

**TUGAS JURNAL
MODUL 13**



Disusun Oleh :

Izzaty Zahara Br Barus – 23111040452

Kelas :

SE-07-02

Dosen :

Yudha Islami Sulistya

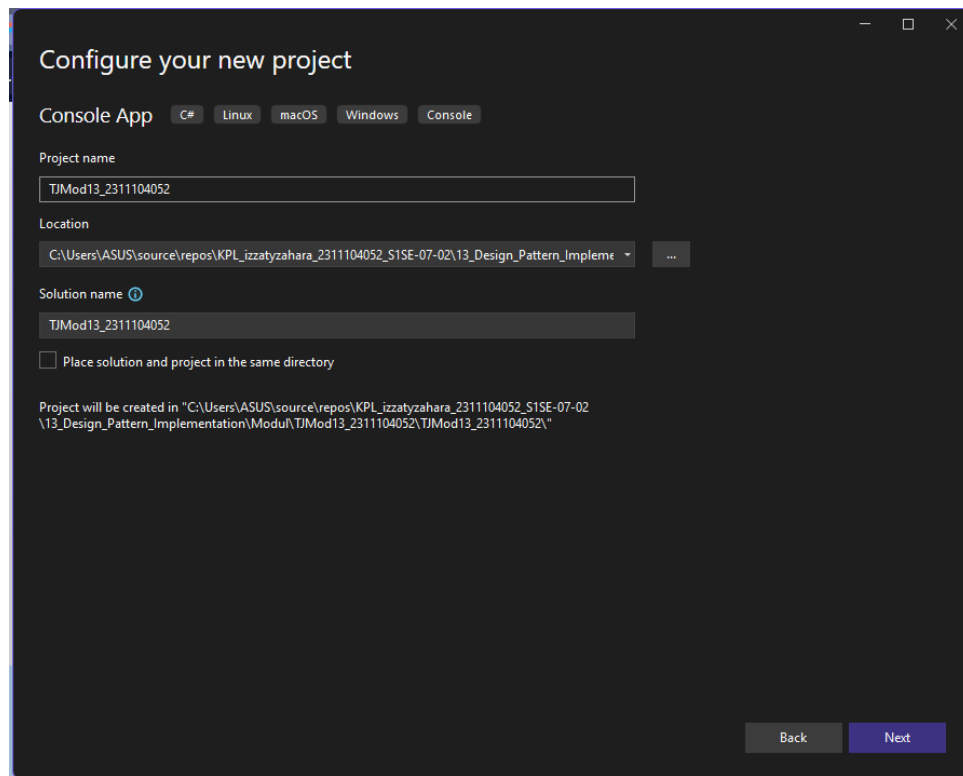
**PROGRAM STUDI SOFTWARE ENGINEERING
DIREKTORAT KAMPUS PURWOKERTO
TELKOM UNIVERSITY
PURWOKERTO
2025**

I. Link Github

● https://github.com/Izzaaaaaaaaaa/KPL_izzatyzahara_2311104052_S1SE-07-02.git

II. Penjelasan

1. Membuat New Project Menggunakan Console App dengan project name TJMod13_2311104052



2. Menambahkan syntax di class PusatDataSingleton.cs

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

namespace pusatdatasingleton
{
    public class PusatDataSingleton
    {
        private static PusatDataSingleton _instance;
        public List<string> DataTersimpan { get; set; }

        // Konstruktor private agar tidak bisa diinstansiasi dari luar
        private PusatDataSingleton()
        {
            DataTersimpan = new List<string>();
        }

        // Method Singleton
        public static PusatDataSingleton GetDataSingleton()
        {
            if (_instance == null)
```

```

        {
            _instance = new PusatDataSingleton();
        }
        return _instance;
    }

    public List<string> GetSemuaData()
    {
        return DataTersimpan;
    }

    public void PrintSemuaData()
    {
        foreach (var data in DataTersimpan)
        {
            Console.WriteLine(data);
        }
    }

    public void AddSebuahData(string input)
    {
        DataTersimpan.Add(input);
    }

    public void HapusSebuahData(int index)
    {
        if (index >= 0 && index < DataTersimpan.Count)
        {
            DataTersimpan.RemoveAt(index);
        }
        else
        {
            Console.WriteLine("Index tidak valid!");
        }
    }
}

```

Penjelasan Singkat:

Class `PusatDataSingleton` merupakan implementasi dari design pattern Singleton dalam bahasa C#. Tujuannya adalah untuk memastikan bahwa hanya ada satu objek/instance dari class ini yang digunakan selama program berjalan, dan objek tersebut dapat diakses secara global melalui method `GetDataSingleton()`.

Class ini memiliki satu properti utama yaitu `DataTersimpan`, yang bertipe `List<string>` dan berfungsi untuk menyimpan kumpulan data string. Konstruktors dari class ini dibuat private agar tidak bisa dipanggil dari luar class, yang merupakan karakteristik khas dari pattern Singleton.

Untuk mengakses instance dari class ini, digunakan method public static bernama `GetDataSingleton()`, yang akan membuat instance baru hanya jika belum pernah dibuat sebelumnya. Selain itu, class ini menyediakan beberapa method untuk mengelola isi data seperti: `AddSebuahData(string input)` untuk menambahkan data baru, `HapusSebuahData(int index)` untuk menghapus data berdasarkan indeks, `PrintSemuaData()` untuk mencetak semua data ke layar, dan `GetSemuaData()` untuk mengambil seluruh data yang tersimpan dalam bentuk list.

Dengan cara ini, meskipun kita memanggil `GetDataSingleton()` dari bagian program manapun, kita tetap akan bekerja dengan objek yang sama, sehingga semua perubahan data bersifat global dan konsisten.

3. Menambah syntax Class Program.cs

