**Week 1 - Design Patterns and Principles - Hands-on**

DEBANJAN KAURI(Superset ID-6362388)

**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.

To achieve this:

1. Create a java project named *SingletonPatternExample*.

2. Define a Logger(Singleton) Class:

* It should contain a private static instance of itself.
* The constructor must be private to prevent direct instantiation.
* Provide a public static method that returns the single instance.

3. Implement the Singleton Pattern inside Logger class.

4. Create a test class to confirm getInstance() always returns the same Logger instance.

### **Code:**

using System;

namespace SingletonPatternExample

{

public sealed class Logger

{

private static Logger? \_instance;

private static readonly object \_lock = new();

private Logger()

{

Console.WriteLine("Logger instance created.");

}

public static Logger GetInstance()

{

if (\_instance == null)

{

lock (\_lock)

{

if (\_instance == null)

\_instance = new Logger();

}

}

return \_instance;

}

public void Log(string message)

{

Console.WriteLine($"[LOG]: {message}");

}

}

public class Program

{

public static void Main(string[] args)

{

Console.WriteLine("---Singleton Pattern Example---");

Logger logger1 = Logger.GetInstance();

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

Logger logger2 = Logger.GetInstance();

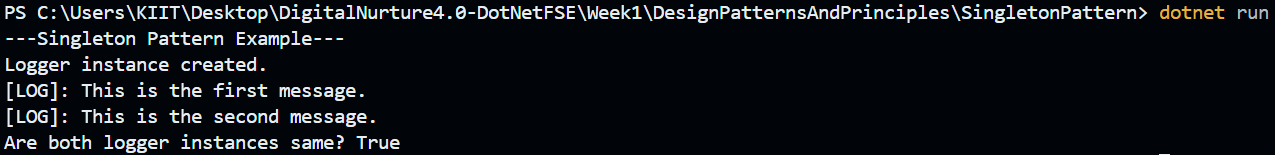
logger2.Log("This is the second message.");

Console.WriteLine($"Are both logger instances same? {ReferenceEquals(logger1, logger2)}");

}

}

}

**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) using the Factory Method Pattern.

To achieve this:

1. Create a Java project *FactoryMethodPatternExample.*

2. Define interfaces or abstract classes like WordDocument, PdfDocument, ExcelDocument.

3. Create concrete classes for each document type.

4. Implement the Factory Method:

* Create an abstract DocumentFactory class with a createDocument() method.
* Create a concrete factory for each document type that overrides createDocument().

5. Add a test class to demonstrate document creation using the factory method.

### **Code:**

using System;

namespace FactoryMethodPattern

{

public interface IDocument

{

void Open();

string GetTypeDescription();

}

public class WordDocument : IDocument

{

public void Open() => Console.WriteLine("Opening Word document...");

public string GetTypeDescription() => "Microsoft Word Document";

}

public class PdfDocument : IDocument

{

public void Open() => Console.WriteLine("Opening PDF document...");

public string GetTypeDescription() => "Adobe PDF Document";

}

public class ExcelDocument : IDocument

{

public void Open() => Console.WriteLine("Opening Excel spreadsheet...");

public string GetTypeDescription() => "Microsoft Excel Spreadsheet";

}

public abstract class DocumentFactory

{

public abstract IDocument CreateDocument();

public void ProcessDocument()

{

IDocument doc = CreateDocument();

Console.WriteLine($"\nDocument Type: {doc.GetTypeDescription()}");

doc.Open();

Console.WriteLine("Document processed.\n");

}

}

public class WordFactory : DocumentFactory

{

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

}

public class PdfFactory : DocumentFactory

{

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

}

public class ExcelFactory : DocumentFactory

{

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

}

public class Program

{

public static void Main()

{

Console.WriteLine("--- Factory Method Pattern – Document Management System ---\n");

Console.Write("Enter the type of document to create (word/pdf/excel): ");

string? input = Console.ReadLine()?.Trim().ToLower();

DocumentFactory? factory = input switch

{

"word" => new WordFactory(),

"pdf" => new PdfFactory(),

"excel" => new ExcelFactory(),

\_ => null

};

if (factory != null)

{

factory.ProcessDocument();

}

else

{

Console.WriteLine("Invalid input. Please enter 'word', 'pdf', or 'excel'.");

}

}

}

}

**OUTPUT:**