**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:**  
**Main.java**

package com.singleton.example;

public class Main {

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 logger instances are the same (Singleton works).");

} **else** {

System.*out*.println("Logger instances are different (Singleton failed).");

}

}

}

**Logger.java**  
package com.singleton.example;

public class Logger {

private static Logger *instance*;

private Logger() {

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

}

public static Logger getInstance() {

if (*instance* == null) {

*instance* = new Logger();

}

return *instance*;

}

public void log(String message) {

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

}

}

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

**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**package documents;

public interface Document {

void open();

}

**WordDocument.java**

package documents;

public class WordDocument implements Document {

@Override

public void open() {

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

}

}  
  
**PdfDocument.java**  
package documents;

public class PdfDocument implements Document {

@Override

public void open() {

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

}

}  
  
**ExcelDocument.java**  
package documents;

public class ExcelDocument implements Document {

@Override

public void open() {

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

}

}  
  
**DocumentFactory.java**  
package factory;

import documents.Document;

public abstract class DocumentFactory {

public abstract Document createDocument();

}  
  
**WordDocumentFactory.java**  
package factory;

import documents.Document;

import documents.WordDocument;

public class WordDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new WordDocument();

}

}  
  
**PdfDocumentFactory.java**  
package factory;

import documents.Document;

import documents.PdfDocument;

public class PdfDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new PdfDocument();

}

}  
  
**ExcelDocumentFactory.java**  
  
package factory;

import documents.Document;

import documents.ExcelDocument;

public class ExcelDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new ExcelDocument();

}

}  
  
**Main.java**  
package main;

import factory.\*;

import documents.Document;

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:**

A screenshot of a computer

AI-generated content may be incorrect.