**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.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **SingletonPatternExample**.
2. **Define a Singleton Class:**
   * Create a class named Logger that has a private static instance of itself.
   * Ensure the constructor of Logger is private.
   * Provide a public static method to get the instance of the Logger class.
3. **Implement the Singleton Pattern:**
   * Write code to ensure that the Logger class follows the Singleton design pattern.
4. **Test the Singleton Implementation:**
   * Create a test class to verify that only one instance of Logger is created and used across the application.

**Code:**

package singletonpattern;

//Logger class

public class Logger {

//Logger constructor and instance of logger class

private static Logger *instance*;

private Logger ()

{

System.*out*.println("Logger Instance Created: ");

}

public static Logger getInstance() {

if (*instance* == null) {

*instance*=new Logger ();

}

return *instance*;

}

public void log (String message) {

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

}

}

package singletonpattern;

// to test the logger class with main

public class Loggertest {

public static void main (String [] args) {

Logger logger1 = Logger.*getInstance*();

logger1.log ("This is the 1st log message.");

Logger logger2 = Logger.*getInstance*();

logger2.log ("This is the 2nd log message.");

if (logger1 == logger2)

{

System.out.println("Both logger instances are the same (Singleton confirmed).");

} else {

System.out.println("Different instances (Singleton failed).");

}

}

}

**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.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **FactoryMethodPatternExample**.
2. **Define Document Classes:**
   * Create interfaces or abstract classes for different document types such as **WordDocument**, **PdfDocument**, and **ExcelDocument**.
3. **Create Concrete Document Classes:**
   * Implement concrete classes for each document type that implements or extends the above interfaces or abstract classes.
4. **Implement the Factory Method:**
   * Create an abstract class **DocumentFactory** with a method **createDocument()**.
   * Create concrete factory classes for each document type that extends DocumentFactory and implements the **createDocument()** method.
5. **Test the Factory Method Implementation:**
   * Create a test class to demonstrate the creation of different document types using the factory method.

**Code:**

package factorymethodpattern;

//class interface document

public interface Document {

void open ();

}

// these are 3 class that interface class for different type of documents

public class WordDocument implements Document {

@Override

public void open () {

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

}

}

public class PdfDocument implements Document {

@Override

public void open () {

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

}

}

public class ExcelDocument implements Document {

@Override

public void open () {

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

}

}

//abstract class

public abstract class DocumentFactory {

public abstract Document createDocument();

}

//class extends doucumentfactory

public class WordDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new WordDocument();

}

}

public class PdfDocumentFactory extends DocumentFactory{

@Override

public Document createDocument() {

return new PdfDocument();

}

}

public class ExcelDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new ExcelDocument();

}

}

// to test the factorypattern

public class Docu {

public static void main (String [] args) {

DocumentFactory wordfac = new WordDocumentFactory();

Document worddoc = wordfac.createDocument();

worddoc.open();

DocumentFactory pdffac = new PdfDocumentFactory();

Document pdfdoc = pdffac.createDocument();

pdfdoc.open();

DocumentFactory xcelfac = new ExcelDocumentFactory();

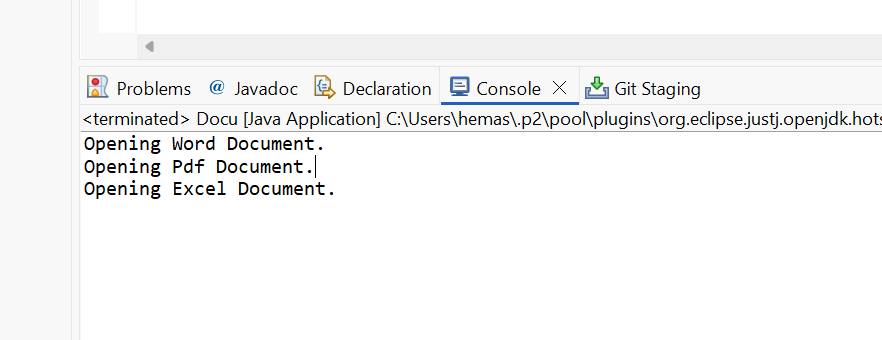
Document xceldoc = xcelfac.createDocument();

xceldoc.open();

}

}

**Output:**

****