

**TUGAS JURNAL
KONSTRUKSI PERANGKAT LUNAK
Modul XIII
DESIGN PATTERN IMPLEMENTATION**



**Disusun Oleh :
Ahmad Junaidi / 2211104002
SE-06-01**

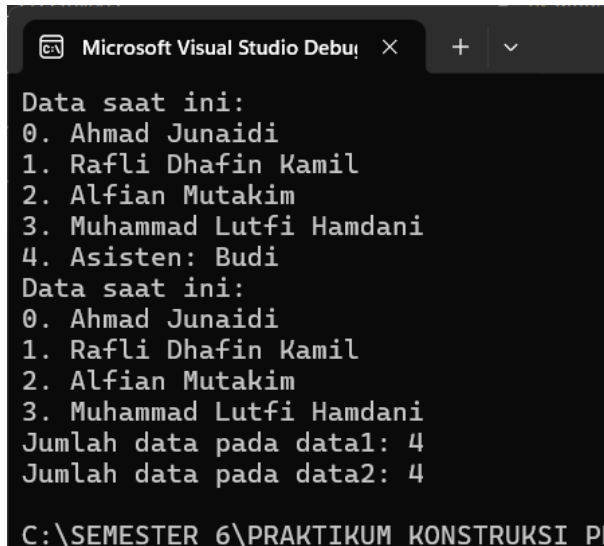
**Asisten Praktikum:
Naufal El Kamil Aditya Pratama Rahman
Imelda**

**Dosen Pengampu :
Yudha Islami Sulistya, S.Kom., M.Cs.**

**PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

TUGAS JURNAL

1. Screenshoot Hasil Run

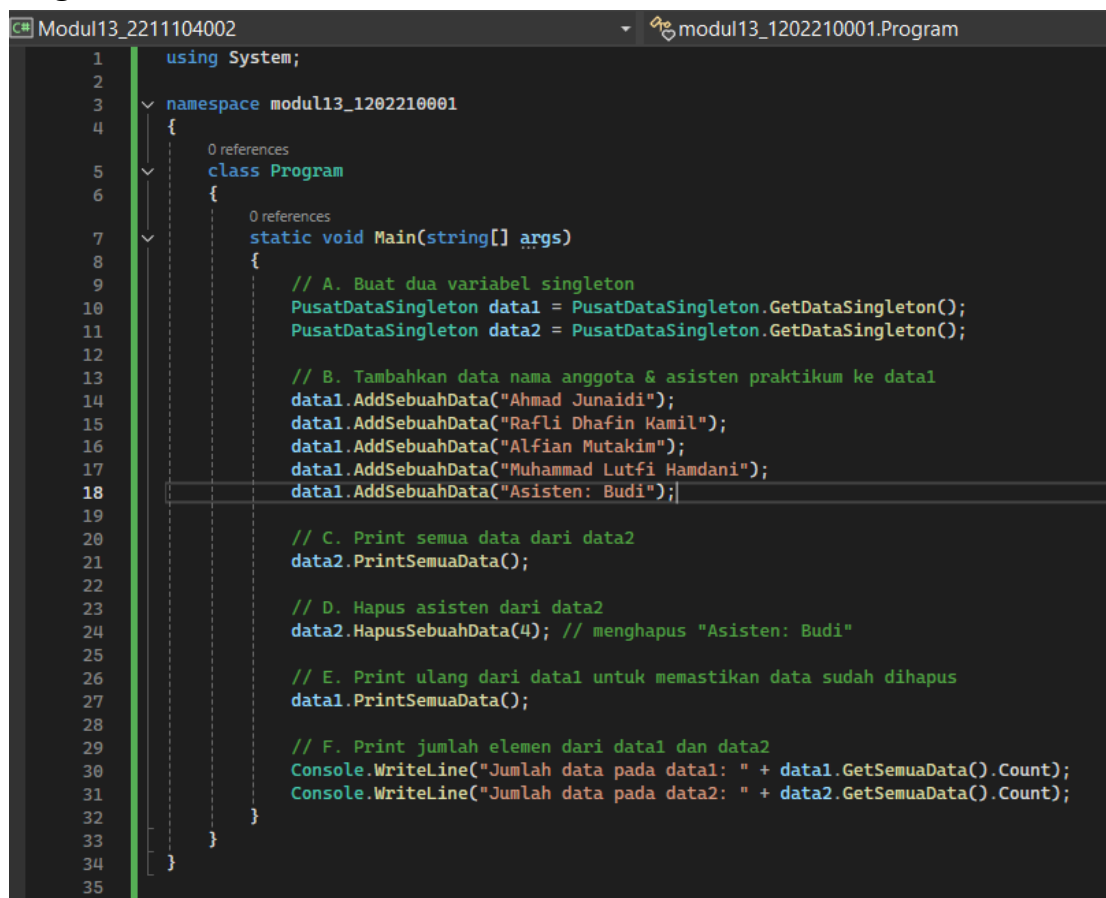


```
Microsoft Visual Studio Debug Console
Data saat ini:
0. Ahmad Junaidi
1. Rafli Dhafin Kamil
2. Alfian Mutakim
3. Muhammad Lutfi Hamdani
4. Asisten: Budi
Data saat ini:
0. Ahmad Junaidi
1. Rafli Dhafin Kamil
2. Alfian Mutakim
3. Muhammad Lutfi Hamdani
Jumlah data pada data1: 4
Jumlah data pada data2: 4
C:\SEMESTER 6\PRAKTIKUM KONSTRUKSI P
```

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

Source Code :

