**Design Patterns and Principles**

**Week 1**

**Mandatory Hands-on**

**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

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);

    }

}

TestSingleton.java

public class TestSingleton{

    public static void main(String[] args){

        Logger logger1 = Logger.getInstance();

        Logger logger2 = Logger.getInstance();

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

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

        if(logger1 == logger2){

            System.out.println("Both loggers are the same instance.");

        }else{

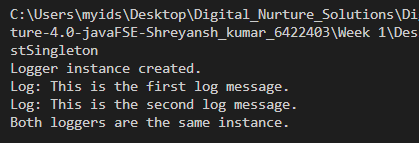
            System.out.println("Different instances found! Singleton pattern not working.");

        }

    }

}

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();

}

// WordDocument.java

public class WordDocument implements Document {

    @Override

    public void open() {

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

    }

}

// PdfDocument.java

public class PdfDocument implements Document {

    @Override

    public void open() {

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

    }

}

// ExcelDocument.java

public class ExcelDocument implements Document {

    @Override

    public void open() {

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

    }

}

// DocumentFactory.java

public abstract class DocumentFactory {

    public abstract Document createDocument();

}

// WordDocumentFactory.java

public class WordDocumentFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new WordDocument();

    }

}

// PdfDocumentFactory.java

public class PdfDocumentFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new PdfDocument();

    }

}

// ExcelDocumentFactory.java

public class ExcelDocumentFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new ExcelDocument();

    }

}

// FactoryPatternTest.java

public class FactoryPatternTest {

    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();

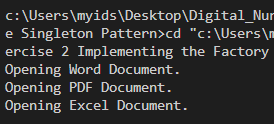
        DocumentFactory excelFactory = new ExcelDocumentFactory();

        Document excelDoc = excelFactory.createDocument();

        excelDoc.open();

    }

}

Output:

Submitted By:  
Name: Shreyansh Kumar  
Superset Id: 6422403