Nama: Marvel Sanjaya Setiawan

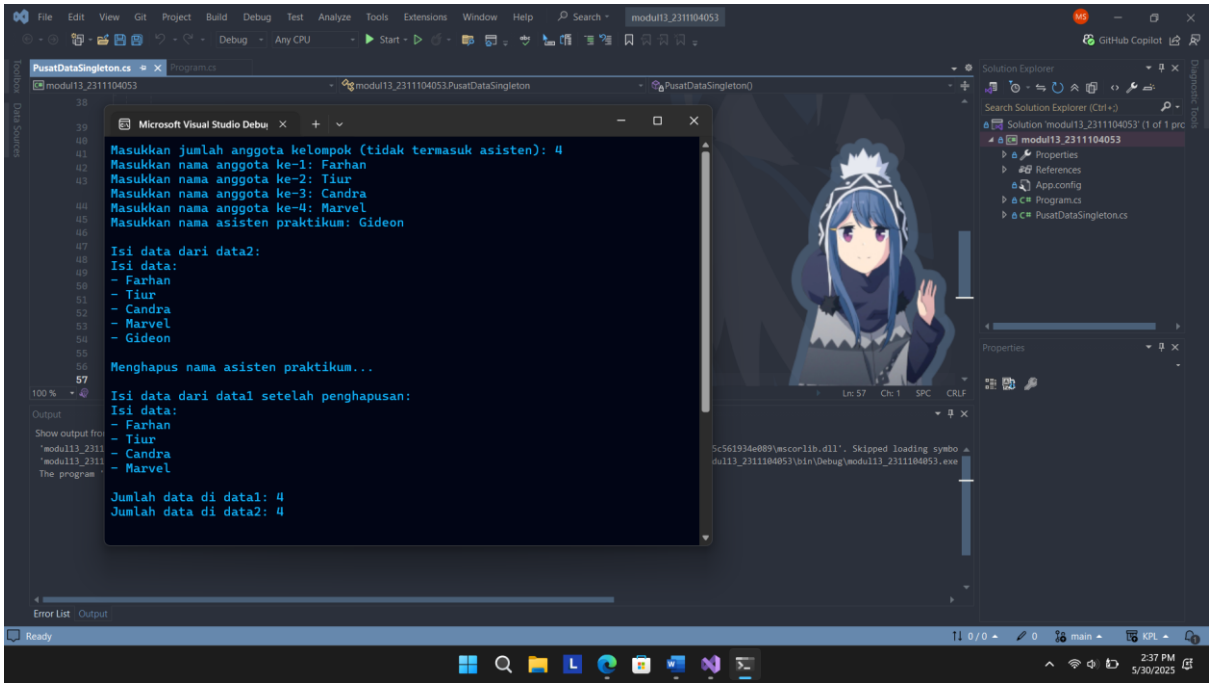
NIM: 2311104053

Kelas: SE07-02

JURNAL MODUL 13

Link github:
https://github.com/Meph1sto14/KPL_Marvel_Sanjaya_Setiawan_2311104053_SE07-02/tree/0c13ff8965ff0c2738faf744d98ad008f96f1201/13_Design_Pattern_Implementation/JURNAL/modul13_2311104053

Hasil run:



PusatDataSingleton.cs

```
1. using System;
2. using System.Collections.Generic;
3.
4. namespace modul13_2311104053
5. {
6.     public sealed class PusatDataSingleton
7.     {
8.         private static PusatDataSingleton _instance;
9.         public List<string> DataTersimpan;
10.
11.         private PusatDataSingleton()
12.         {
13.             DataTersimpan = new List<string>();
14.         }
15.
16.         public static PusatDataSingleton GetDataSingleton()
17.         {
18.             if (_instance == null)
19.             {
20.                 _instance = new PusatDataSingleton();
21.             }
22.             return _instance;
```

```
        }

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

        public void PrintSemuaData()
        {
            Console.WriteLine("Isi data:");
            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.");
            }
        }
    }
}

```

Program.cs

```
using System;

namespace modul13_2311104053
{
    class Program
    {
        static void Main(string[] args)
        {
            PusatDataSingleton data1 = PusatDataSingleton.GetDataSingleton();
            PusatDataSingleton data2 = PusatDataSingleton.GetDataSingleton();

            Console.Write("Masukkan jumlah anggota kelompok (tidak termasuk asisten): ");
            int jumlahAnggota = int.Parse(Console.ReadLine());

            for (int i = 0; i < jumlahAnggota; i++)
            {
                Console.Write($"Masukkan nama anggota ke-{i + 1}: ");
                string anggota = Console.ReadLine();
                data1.AddSebuahData(anggota);
            }

            Console.Write("Masukkan nama asisten praktikum: ");
            string asisten = Console.ReadLine();
            data1.AddSebuahData(asisten);

            Console.WriteLine("\nIsi data dari data2:");
            data2.PrintSemuaData();

            Console.WriteLine("\nMenghapus nama asisten praktikum...");
            data2.HapusSebuahData(data2.GetSemuaData().Count - 1);

```

```
            Console.WriteLine("\nIsi data dari data1 setelah penghapusan:");
            data1.PrintSemuaData();

            Console.WriteLine($"\nJumlah data di data1: {data1.GetSemuaData().Count}");
            Console.WriteLine($"Jumlah data di data2: {data2.GetSemuaData().Count}");
        }
    }
}

```

MENJELASKAN SALAH SATU DESIGN PATTERN

A. Berikan salah dua contoh kondisi dimana design pattern “Singleton” dapat digunakan.
B. Berikan penjelasan singkat mengenai langkah-langkah dalam mengimplementasikan design pattern “Singleton”.
C. Berikan tiga kelebihan dan kekurangan dari design pattern “Singleton”.

