**Week 1 (Design Pattern)**

**Exercise 1: Implementing the Singleton Pattern**

**//Code**

**//Logger.java**

public class Logger {

    // Private static instance of Logger (eager initialization or lazy - see below)

    private static Logger instance;

    // Private constructor to prevent instantiation from outside

    private Logger() {

        System.out.println("Logger instance created.");

    }

    // Public static method to return the same instance

    public static Logger getInstance() {

        if (instance == null) {

            instance = new Logger(); // Only creates once

        }

        return instance;

    }

    // Example logging method

    public void log(String message) {

        System.out.println("[LOG] " + message);

    }

}

**//Main.java**

public class Main {

    public static void main(String[] args) {

        // Try to get multiple Logger instances

        Logger logger1 = Logger.getInstance();

        Logger logger2 = Logger.getInstance();

        // Test logging

        logger1.log("Application started.");

        logger2.log("Application running.");

        // Check if both instances are the same

        if (logger1 == logger2) {

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

        } else {

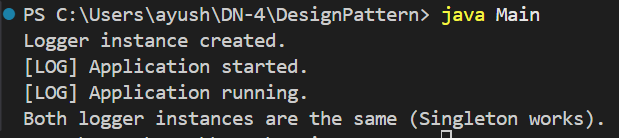
            System.out.println("Different instances (Singleton failed).");

        }

    }

}

**//Output:**



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

Project name: **FactoryMethodPatternExample**

**document/ package**

**//Document.java**

package document;

public interface Document {

void open();

}

**//WordDocument.java**

package document;

public class WordDocument implements Document {

public void open() {

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

}

}

**//PdfDocument.java**

package document;

public class PdfDocument implements Document {

public void open() {

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

}

}

**//ExcelDocument.java**

package document;

public class ExcelDocument implements Document {

public void open() {

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

}

}

**factory/ package**

**//DocumentFactory.java**

package factory;

import document.Document;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

**//WordDocumentFactory.java**

package factory;

import document.Document;

import document.WordDocument;

public class WordDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

**//PdfDocumentFactory.java**

package factory;

import document.Document;

import document.PdfDocument;

public class PdfDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

**//ExcelDocumentFactory.java**

package factory;

import document.Document;

import document.ExcelDocument;

public class ExcelDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

**//Main.java**

import factory.\*;

import document.Document;

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();

}

}

**Output:**

