

Nama : Dimastian Aji Wibowo

NIM : 2311104058

JURNAL MODUL 13

A. Design Pattern Singleton

1. Contoh Penggunaan Singleton

a. **Logger**

Digunakan untuk menyimpan instance logger tunggal agar semua bagian aplikasi mencatat ke log yang sama.

b. **Database Connection**

Design pattern singleton juga dapat digunakan untuk memastikan hanya satu koneksi database yang aktif pada satu waktu.

2. Langkah-langkah Implementasi Singleton

- a. Membuat atribut static private untuk menyimpan objek Singleton
- b. Membuat method static public untuk mendapatkan instance Singleton
- c. Membuat konstruktor kelas menjadi private
- d. Ubah semua kode lain agar tidak lagi membuat objek secara langsung, tapi pakai method static GetDataSingleton() tadi.

3. Kelebihan dan Kekurangan Singleton

a. **Kelebihan**

- Mudah diimplementasikan
- Hanya ada satu objek dari kelas tersebut.
- Menghemat memori
- Objeknya bisa diakses secara global.

b. **Kekurangan**

- Sulit untuk unit test
- Bisa terjadi bottleneck
- Melanggar prinsip Single Responsibility
- Sulit digunakan di lingkungan multi-thread

B. Implementasi

1. Source Code

PusatDataSingleton.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace modul13_2311104058
{
    public class PusatDataSingleton
    {
        private static PusatDataSingleton _instance;
        private List<string> DataTersimpan;

        private PusatDataSingleton()
        {
            DataTersimpan = new List<string>();
        }

        public static PusatDataSingleton GetDataSingleton()
        {
            if (_instance == null)
            {
                _instance = new PusatDataSingleton();
            }
            return _instance;
        }

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

        public void PrintSemuaData()
        {
            foreach (string 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);
            }
        }
    }
}
```

Kode di atas merupakan implementasi dari design pattern Singleton. Class `PusatDataSingleton` dirancang agar hanya memiliki satu instance yang dapat diakses secara global selama program berjalan. Hal ini dicapai dengan membuat konstruktor private dan menyediakan method static `GetDataSingleton()` yang akan membuat objek hanya jika belum ada instance sebelumnya. Kelas ini menyimpan data dalam bentuk list bertipe string dengan nama `DataTersimpan`. Untuk mengelola data, disediakan beberapa method publik yaitu: `AddSebuahData()` untuk menambahkan data baru ke list, `HapusSebuahData()` untuk menghapus data berdasarkan indeks tertentu, `GetSemuaData()` untuk mengambil seluruh isi list, dan

PrintSemuaData() untuk mencetak seluruh elemen yang tersimpan satu per satu ke layar. Dengan pendekatan Singleton ini, semua akses ke data dilakukan melalui satu objek yang sama, memastikan konsistensi data antar bagian program.

Program.cs

```
using modul13_2311104058;

class Program
{
    static void Main(string[] args)
    {
        // A & B
        var data1 = PusatDataSingleton.GetDataSingleton();
        var data2 = PusatDataSingleton.GetDataSingleton();

        // C. Tambah nama anggota kelompok dan asisten
        data1.AddSebuahData("Anggota 1 - Dimastian");
        data1.AddSebuahData("Anggota 2 - Fahmi");
        data1.AddSebuahData("Anggota 3 - Rifqi");
        data1.AddSebuahData("Anggota 4 - Berlian");
        data1.AddSebuahData("Asisten 1 - Kak Gideon");
        data1.AddSebuahData("Asisten 2 - Kak Devrin");

        // D. Cetak semua data melalui data2
        Console.WriteLine("Data2 - Sebelum penghapusan:");
        data2.PrintSemuaData();

        // E. Hapus dua asisten dari index 4 dan 5
        data2.HapusSebuahData(5);
        data2.HapusSebuahData(4);

