**Design Patterns Questions**

**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**

using System;

class Logger

{

    public static Logger instance;

    private Logger()

    {

        Console.WriteLine($"Instance is Created");

    }

    public static Logger getInstance()

    {

        if (instance == null)

        {

            instance = new Logger();

            return instance;

        }

        return instance;

    }

     public override string ToString()

    {

        return $"Logger Singleton Instance (HashCode: {this.GetHashCode()})";

    }

}

class TestLogger

{

    Logger instance = Logger.getInstance();

    public void printInstance()

    {

        Console.WriteLine($"Logger Instance: {instance}");

    }

    public Logger getInstance() => instance;

}

public class Singleton

{

    public static void Main(string[] args)

    {

        TestLogger instance1 = new TestLogger();

        instance1.printInstance();

        TestLogger instance2 = new TestLogger();

        instance2.printInstance();

        if (instance1.getInstance() == instance2.getInstance())

        {

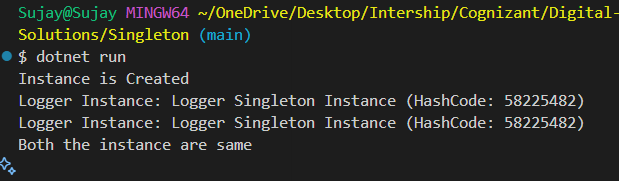
            Console.WriteLine($"Both the instance are same");

        }

    }

}

**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.

**SOLUTION**

using System;

interface Document

{

    void open();//c sharp doesn't support any access modifiers inside an interface

}

class WordDocument : Document

{

    public void open()

    {

        Console.WriteLine("Opening Word Document");

    }

}

class PdfDocument : Document

{

    public void open()

    {

        Console.WriteLine("Opening Pdf Document");

    }

}

class ExcelDocument: Document

{

    public void open()// no override keyword is used

    {

        Console.WriteLine("Opening Excel Document");

    }

}

abstract class DocumentFactory

{

    public abstract Document createDocument();

}

class PdfFactory : DocumentFactory

{

    public override Document createDocument() => new PdfDocument();

}

class WordFactory : DocumentFactory

{

    public override Document createDocument() => new WordDocument();

}

class ExcelFactory : DocumentFactory

{

    public override Document createDocument() => new ExcelDocument();

}

public class Program

{

    public static void Main(string[] args)

    {

        DocumentFactory wordDocFactory = new WordFactory();

        Document wordDoc = wordDocFactory.createDocument();

        wordDoc.open();

        DocumentFactory PdfDocFactory = new PdfFactory();

        Document PdfDoc = PdfDocFactory.createDocument();

        PdfDoc.open();

        DocumentFactory ExcelDocFactory = new ExcelFactory();

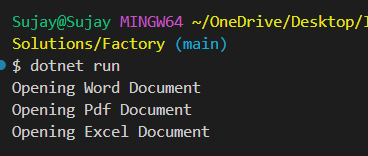
        Document ExcelDoc = ExcelDocFactory.createDocument();

        ExcelDoc.open();

    }

}

**OUTPUT**

****