**DN 4.0 JAVA FSE SOLUTIONS – WEEK 1**

**SKILL: Design principles & Patterns**

**Exercise 1:** Implementing the Singleton Pattern

**Code:**

**Logger.java -**

public class Logger {

    // Private static instance of Logger

    private static Logger instance = new Logger();

    // Constructor is private so that this class cannot be instantiated from outside

    private Logger() {

        System.out.println("Initialized Logger");

    }

    // Public method to return the single instance

    public static Logger getInstance() {

        return instance;

    }

    // Sample logging method

    public void log(String message) {

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

    }

}

**Main.java –**

public class Main {

    public static void main(String[] args) {

        // Getting Logger instances

        Logger log1 = Logger.getInstance();

        Logger log2 = Logger.getInstance();

        // Using logger

        log1.log("This is the message of log1");

        log2.log("This is the message of log2");

        // Verifying if both references are pointing to the same object

        if (log1 == log2) {

            System.out.println("Singleton works because both logger instances are same");

        } else {

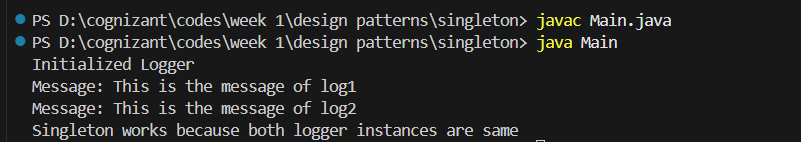
            System.out.println("Different logger instances exist(Singleton failed)");

        }

    }

}

**Output:**

****

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

**Code:**

**FactoryMethodPatternExample.java –**

// Interface for all types of documents

interface Document {

    void open();

}

// Concrete document classes

class WordDocument implements Document {

    public void open() {

        System.out.println("Word document is created");

    }

}

class PdfDocument implements Document {

    public void open() {

        System.out.println("PDF document is created");

    }

}

class ExcelDocument implements Document {

    public void open() {

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

    }

}

// Abstract factory

abstract class DocumentFactory {

    public abstract Document createDocument();

}

// Concrete factories

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();

    }

}

// Main class

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();

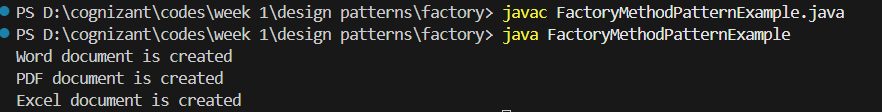
        Document excelDoc = excelFactory.createDocument();

        excelDoc.open();

    }

}

**Output:**

****