**Exercise 1: Implementing the Singleton Pattern**

Mandatory Hands-on

Sparshak Ghosh

**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.

**ANSWER:   
 Logger.java**public class Logger

{

    private static Logger instance;

    private Logger()

    {

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

    }

    public static Logger getInstance()

    {

        if (instance==null)

            instance = new Logger();

        return  instance;

    }

    public void logPrinter(String message)

    {

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

    }

}

**Test.java //test class**

public class Test

{

    public static void main(String args[])

{

        Logger l1 = Logger.getInstance();

        Logger l2 = Logger.getInstance();

        l1.logPrinter("This is the first log message.");

        l2.logPrinter("This is the second log message.");

        if (l1 == l2)

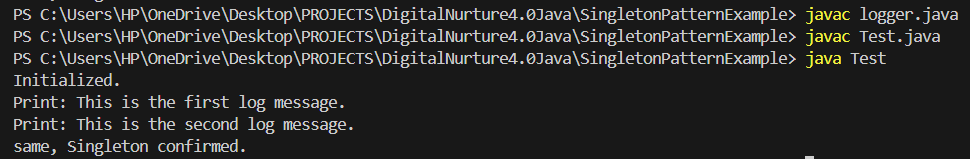
            System.out.println("same, Singleton confirmed.");

        else

            System.out.println("different, Singleton failed.");

    }

}

 **OUTPUT**

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

Sparshak Ghosh

**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.

**ANSWER:**

**Document.java //common interface**

public interface Document

{

    void open();

}

**WordDocument.java**

public class WordDocument implements Document

{

    public void open()

{

        System.out.println("OPENING WORD DOC.");

    }

}

**PDFDocument.java**

public class PDFDocument implements Document

{

    public void open()

{

        System.out.println("OPENING PDF DOC.");

    }

}

**ExcelDocument.java**

public class ExcelDocument implements Document

{

    public void open()

{

        System.out.println("OPENING EXCEL DOC.");

    }

}

**DocumentFactory.java**

public abstract class DocumentFactory

{

    public abstract Document createDocument();

}

**WordDocumentFactory.java**

Sparshak Ghosh

public class WordDocumentFactory extends DocumentFactory

{

    public Document createDocument()

{

        return new WordDocument();

    }

}

**PDFDocumentFactory.java**

public class PDFDocumentFactory extends DocumentFactory

{

    public Document createDocument()

{

        return new PDFDocument();

    }

}

**ExcelDocumentFactory.java**

public class ExcelDocumentFactory extends DocumentFactory

{

    public Document createDocument()

{

        return new ExcelDocument();

    }

}

**Test.java // test class**

public class Test

{

    public static void main(String args[])

    {

        DocumentFactory word = new WordDocumentFactory();

        Document wordDoc = word.createDocument();

        wordDoc.open();

        DocumentFactory pdf = new PDFDocumentFactory();

        Document pdfDoc = pdf.createDocument();

        pdfDoc.open();

        DocumentFactory excel = new ExcelDocumentFactory();

        Document excelDoc = excel.createDocument();

        excelDoc.open();

    }

}

