Nama : Prajna Paramitha Wardhany
Kelas : SE 07 01
NIM : 2311104016

# TP MODUL 14

## 1. MEMBUAT PROJECT MODUL ☑

Buka IDE misalnya dengan Visual Studio

A. Copy salah satu folder tugas pendahuluan yang dimiliki sebelumnya (dari modul 2 sampai modul 10), kemudian rename folder hasil copy-paste tersebut dengan tpmodul14_NIM (coba pilih tugas pendahuluan yang paling sederhana)

B. Misalnya menggunakan Visual Studio, bukalah project/folder yang di-copy sebelumnya.

## 2. REFACTORING DENGAN STANDAR CODE

Dengan mengikuti standard code yang digunakan (misal C# dengan standar dari .NET), pastikan kode yang dikumpulkan memenuhi faktor-faktor berikut:

A. Naming convention

i. Variable / Property / Attribute

ii. Method / Function / Procedure

B. White space dan indentation

C. Variable / attribute declarations

D. Comments

Kodingan dalam modul 13 diubah agar menerapkan clean code

Program.cs

```
using System;

public class Program
{
    public static void Main()
    {
        // ✅ Clean Code: Penamaan variabel menggunakan camelCase untuk variabel lokal
        // 🔧 Diperbaiki: Pemanggilan GetDataSingleton() → GetInstance()
        PusatDataSingleton data1 = PusatDataSingleton.GetInstance();
        PusatDataSingleton data2 = PusatDataSingleton.GetInstance();

        // ✅ Clean Code: Method AddData lebih ringkas dan sesuai konvensi
        // 🔧 Sebelumnya bernama AddSebuahData (kurang singkat dan eksplisit)
        data1.AddData("Nama 1: Nana");
        data1.AddData("Nama 2: Andi");
        data1.AddData("Asisten Praktikum: Budi");

        // ✅ Clean Code: PrintAllData adalah nama method yang jelas dan sesuai standar PascalCase
        // 🔧 Sebelumnya bernama PrintSemuaData
        Console.WriteLine("\n[Cetak dari data2]");
        data2.PrintAllData();

        // ✅ Clean Code: Gunakan nama method yang menjelaskan tindakan (RemoveDataAt)
        // 🔧 Sebelumnya HapusSebuahData
        data2.RemoveDataAt(2);

        Console.WriteLine("\n[Cetak dari data1 setelah penghapusan]");
        data1.PrintAllData();

        // ✅ Clean Code: Tidak ada duplikasi logika, method GetAllData() digunakan kembali
        Console.WriteLine($"\nJumlah data di data1: {data1.GetAllData().Count}");
        Console.WriteLine($"Jumlah data di data2: {data2.GetAllData().Count}");
    }
}
```

Pusatdatasingleton.cs

```
using System;
using System.Collections.Generic;

/// ✅ Clean Code: Kelas ini menerapkan pola desain Singleton dengan jelas
public class PusatDataSingleton
{
    // ✅ Clean Code: Penamaan _instance mengikuti konvensi field private di C#
    private static PusatDataSingleton _instance;

    // ✅ Clean Code: Gunakan properti publik dengan akses terbatas untuk keamanan
    // 🔄 Sebelumnya: field public biasa tanpa encapsulation
    public List<string> DataTersimpan { get; private set; }

    // ✅ Clean Code: Konstruktor private untuk mencegah pembuatan instance ganda
    private PusatDataSingleton()
    {
        // ✅ Clean Code: Inisialisasi langsung di konstruktor
        DataTersimpan = new List<string>();
    }

    /// ✅ Clean Code: Nama method GetInstance lebih deskriptif dan umum digunakan pada Singleton
    /// 🔄 Sebelumnya bernama GetDataSingleton
    public static PusatDataSingleton GetInstance()
    {
        if (_instance == null)
        {
            _instance = new PusatDataSingleton();
        }

        return _instance;
    }

    public List<string> GetAllData()
    {
        return DataTersimpan;
    }
    /// ✅ Clean Code: PrintAllData menjelaskan tindakan secara jelas
    /// 🔄 Sebelumnya PrintSemuaData
    public void PrintAllData()
    {
        if (DataTersimpan.Count == 0)
        {
            Console.WriteLine("Belum ada data.");
            return;
        }

        Console.WriteLine("Data yang tersimpan:");
        foreach (string data in DataTersimpan)
        {
            Console.WriteLine("- " + data);
        }
    }
    /// ✅ Clean Code: Nama AddData lebih konsisten dan ringkas
    /// 🔄 Sebelumnya AddSebuahData
    public void AddData(string input)
    {
        DataTersimpan.Add(input);
    }
    /// ✅ Clean Code: RemoveDataAt menjelaskan bahwa kita menghapus data berdasarkan index
    /// 🔄 Sebelumnya HapusSebuahData
    public void RemoveDataAt(int index)
    {
        if (index >= 0 && index < DataTersimpan.Count)
        {
            DataTersimpan.RemoveAt(index);
        }
        else
        {
            Console.WriteLine("Index tidak valid.");
        }
    }
}
```

*Penjelasan dalam bentuk komentar didalam baris kodingan

**SELAMAT MENGERJAKAN!**

## 3. PENGUMPULAN FILE/TUGAS PENDAHULUAN

Kumpulkan semua file berikut dalam bentuk file zip/rar/7zip:

A. Source code dari project yang dibuat

B. File docx/pdf yang berisi:

i. Screenshot hasil run

ii. Penjelasan singkat dari kode implementasi yang dibuat (beserta screenshot dari potongan source code yang dijelaskan).

**SELAMAT MENGERJAKAN!**