Program.cs:



```
Modul13_2211104002 modul13_1202210001.Program
1 using System;
2
3 namespace modul13_1202210001
4 {
5     0 references
6     class Program
7     {
8         0 references
9         static void Main(string[] args)
10        {
11            // A. Buat dua variabel singleton
12            PusatDataSingleton data1 = PusatDataSingleton.GetDataSingleton();
13            PusatDataSingleton data2 = PusatDataSingleton.GetDataSingleton();
14
15            // B. Tambahkan data nama anggota & asisten praktikum ke data1
16            data1.AddSebuahData("Ahmad Junaidi");
17            data1.AddSebuahData("Rafli Dhafin Kamil");
18            data1.AddSebuahData("Alfian Mutakim");
19            data1.AddSebuahData("Muhammad Lutfi Hamdani");
20            data1.AddSebuahData("Asisten: Budi");
21
22            // C. Print semua data dari data2
23            data2.PrintSemuaData();
24
25            // D. Hapus asisten dari data2
26            data2.HapusSebuahData(4); // menghapus "Asisten: Budi"
27
28            // E. Print ulang dari data1 untuk memastikan data sudah dihapus
29            data1.PrintSemuaData();
30
31            // F. Print jumlah elemen dari data1 dan data2
32            Console.WriteLine("Jumlah data pada data1: " + data1.GetSemuaData().Count);
33            Console.WriteLine("Jumlah data pada data2: " + data2.GetSemuaData().Count);
34        }
35    }
36 }
```

PusatDataSingleton.cs

```
Modul13_2211104002 modul13_1202210001.PusatDataSi

1  using System;
2  using System.Collections.Generic;
3
4  namespace modul13_1202210001
5  {
6      8 references
7      public class PusatDataSingleton
8      {
9          private static PusatDataSingleton _instance;
10         7 references
11         public List<string> DataTersimpan { get; set; }
12
13         // Konstruktor private
14         1 reference
15         private PusatDataSingleton()
16         {
17             DataTersimpan = new List<string>();
18         }
19
20         // Method singleton
21         2 references
22         public static PusatDataSingleton GetDataSingleton()
23         {
24             if (_instance == null)
25             {
26                 _instance = new PusatDataSingleton();
27             }
28             return _instance;
29         }
30
31         2 references
32         public List<string> GetSemuaData()
33         {
34             return DataTersimpan;
35         }
36
37         2 references
38         public void PrintSemuaData()
39         {
40             Console.WriteLine("Data saat ini:");
41             for (int i = 0; i < DataTersimpan.Count; i++)
42             {
43                 Console.WriteLine($"{i}. {DataTersimpan[i]}");
44             }
45         }
46
47         5 references
48         public void AddSebuahData(string input)
49         {
50             DataTersimpan.Add(input);
51         }
52
53         1 reference
54         public void HapusSebuahData(int index)
55         {
56             if (index >= 0 && index < DataTersimpan.Count)
57             {
58                 DataTersimpan.RemoveAt(index);
59             }
60             else
61             {
62                 Console.WriteLine("Index tidak valid.");
63             }
64         }
65     }
66 }
```

penjelasan:

1. Class Singleton dibuat dengan private constructor dan static instance.
2. GetDataSingleton() memastikan hanya ada satu instance dari PusatDataSingleton.
3. Semua metode (Add, Print, Get, Hapus) memanipulasi satu instance list yang sama.
4. Contoh program utama menunjukkan bahwa data1 dan data2 adalah referensi ke objek yang sama.
5. Jumlah data selalu sinkron antara data1 dan data2 karena Singleton memastikan hanya ada satu objek.

Pada praktikum Modul 13 mata kuliah Konstruksi Perangkat Lunak, mahasiswa diminta untuk memahami dan mengimplementasikan design pattern bernama Singleton menggunakan bahasa pemrograman C# di dalam Visual Studio. Desain ini digunakan untuk memastikan bahwa sebuah kelas hanya memiliki satu objek (instance) yang dapat digunakan secara global di seluruh program. Pada tugas ini, dibuat sebuah project Console App dengan nama modul13_NIM (disesuaikan dengan NIM masing-masing). Di dalamnya, dibuat kelas PusatDataSingleton yang memiliki dua atribut utama: `_instance` bertipe `PusatDataSingleton` sebagai properti singleton, dan `DataTersimpan` bertipe `List<string>` sebagai wadah untuk menyimpan data nama. Kelas ini menerapkan konstruktor privat agar hanya bisa diakses melalui method `GetDataSingleton()` yang memastikan hanya satu objek pernah dibuat. Kemudian, terdapat beberapa method lain yaitu `AddSebuahData()` untuk menambah data ke list, `HapusSebuahData()` untuk menghapus data berdasarkan indeks, `GetSemuaData()` untuk mengembalikan isi list, dan `PrintSemuaData()` untuk menampilkan data yang tersimpan. Pada fungsi Main, dibuat dua variabel yaitu `data1` dan `data2`, yang keduanya diisi menggunakan `GetDataSingleton()`. Setelah itu, data nama anggota kelompok dan asisten praktikum ditambahkan melalui `data1`, lalu ditampilkan menggunakan `data2`. Kemudian data asisten dihapus melalui `data2`, dan saat data ditampilkan kembali melalui `data1`, hasilnya sudah tidak terdapat nama asisten, membuktikan bahwa keduanya mengakses objek yang sama. Program diakhiri dengan mencetak jumlah data dari `data1` dan `data2` yang bernilai sama. Dari implementasi ini, dapat disimpulkan bahwa Singleton Pattern memberikan kemudahan untuk mengelola satu sumber data secara global dan konsisten dalam satu siklus hidup aplikasi.