**WEEK-1**

**Design Patterns and Principles**

**Exercise 1: Implementing the Singleton Pattern**

**Main.java:**

package com.singleton;

public class Main {

    public static void main(String[] args) {

        // Get two instances of Logger

        Logger logger1 = Logger.getInstance();

        Logger logger2 = Logger.getInstance();

        // Log messages

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

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

        // Verify if both instances are the same

        if (logger1 == logger2) {

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

        } else {

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

        }

    }

}

**Logger.java:**

package com.singleton;

public class Logger {

    // Private static instance of Logger

    private static Logger instance;

    // Private constructor to prevent instantiation

    private Logger() {

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

    }

    // Public static method to return the singleton instance

    public static Logger getInstance() {

        if (instance == null) {

            // Lazy initialization

            instance = new Logger();

        }

        return instance;

    }

    // Example logging method

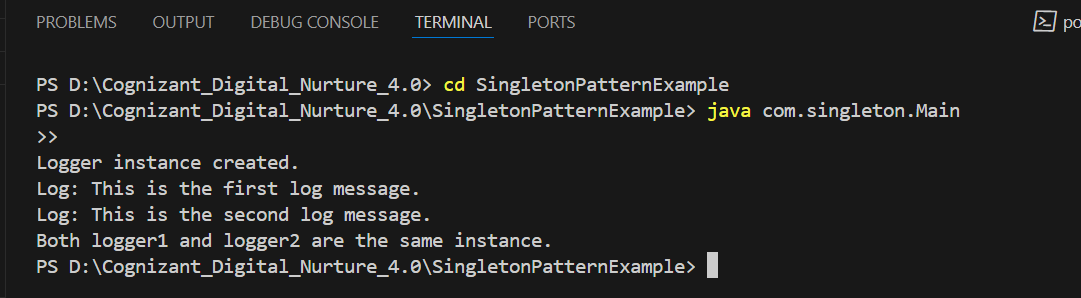
    public void log(String message) {

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

    }

}

Output:



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

**FactoryPatternDemo.java:**

public class FactoryPatternDemo {

    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();

    }

}

**Document.java:**

public interface Document {

    void open();

}

**DocumentFactory.java:**

public abstract class DocumentFactory {

    public abstract Document createDocument();

}

**ExcelDocument.java:**

public class ExcelDocument implements Document {

    public void open() {

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

    }

}

**ExcelDocumentFactory.java:**

public class ExcelDocumentFactory extends DocumentFactory {

    public Document createDocument() {

        return new ExcelDocument();

    }

}

**PdfDocument.java:**

public class PdfDocument implements Document {

    public void open() {

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

    }

}

**PdfDocumentFactory.java:**

public class PdfDocumentFactory extends DocumentFactory {

    public Document createDocument() {

        return new PdfDocument();

    }

}

**WordDocument.java:**

public class WordDocument implements Document {

    public void open() {

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

    }

}

**WordDocumentFactory.java:**

public class WordDocumentFactory extends DocumentFactory {

    public Document createDocument() {

        return new WordDocument();

    }

}

Output:

