# Week1\_DesignPrinciplesAndPatterns\_HandsOn

## 📌 Exercise 1: Implementing the Singleton Pattern

🧾 Problem Statement:  
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:

public class Logger {  
 private static Logger instance;  
  
 private Logger() {  
 System.out.println("Logger created");  
 }  
  
 public static Logger getInstance() {  
 if (instance == null) {  
 instance = new Logger();  
 }  
 return instance;  
 }  
  
 public void log(String msg) {  
 System.out.println("LOG: " + msg);  
 }  
}  
  
public class Main {  
 public static void main(String[] args) {  
 Logger l1 = Logger.getInstance();  
 l1.log("First log");  
  
 Logger l2 = Logger.getInstance();  
 l2.log("Second log");  
  
 if (l1 == l2) {  
 System.out.println("Same instance");  
 }  
 }  
}

📸 Output Screenshot: (Insert manually on upload)  
🧠 Explanation:  
- Ensures only one logger instance is used.  
- getInstance() creates only if instance is null.  
- Follows the Singleton design pattern.

## 📌 Exercise 2: Implementing the Factory Method Pattern

🧾 Problem Statement:  
You are developing a document management system that needs to create different types of documents (Word, PDF, Excel) using the Factory Method Pattern.

💡 Code:

public interface Document {  
 void open();  
}  
  
public class WordDocument implements Document {  
 public void open() {  
 System.out.println("Opening Word Document.");  
 }  
}  
  
public class PdfDocument implements Document {  
 public void open() {  
 System.out.println("Opening PDF Document.");  
 }  
}  
  
public class ExcelDocument implements Document {  
 public void open() {  
 System.out.println("Opening Excel Document.");  
 }  
}  
  
public abstract class DocumentFactory {  
 public abstract Document createDocument();  
}  
  
public class WordDocumentFactory extends DocumentFactory {  
 public Document createDocument() {  
 return new WordDocument();  
 }  
}  
  
public class PdfDocumentFactory extends DocumentFactory {  
 public Document createDocument() {  
 return new PdfDocument();  
 }  
}  
  
public class ExcelDocumentFactory extends DocumentFactory {  
 public Document createDocument() {  
 return new ExcelDocument();  
 }  
}  
  
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 Screenshot: (Insert manually on upload)  
🧠 Explanation:  
- Demonstrates polymorphism and encapsulation.  
- Each factory class creates one type of document.  
- Easy to extend for new document types.