**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.

**ANSWER**

**Code :-**

***File : SingletonPatternExample/Logger.java***

public class Logger

{

    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);

    }

}

***File : SingletonPatternExample.Main.java***

public class Main

{

    public static void main(String[] args)

    {

        Logger logger1 = Logger.getInstance();

        logger1.log("This is the first log message");

        Logger logger2 = Logger.getInstance();

        logger2.log("This is the second log message");

        if (logger1 == logger2)

        {

            System.out.println("logger1 and logger2 are the same instance");

        } else

        {

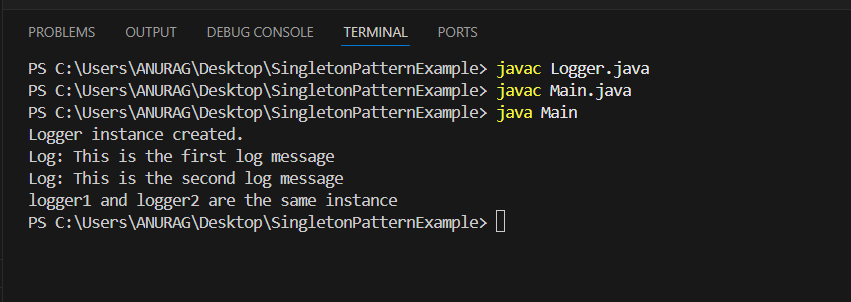
            System.out.println("Different logger instances found");

        }

    }

}

**Output:-**

****

**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.**

**ANSWER**

**Code :-**

***File : FactoryMethodPatternExample/Document.java***

public interface Document

{

    void open();

}

***File : FactoryMethodPatternExample/WordDocument.java***

public class WordDocument implements Document

{

    @Override

    public void open()

    {

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

    }

}

***File : FactoryMethodPatternExample/PdfDocument.java***

public class PdfDocument implements Document

{

    @Override

    public void open()

    {

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

    }

}

***File : FactoryMethodPatternExample/ExcelDocument.java***

public class ExcelDocument implements Document

{

    @Override

    public void open()

    {

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

    }

}

***File : FactoryMethodPatternExample/DocumentFactory.java***

public abstract class DocumentFactory

{

    public abstract Document createDocument();

}

***File : FactoryMethodPatternExample/WordDocumentFactory.java***

public class WordDocumentFactory extends DocumentFactory

{

    @Override

    public Document createDocument()

    {

        return new WordDocument();

    }

}

***File : FactoryMethodPatternExample/PdfDocumentFactory.java***

public class PdfDocumentFactory extends DocumentFactory

{

    @Override

    public Document createDocument()

    {

        return new PdfDocument();

    }

}

***File : FactoryMethodPatternExample/ExcelDocumentFactory.java***

public class ExcelDocumentFactory extends DocumentFactory

{

    @Override

    public Document createDocument()

    {

        return new ExcelDocument();

    }

}

***File : FactoryMethodPatternExample/FactoryMethodTest.java***

public class FactoryMethodTest

{

    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();

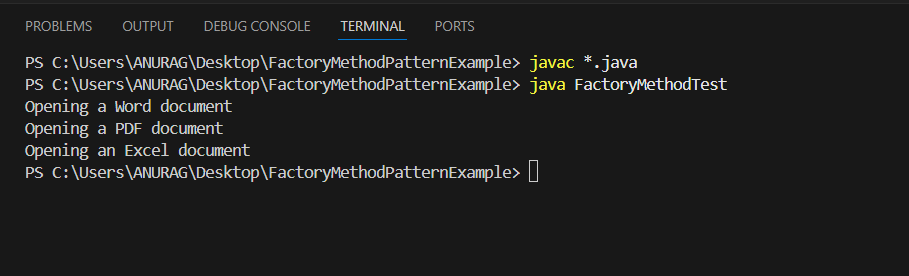
        Document excelDoc = excelFactory.createDocument();

        excelDoc.open();

    }

}

**Output :-**

****