**Digital Nurture 4.0 – Week 1**

**1.Design principles & Patterns**

**Exercise 1:** Implementing the Singleton Pattern

**Program:**

I created a project named Logger and a separate Main class to verify that only one instance of Logger is created and used across the application.

**Logger.java**

public class Logger {

    private static Logger singleInstance;

    private Logger() {

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

    }

    public static Logger getInstance() {

        if (singleInstance == null) {

            singleInstance = new Logger();

        }

        return singleInstance;

    }

    public void log(String message) {

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

    }

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        Logger logger1 = Logger.getInstance();

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

        Logger logger2 = Logger.getInstance();

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

        if (logger1 == logger2) {

            System.out.println("Both logger1 and logger2 are the same instance.");

        } else {

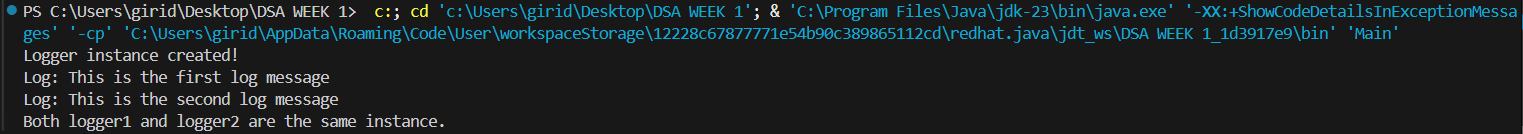
            System.out.println("Different instances exist! Singleton failed.");

        }

    }

}

**Output:**

****

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

**Program:**

I created a project and package named FactoryMethodPatternExample and a separate test class to demonstrate the creation of different types of documents (Word, PDF, Excel) using the Factory Method design pattern.

**FactoryMethodPatternExample.java**

package FactoryMethodPatternExample;

interface Document {

    void open();

}

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...");

    }

}

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();

    }

}

public class FactoryMethodPatternExample {

    public static void main(String[] args) {

        DocumentFactory wordFactory = new WordDocumentFactory();

        Document wordDoc = wordFactory.createDocument();

        wordDoc.open();

        DocumentFactory pdfFactory = new PdfDocumentFactory();

        Document pdfDoc = pdfFactory.createDocument();

        pdfDoc.open();

        DocumentFactory excelFactory = new ExcelDocumentFactory();

        Document excelDoc = excelFactory.createDocument();

        excelDoc.open();

    }

}

**Output:**

