**Deep Skilling - Week 1 Mandatory Hands-on Exercises**

**Skill : Design Patterns and Principles**

**Exercise - 1: Implementing the Singleton Pattern**

**Code:**

**Logger.java Class :**

package LoggerExample;

public class Logger {

private static Logger *instance*;

private Logger() {

System.***out***.println("Logger Initialised in constructor");

}

public static Logger getInstance() {

if(*instance* == null) {

*instance* = new Logger();

}

return *instance*;

}

public void logMessage(String msg) {

System.***out***.println("[LOG] : " + msg);

}

}

**LoggerTest.java Class :**

package LoggerExample;

public class LoggerTest {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

Logger log1 = Logger.*getInstance*();

Logger log2 = Logger.*getInstance*();

log1.logMessage("First Message");

log2.logMessage("Second Message");

if(log1 == log2) {

System.***out***.println("Both logger instances are the same. Hence, singleton works!!");

} else {

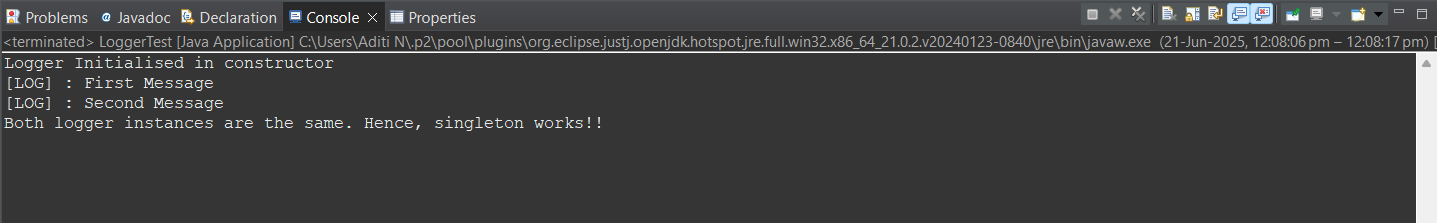
System.***out***.println("Logger instances are different. Hence, singleton failed!!");

}

}

}

**Output:**

****

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

**Code:**

**Document.java**

**package FactoryMethodPatternExample;**

**public interface Document {**

**void open();**

**}**

**WordDocument.java**

**package FactoryMethodPatternExample;**

**public class WordDocument implements Document{**

***@Override***

**public void open() {**

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

**}**

**}**

**ExcelDocument.java**

**package FactoryMethodPatternExample;**

**public class ExcelDocument implements Document{**

***@Override***

**public void open() {**

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

**}**

**}**

**PdfDocument.java**

**package FactoryMethodPatternExample;**

**public class PdfDocument implements Document{**

***@Override***

**public void open() {**

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

**}**

**}**

**DocumentFactory.java**

**package FactoryMethodPatternExample;**

**public abstract class DocumentFactory {**

**public abstract Document createDocument();**

**}**

**WordDocFactory.java**

**package FactoryMethodPatternExample;**

**public class WordDocFactory extends DocumentFactory{**

***@Override***

**public Document createDocument() {**

**return new WordDocument();**

**}**

**}**

**PdfDocFactory.java**

**package FactoryMethodPatternExample;**

**public class PdfDocFactory extends DocumentFactory{**

***@Override***

**public Document createDocument() {**

**return new PdfDocument();**

**}**

**}**

**ExcelDocFactory.java**

**package FactoryMethodPatternExample;**

**public class ExcelDocFactory extends DocumentFactory{**

***@Override***

**public Document createDocument() {**

**return new ExcelDocument();**

**}**

**}**

**Test.java**

**package FactoryMethodPatternExample;**

**public class Test {**

**public static void main(String[] args) {**

**DocumentFactory wordFactory = new WordDocFactory();**

**Document wordDoc = wordFactory.createDocument();**

**wordDoc.open();**

**DocumentFactory pdfFactory = new PdfDocFactory();**

**Document pdfDoc = pdfFactory.createDocument();**

**pdfDoc.open();**

**DocumentFactory excelFactory = new ExcelDocFactory();**

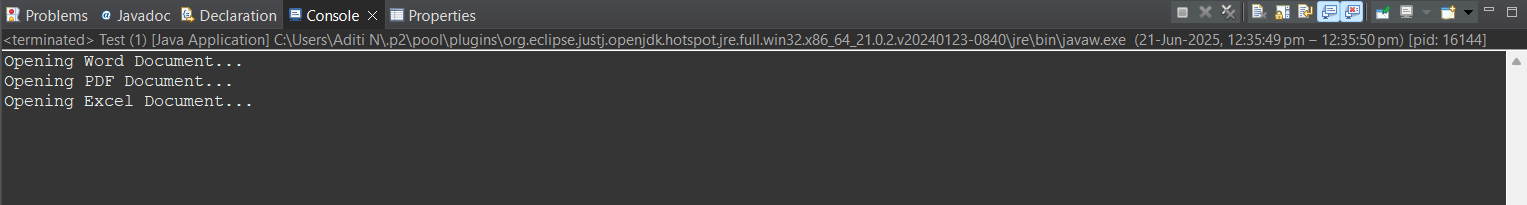
**Document excelDoc = excelFactory.createDocument();**

**excelDoc.open();**

**}**

**}**

**Output:**

****