**WEEK**-**1 Design Patterns and Principles**

Practice solutions

**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.

**Code:**

**Logger.java**

package week1.design\_patterns.singleton;

public class Logger {

    private static Logger instance;

    private Logger() {

        System.out.println("Logger started");}

    public static Logger getInstance() {

        if (instance == null)

           { instance = new Logger();}

        return instance;

    }

    public void log(String message) {

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

    }}

**Main.java**

package week1.design\_patterns.singleton;

public class Main {

    public static void main(String[] args) {

        Logger logger1 = Logger.getInstance();

        logger1.log("hi");

        Logger logger2 = Logger.getInstance();

        logger2.log("bye");

        if (logger1 == logger2) {

            System.out.println("Only one instance of Logger is used.");

        } else {

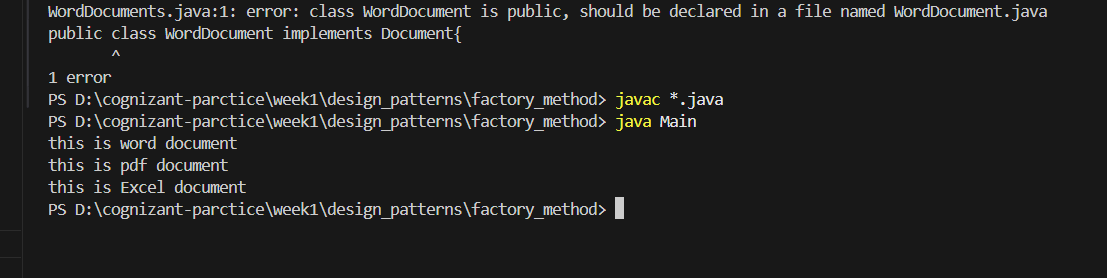
            System.out.println("Different Logger instances exist.");

        }

    }

}

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.

Code:

**Document.java**

public interface Document{

    void open();

}

**DocumentFactory.java**

public abstract class DocumentFactory{

   public abstract Document createDocument();

}

**PdfFactory.java**

public class PdfFactory extends DocumentFactory {

    public Document createDocument() {

        return new PdfDocument();

    }}

**PdfDocument.java**

public class PdfDocument implements Document{

    public void open()

    {

        System.out.println("this is pdf document");

    }}

**ExcelFactory.java**

public class ExcelFactory extends DocumentFactory {

    public Document createDocument() {

        return new ExcelDocument(); }}

**ExcelDocument.java**

public class ExcelDocument implements Document{

    public void open()

    {

        System.out.println("this is Excel document");

    }}

**WordDocument.java**

public class WordDocument implements Document{

    public void open()

    {

        System.out.println("this is word document");

    }}

**WordFactory.java**

public class WordFactory extends DocumentFactory {

    public Document createDocument() {

        return new WordDocument();

 }   }

**Main.java**

public class Main {

    public static void main(String[] args) {

        DocumentFactory wordFactory = new WordFactory();

        Document word = wordFactory.createDocument();

        word.open();

        DocumentFactory pdfFactory = new PdfFactory();

        Document pdf = pdfFactory.createDocument();

        pdf.open();

        DocumentFactory excelFactory = new ExcelFactory();

        Document excel = excelFactory.createDocument();

        excel.open(); } }

OUTPUT:

