**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.

**Solution:**

* **Code(s):**

*// Created a new Java project named* ***SingletonPatternExample****.*

***// Logger.java***

package com.singletonpattern;

public class Logger {

    private static Logger instance;

*//Ensuring the constructor of Logger is private*

    private Logger() {}

*//Creating a public static method to get the instance of the Logger class*

    public static Logger getInstance() {

        if (instance == null) {

            instance = new Logger();

        }

        return instance;

    }

    public void log(String message) {

        System.out.println(message);

    }

}

***// Test.java***

package com.singletonpattern;

public class Test {

    public static void main(String[] args) {

        Logger logger1 = Logger.getInstance();

        Logger logger2 = Logger.getInstance();

        logger1.log("This is a log message from logger1.");

        logger2.log("This is a log message from logger2.");

*// Check if both references point to the same instance*

        if (logger1 == logger2) {

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

        }

        else {

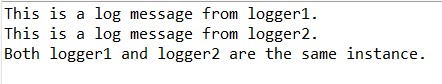
            System.out.println("logger1 and logger2 are different instances.");

        }

    }

}

* **Output(s):**



**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.

**Solution:**

* **Code(s):**

1. **Create a New Java Project:**

*// Created a new Java project named* ***FactoryMethodPatternExample.***

1. **Define Document Classes:**

***// Document.java***

package com.factorymethod;

public interface Document {

    public void openDocument();

}

***// WordDocument.java***

package com.factorymethod;

public abstract class WordDocument implements Document {

    public abstract void editDocument();

}

***// PdfDocument.java***

package com.factorymethod;

public abstract class PdfDocument implements Document {

    public abstract void viewDocument();

}

***// ExcelDocument.java***

package com.factorymethod;

public abstract class ExcelDocument implements Document {

    public abstract void calculateDocument();

}

1. **Create Concrete Document Classes:**

***// Word.java***

package com.factorymethod;

public class Word extends WordDocument {

    @Override

    public void openDocument() {

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

    }

    @Override

    public void editDocument() {

        System.out.println("Editing the Word Document.");

    }

}

***// Pdf.java***

package com.factorymethod;

public class Pdf extends PdfDocument {

    @Override

    public void openDocument() {

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

    }

    @Override

    public void viewDocument() {

        System.out.println("Viewing the PdfDocument.");

    }

}

***// Excel.java***

package com.factorymethod;

public class Excel extends ExcelDocument {

    @Override

    public void openDocument() {

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

    }

    @Override

    public void calculateDocument() {

        System.out.println("Performing Calculation in the Excel Document.");

    }

}

1. **Implement the Factory Method:**

***// DocumentFactory.java***

package com.factorymethod;

public abstract class DocumentFactory {

    public abstract Document createDocument();

}

***// WordDocFactory.java***

package com.factorymethod;

public class WordDocFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new Word();

    }

}

***// PdfDocFactory.java***

package com.factorymethod;

public class PdfDocFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new Pdf();

    }

}

***// ExcelDocFactory.java***

package com.factorymethod;

public class ExcelDocFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new Excel();

    }

}

1. **Test the Factory Method Implementation:**

*// Test.java*

package com.factorymethod;

public class Test {

    public static void main (String args[]) {

        DocumentFactory wordFactory = new WordDocFactory();

        Document wordDoc = wordFactory.createDocument();

        wordDoc.openDocument();

        ((WordDocument)wordDoc).editDocument();

        DocumentFactory excelFactory = new ExcelDocFactory();

        Document excelDoc = excelFactory.createDocument();

        excelDoc.openDocument();

        ((ExcelDocument)excelDoc).calculateDocument();

        DocumentFactory pdfFactory = new PdfDocFactory();

        Document pdfDoc = pdfFactory.createDocument();

        pdfDoc.openDocument();

        ((Pdf)pdfDoc).viewDocument();

    }

}

* **Output(s):**

