryMethods;

public class WordDocument implements Document {

@Override

public void open() {

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

}

}

##### ****PdfDocument.java****

package FactoryMethods;

public class PdfDocument implements Document {

@Override

public void open() {

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

}

}

##### ****ExcelDocument.java****

package FactoryMethods;

public class ExcelDocument implements Document {

@Override

public void open() {

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

}

}

##### ****DocumentFactory.java****

package FactoryMethods;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

##### ****WordDocumentFactory.java****

package FactoryMethods;

public class WordDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new WordDocument();

}

}

##### ****PdfDocumentFactory.java****

package FactoryMethods;

public class PdfDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new PdfDocument();

}

}

##### ****ExcelDocumentFactory.java****

package FactoryMethods;

public class ExcelDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new ExcelDocument();

}

}

#### ****Main.java****

package FactoryMethods;

public class Main {

public static void main(String[] args) {

// Create a Word document

DocumentFactory wordFactory = new WordDocumentFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

// Create a PDF document

DocumentFactory pdfFactory = new PdfDocumentFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

// Create an Excel document

DocumentFactory excelFactory = new ExcelDocumentFactory();

Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

**Output:**