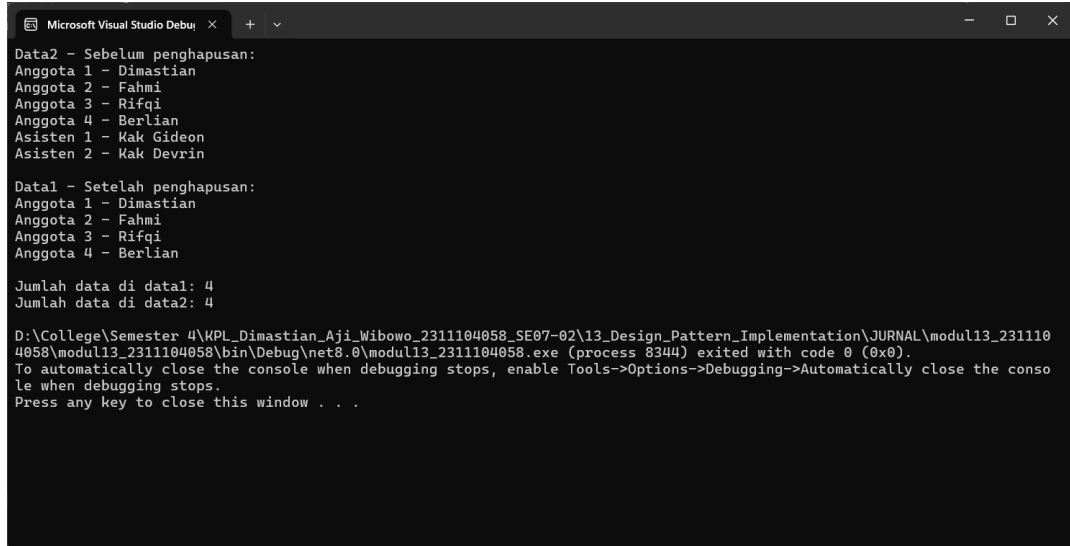
        // F. Print ulang melalui data1
        Console.WriteLine("\nData1 - Setelah penghapusan:");
        data1.PrintSemuaData();

        // G. Tampilkan jumlah elemen dari list pada kedua objek
        Console.WriteLine($"Jumlah data di data1:
{data1.GetSemuaData().Count}");
        Console.WriteLine($"Jumlah data di data2:
{data2.GetSemuaData().Count}");
    }
}
```

Kode Main() ini berfungsi untuk menguji implementasi dari class Singleton PusatDataSingleton. Pada bagian awal, dibuat dua variabel data1 dan data2 yang keduanya diisi menggunakan method GetDataSingleton(). Karena menggunakan design pattern Singleton, kedua variabel tersebut sebenarnya menunjuk ke objek yang sama. Selanjutnya, melalui data1, ditambahkan enam data berupa nama empat anggota kelompok dan dua asisten praktikum. Data ini kemudian dicetak ke konsol menggunakan PrintSemuaData() melalui variabel data2, yang seharusnya mencetak seluruh data yang sama. Kemudian, method HapusSebuahData() dipanggil untuk menghapus dua asisten dari daftar dengan mengacu pada indeks ke-5 dan ke-4 (dilakukan secara urut dari indeks terbesar agar data tidak bergeser). Setelah penghapusan, data dicetak kembali melalui data1 untuk memastikan bahwa perubahan data benar-benar bersifat global. Terakhir, program mencetak jumlah

data yang tersisa di data1 dan data2 sebagai verifikasi akhir bahwa Singleton bekerja dengan baik, di mana kedua variabel mereferensikan list yang sama.

2. Hasil Run



```
Microsoft Visual Studio Debug
Data2 - Sebelum penghapusan:
Anggota 1 - Dimastian
Anggota 2 - Fahmi
Anggota 3 - Rifqi
Anggota 4 - Berlian
Asisten 1 - Kak Gideon
Asisten 2 - Kak Devrin

Data1 - Setelah penghapusan:
Anggota 1 - Dimastian
Anggota 2 - Fahmi
Anggota 3 - Rifqi
Anggota 4 - Berlian

Jumlah data di data1: 4
Jumlah data di data2: 4

D:\College\Semester 4\KPL_Dimastian_Aji_Wibowo_2311104058_SE07-02\13_Design_Pattern_Implementation\JURNAL\modul13_2311104058\modul13_2311104058\bin\Debug\net8.0\modul13_2311104058.exe (process 8344) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```