Jawaban:

A. Berikut adalah 2 contoh kondisi untuk bisa mengimplementasikan “Singleton”:
    1. Pola Singleton memastikan bahwa sebuah kelas hanya memiliki satu instance selama siklus hidup aplikasi. Hal ini berguna untuk mengontrol akses ke sumber daya bersama, seperti database atau file, sehingga tidak ada duplikasi yang dapat menyebabkan inkonsistensi atau penggunaan memori yang tidak efisien.

       Cara kerja pola ini adalah dengan membuat satu instance dari kelas dan memastikan bahwa setiap permintaan instance selanjutnya tetap mengacu pada objek yang sama. Dengan demikian, meskipun ada permintaan untuk membuat objek baru, program hanya mengembalikan instance yang telah ada sebelumnya.

       Perilaku ini tidak dapat diterapkan menggunakan konstruktor biasa, karena setiap pemanggilan konstruktor selalu menghasilkan objek baru. Oleh karena itu, pola Singleton menggunakan mekanisme seperti variabel statis dan metode akses kontrol untuk memastikan hanya ada satu instance yang dikelola oleh sistem.

    2. Pola Singleton menyediakan jalur akses global ke satu instance objek dalam sebuah program. Ini menyerupai variabel global yang digunakan untuk menyimpan data penting, tetapi lebih aman karena membatasi akses langsung dan mencegah perubahan sembarangan yang bisa menyebabkan aplikasi crash.

       Mirip dengan variabel global, Singleton memungkinkan akses ke objek dari mana saja dalam program. Namun, tidak seperti variabel global yang rentan ditimpa oleh kode lain, pola ini memastikan bahwa hanya satu instance yang dapat dibuat dan dikelola secara terkendali, menjaga integritas data.

       Selain itu, mengandalkan satu instance global yang tersebar di seluruh program dapat menyulitkan pemeliharaan. Dengan menggunakan Singleton, kode yang mengontrol akses dan pengelolaan instance tetap terorganisir dalam satu kelas, sehingga memudahkan penggunaan dan integrasi dengan bagian lain dalam program.

B. Penjelasan singkat mengenai langkah-langkah implementasi Singleton Pattern:
   1. Tambahkan bidang statis privat → Menyimpan instance tunggal di dalam kelas.

   2. Buat metode pembuatan statis publik → Mengembalikan instance tunggal ke semua pemanggilan.

   3. Gunakan inisialisasi malas → Membuat objek baru saat pertama kali dipanggil, lalu menyimpannya untuk pemanggilan berikutnya.

   4. Jadikan konstruktor privat → Mencegah instansiasi langsung di luar kelas, hanya bisa dipanggil oleh metode statis.

   5. Perbaiki kode klien → Ganti semua pemanggilan langsung ke konstruktor dengan metode pembuatan statis.

C. Kelebihan:
   1. Memastikan sebuah kelas hanya memiliki satu instance sepanjang siklus hidup aplikasi.

   2. Memberikan akses global ke instance tunggal tanpa membuat ulang objek setiap kali dibutuhkan.

   3. Menginisialisasi instance hanya saat pertama kali diminta untuk menghemat sumber daya dan memastikan konsistensi data.

   Kekurangan:
   1. Melanggar Prinsip Tanggung Jawab Tunggal → Pola Singleton menangani dua masalah sekaligus: memastikan satu instance dan menyediakan akses global, tetapi hal ini dapat menyebabkan kelas memiliki lebih dari satu tanggung jawab, melanggar prinsip desain yang baik.

   2. Dapat Menutupi Desain yang Buruk → Dalam beberapa kasus, Singleton digunakan untuk menghindari perbaikan arsitektur yang lebih baik. Misalnya, ketika komponen dalam program memiliki ketergantungan yang terlalu erat satu sama lain, sehingga sulit untuk diuji dan dikelola.

   3. Memerlukan Perlakuan Khusus dalam Multithreading → Jika tidak dikelola dengan baik dalam lingkungan multithreaded, beberapa thread dapat membuat instance Singleton lebih dari satu kali, menyebabkan inkonsistensi. Oleh karena itu, diperlukan mekanisme seperti locking atau double-checked locking untuk menghindari masalah ini.

Penjelasan Codingan:

1. PusatDataSingleton.cs:

a. Menerapkan Singleton Pattern untuk memastikan hanya ada satu instance PusatDataSingleton.
b. GetDataSingleton() → Mengembalikan instance tunggal dari kelas.
c. AddSebuahData(input) → Menambahkan data ke daftar.
d. HapusSebuahData(index) → Menghapus data berdasarkan indeks.
e. PrintSemuaData() → Menampilkan seluruh data yang tersimpan.

2. Program.cs:
   a. Mengambil instance PusatDataSingleton (dua variabel data1 dan data2 merujuk ke objek yang sama).
   b. Meminta pengguna memasukkan nama anggota kelompok dan nama asisten.
   c. Menampilkan daftar anggota kelompok.
   d. Menghapus asisten dari daftar.
   e. Menampilkan kembali daftar setelah penghapusan.