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.

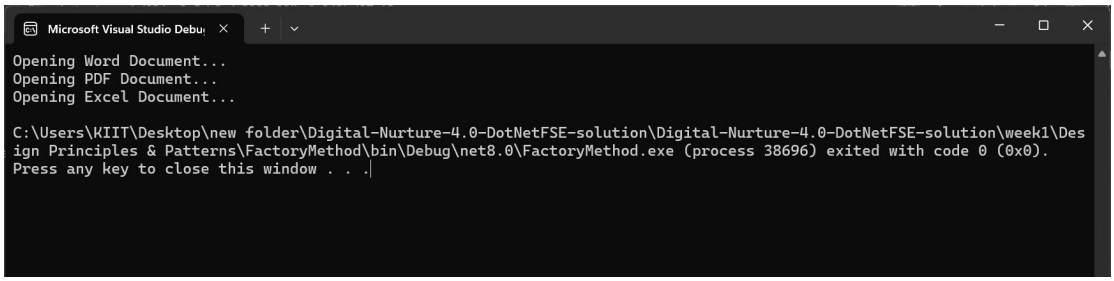
Steps:

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

ANSWER

```
namespace SingletonPattern
{

    public class Logger
    {
        private static Logger _instance;
        private static readonly object _lock = new object();

        // Private constructor prevents external instantiation
        private Logger()
        {
            Console.WriteLine("Logger Initialized");
        }

        public static Logger GetInstance()
        {
            // Thread-safe lazy initialization
            if (_instance == null)
            {
                lock (_lock)
                {
                    if (_instance == null)
                        _instance = new Logger();
                }
            }
            return _instance;
        }

        public void Log(string message)
        {
            Console.WriteLine($"[LOG]: {message}");
        }
    }

    class SingletonTest
    {
```

```
        static void Main(string[] args)
        {
            Logger logger1 = Logger.GetInstance();
            Logger logger2 = Logger.GetInstance();

            logger1.Log("This is the first log.");
            logger2.Log("This is the second log.");

            Console.WriteLine($"Are both loggers same instance?
{object.ReferenceEquals(logger1, logger2)}");
        }
    }
}
```



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:

- o Create a new Java project named FactoryMethodPatternExample.

2. Define Document Classes:

- o Create interfaces or abstract classes for different document types such as WordDocument, PdfDocument, and ExcelDocument.

3. Create Concrete Document Classes:

- o Implement concrete classes for each document type that implements or extends the above interfaces or abstract classes.

4. Implement the Factory Method:

- o Create an abstract class DocumentFactory with a method createDocument().
- o Create concrete factory classes for each document type that extends DocumentFactory and implements the createDocument() method.

5. Test the Factory Method Implementation:

- o Create a test class to demonstrate the creation of different document types using the factory method.

ANSWER

```
namespace FactoryMethod
{
    // Abstract Product
    public interface IDocument
    {
        void Open();
    }

    // Concrete Products
    public class WordDocument : IDocument
    {
        public void Open() => Console.WriteLine("Opening Word
Document...");
    }

    public class PdfDocument : IDocument
    {
        public void Open() => Console.WriteLine("Opening PDF Document...");
    }

    public class ExcelDocument : IDocument
    {
        public void Open() => Console.WriteLine("Opening Excel
Document...");
    }

    // Abstract Creator
    public abstract class DocumentFactory
    {
        public abstract IDocument CreateDocument();
    }

    // Concrete Creators
    public class WordDocumentFactory : DocumentFactory
    {
        public override IDocument CreateDocument() => new WordDocument();
    }

    public class PdfDocumentFactory : DocumentFactory
    {
        public override IDocument CreateDocument() => new PdfDocument();
    }

    public class ExcelDocumentFactory : DocumentFactory
    {
        public override IDocument CreateDocument() => new ExcelDocument();
    }

    // Test Class
    class FactoryMethodTest
    {
        static void Main(string[] args)
        {
            DocumentFactory factory;

            factory = new WordDocumentFactory();
            IDocument wordDoc = factory.CreateDocument();
            wordDoc.Open();

            factory = new PdfDocumentFactory();
```

```
            IDocument pdfDoc = factory.CreateDocument();
            pdfDoc.Open();

            factory = new ExcelDocumentFactory();
            IDocument excelDoc = factory.CreateDocument();
            excelDoc.Open();
        }
    }
}
```