**Week-1 Design Patterns and Principles**

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:**

**main.java**

// main.java

public class main {

    public static void main(String[] args) {

        System.out.println("Hello, world!");

           // Try to get multiple instances

        Logger logger1 = Logger.getInstance();

        Logger logger2 = Logger.getInstance();

        // Logging some messages

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

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

        // Verifying that both references point to the same instance

        if (logger1 == logger2) {

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

        } else {

            System.out.println("Different instances detected!");

        }

    }

}

**Logger.java**

public class Logger {

    // Private static instance of Logger (eager initialization)

    private static Logger instance = new Logger();

    // Private constructor to prevent instantiation

    private Logger() {

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

    }

    // Public method to provide access to the instance

    public static Logger getInstance() {

        return instance;

    }

    // Example logging method

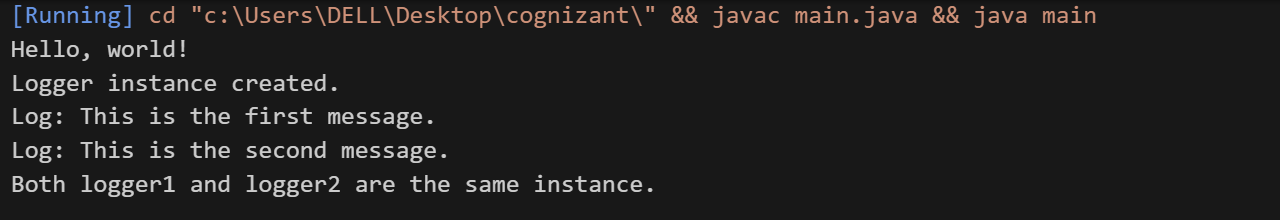
    public void log(String message) {

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

    }

}

**Output:**



1. **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 {

public void open() {

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

}

}

**PdfDocument.java**

public class PdfDocument implements Document {

public void open() {

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

}

}

**ExcelDocument.java**

public class ExcelDocument implements Document {

public void open() {

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

}

}

**DocumentFactory.java**

public abstract class DocumentFactory {

public abstract Document createDocument();

}

**WordDocumentFactory.java**

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();

}

}

**Main.java**

public class Main {

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();

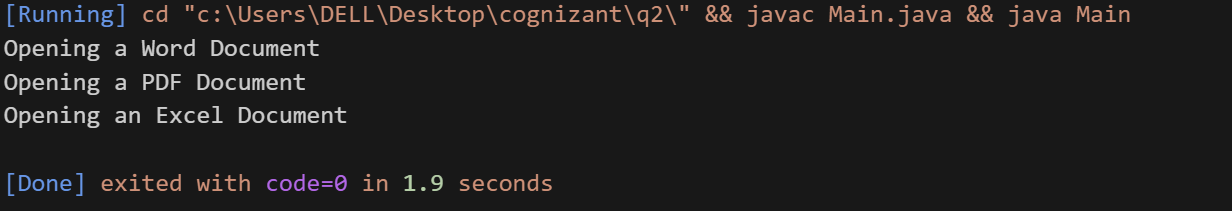
Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

**Output:**

****