```
using pusatdatasingleton;
using System;

namespace program
{
    class Program
    {
        static void Main(string[] args)
        {
            // A. Buat dua variabel singleton
            var data1 = PusatDataSingleton.GetDataSingleton();
            var data2 = PusatDataSingleton.GetDataSingleton();

            // B. Tambahkan data ke data1
            data1.AddSebuahData("Anggota1 - Budi");
            data1.AddSebuahData("Anggota2 - Sari");
            data1.AddSebuahData("Asisten - Rian");

            // C. Print data2
            Console.WriteLine("Data di data2:");
            data2.PrintSemuaData();

            // D. Hapus nama asisten dari data2
            data2.HapusSebuahData(2); // index 2 = "Asisten - Rian"

            // E. Print ulang data1
            Console.WriteLine("\nSetelah penghapusan, data di data1:");
            data1.PrintSemuaData();

            // F. Print jumlah data dari kedua objek
            Console.WriteLine($"Jumlah data (data1):
{data1.GetSemuaData().Count}");
            Console.WriteLine($"Jumlah data (data2):
{data2.GetSemuaData().Count}");
        }
    }
}
```

```
}  
}
```

Penjelasan Singkat:

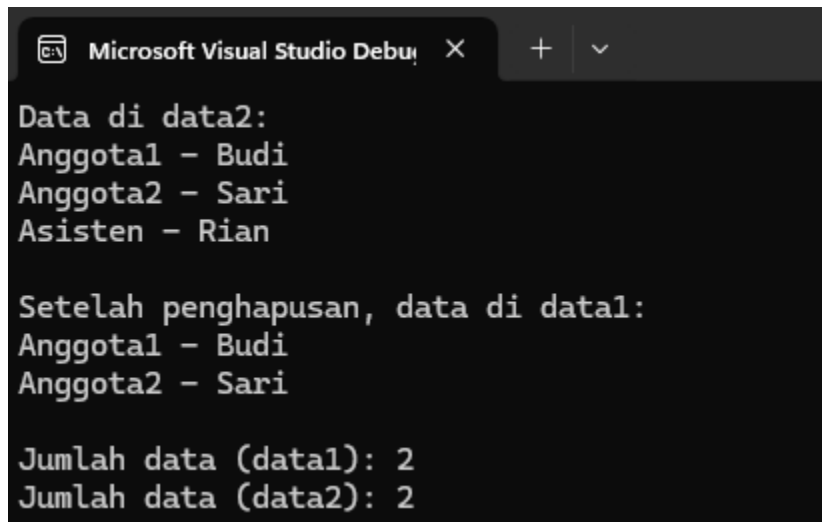
Class Program merupakan titik masuk utama aplikasi (method Main). Di dalamnya terdapat simulasi penggunaan Singleton untuk menguji apakah objek PusatDataSingleton benar-benar bekerja sebagai Singleton.

Pertama, dua variabel data1 dan data2 dideklarasikan dan masing-masing diisi dengan pemanggilan GetDataSingleton(). Karena menggunakan Singleton, data1 dan data2 akan menunjuk pada objek yang sama.

Kemudian, beberapa data ditambahkan melalui data1, termasuk nama anggota kelompok dan nama asisten praktikum. Setelah itu, data yang sama dicetak melalui data2, dan hasilnya tetap sama karena mengacu ke instance yang sama. Lalu, data nama asisten dihapus melalui data2, dan kembali dicetak melalui data1 untuk membuktikan bahwa perubahan memang bersifat global. Terakhir, program mencetak jumlah data dari data1 dan data2, yang harus sama karena keduanya adalah satu instance yang sama.

III. Hasil Running

1. Hasil Running



```
Microsoft Visual Studio Debug Console  
Data di data2:  
Anggota1 - Budi  
Anggota2 - Sari  
Asisten - Rian  
  
Setelah penghapusan, data di data1:  
Anggota1 - Budi  
Anggota2 - Sari  
  
Jumlah data (data1): 2  
Jumlah data (data2): 2
```

IV. Kesimpulan

Implementasi pattern Singleton pada class PusatDataSingleton berhasil memastikan bahwa hanya satu objek yang digunakan selama program berjalan. Hal ini dibuktikan melalui penggunaan dua variabel (data1 dan data2) yang ternyata memanipulasi dan mengakses data dari instance yang sama. Dengan pendekatan ini, kita dapat menjaga konsistensi data yang bersifat global di seluruh aplikasi.

Design pattern Singleton sangat berguna dalam skenario di mana satu sumber data bersama dibutuhkan, seperti konfigurasi aplikasi, koneksi database, ataupun sistem log. Namun, penting juga untuk memperhatikan potensi kelemahannya, seperti kesulitan dalam pengujian (unit testing) dan risiko ketergantungan global (tight coupling) bila tidak digunakan secara bijak.