**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 for implementing the Singleton Pattern for the given scenario:  
  
Logger.java:  
public** **class** Logger {

**private** **static** Logger *logger*;

**private** Logger () {

}

**public** **static** Logger getLogger() {

**return** *logger*;

}

}

**LoggerTest.java:**

**public** **class** LoggerTest {

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

Logger logger1 = Logger.*getLogger*();

Logger logger2 = Logger.*getLogger*();

**if** (logger1 == logger2)

System.***out***.println("Instances logger1 and logger2 are pointing to the same object.");

**else** {

System.***out***.println("Instances logger1 and logger2 are not pointing to the same object.");

}

}

}

The logger class represents the singleton design pattern. First, the declaration of the object reference is executed. Later, If not instantiated, the object is instantiated only once.

**Output Screenshot for Singleton Pattern:**

**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 for implementing the Factory Method Pattern for the given scenario:**

**Document.java (Interface):  
public** **interface** Document {

**void** open();

}

**WordDocument.java:**

**public** **class** WordDocument **implements** Document {

@Override

**public** **void** open() {

System.***out***.println("Opens a word document");

}

}

**ExcelDocument.java:**

**public** **class** ExcelDocument **implements** Document {

@Override

**public** **void** open() {

System.***out***.println("Opens an excel document");

}

}

**PdfDocument.java:**

**public** **class** PdfDocument **implements** Document{

@Override

**public** **void** open() {

System.***out***.println("Opens a pdf 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();

}

}

**ExcelDocumentFactory.java:**

**public** **class** ExcelDocumentFactory **extends** DocumentFactory {

@Override

**public** Document createDocument() {

**return** **new** ExcelDocument();

}

}

**PdfDocumentFactory.java:**

**public** **class** PdfDocumentFactory **extends** DocumentFactory {

@Override

**public** Document createDocument() {

**return** **new** PdfDocument();

}

}

**FactoryTest.java:  
public** **class** FactoryTest {

**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:**

**A screenshot of a computer

AI-generated content may be incorrect.**