|  |  |
| --- | --- |
| **Cognizant Digital Nurture 4.0: Deep Skilling Hands-on** | |
| Name: Rajatesh Paul | Superset ID: 6365356 |
| Week: 01 | **Design Patterns and Principles** |

**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 (C#) Project:

* Create a new Java (C#) project named SingletonPatternExample.

1. 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.

1. Implement the Singleton Pattern:

* Write code to ensure that the Logger class follows the Singleton design pattern.

1. Test the Singleton Implementation:

* Create a test class to verify that only one instance of Logger is created and used across the application.

#Code-

**Logger.cs**-

namespace Singleton{

public class Logger{

private static Logger instance;

private Logger(){

}

public static Logger GetInstance(){

if (instance == null){

instance = new Logger();

}

return instance;

}

}

}

**SingletonTest.cs-**

using System;

namespace Singleton

{

public class SingletonTest

{

public static void Main(string[] args)

{

Logger L1 = Logger.GetInstance();

Logger L2 = Logger.GetInstance();

if (L1 == L2)

{

Console.WriteLine("Both Logger1 and Logger2 are the same instance!");

}

else

{

Console.WriteLine("Both are Different");

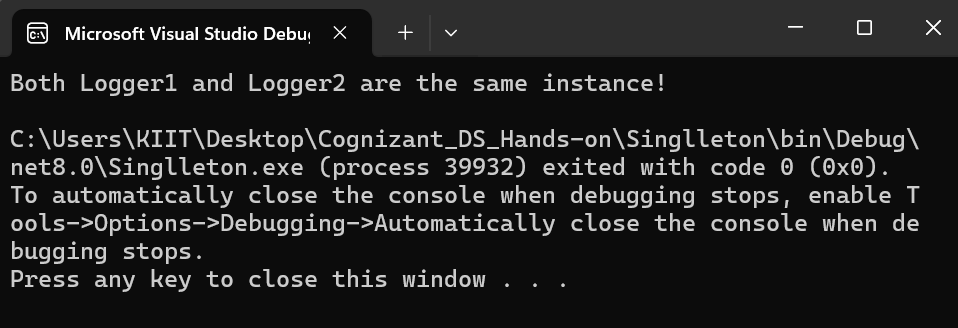
}

}

}

}

**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.

1. Define Document Classes:

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

1. Create Concrete Document Classes:

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

1. 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.

1. Test the Factory Method Implementation:

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

#Code-

**1. Document.cs**

public interface Document

{

void Create();

}

**2. DocumentFactory.cs**

public abstract class DocumentFactory

{

public abstract Document CreateDocument();

}

public class ExcelDocument : Document

{

public void Create() => Console.WriteLine("Created Excel document");

}

**3. WordDocument.cs**

public class WordDocument : Document

{

public void Create() => Console.WriteLine("Created Word document");

}

**4. WordDocumentFactory.cs**

public class WordDocumentFactory : DocumentFactory

{

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

}

**5. ExcelDocument.cs**

public class ExcelDocument : Document

{

public void Create() => Console.WriteLine("Created Excel document");

}

**6. ExcelDocumentFactory.cs**

public class ExcelDocumentFactory : DocumentFactory

{

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

}

**7. PdfDocument.cs**

public class PdfDocument : Document

{

public void Create() => Console.WriteLine("Created PDF document");

}

**8. PdfDocumentFactory.cs**

public class PdfDocumentFactory : DocumentFactory

{

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

}

**9. Program.cs**

class Program

{

static void Main()

{

DocumentFactory wordFactory = new WordDocumentFactory();

Document wordDoc = wordFactory.CreateDocument();

wordDoc.Create();

DocumentFactory pdfFactory = new PdfDocumentFactory();

Document pdfDoc = pdfFactory.CreateDocument();

pdfDoc.Create();

DocumentFactory excelFactory = new ExcelDocumentFactory();

Document excelDoc = excelFactory.CreateDocument();

excelDoc.Create();

}

}

**Output-**

