

Nama : Candra Dinata  
Nim : 2311104061  
Kelas : SE0702

**A. Dua contoh kondisi penggunaan Singleton:**

1. Koneksi Database: Untuk memastikan hanya ada satu koneksi terbuka dalam satu waktu.
2. Logger (Pencatatan Log): Untuk menyimpan catatan kejadian dalam satu instance logger yang sama.

**B. Langkah-langkah implementasi Singleton:**

1. Buat konstruktor private agar tidak bisa diakses dari luar kelas.
2. Tambahkan atribut static yang menyimpan instance dari kelas itu sendiri.
3. Buat method public static untuk mengakses instance tersebut (misalnya `GetInstance()`), yang membuat instance baru hanya jika belum ada.

**C. Kelebihan dan Kekurangan Singleton:**

Kelebihan	Kekurangan
Menghindari duplikasi objek	Menyulitkan testing unit karena instance global
Mudah diakses dari manapun	Potensi menjadi God Object jika disalahgunakan
Menghemat resource	Tidak thread-safe jika tidak diatur

## PusatDataSingleton.cs

```
1  using System;
2  using System.Collections.Generic;
3
4  public class PusatDataSingleton
5  {
6      private static PusatDataSingleton _instance;
7      public List<string> DataTersimpan { get; set; }
8
9      // Private constructor
10     private PusatDataSingleton()
11     {
12         DataTersimpan = new List<string>();
13     }
14
15     // Method untuk mendapatkan instance (singleton)
16     public static PusatDataSingleton GetDataSingleton()
17     {
18         if (_instance == null)
19         {
20             _instance = new PusatDataSingleton();
21         }
22         return _instance;
23     }
24
25     public List<string> GetSemuaData()
26     {
27         return DataTersimpan;
28     }
29
30     public void PrintSemuaData()
31     {
32         foreach (string data in DataTersimpan)
33         {
34             Console.WriteLine(data);
35         }
36     }
37
38     public void AddSebuahData(string input)
39     {
40         DataTersimpan.Add(input);
41     }
42
43     public void HapusSebuahData(int index)
44     {
45         if (index >= 0 && index < DataTersimpan.Count)
46         {
47             DataTersimpan.RemoveAt(index);
48         }
49         else
50         {
51             Console.WriteLine("Index tidak valid.");
52         }
53     }
54 }
55
```

Kode tersebut merupakan implementasi pola desain **Singleton** dalam bahasa C#, yang bertujuan untuk memastikan hanya ada satu objek dari class `PusatDataSingleton` yang dibuat selama runtime aplikasi. Class ini memiliki atribut `DataTersimpan` berupa `List<string>` yang berfungsi untuk menyimpan data string. Konstruktor class bersifat `private`, sehingga tidak bisa diakses langsung dari luar class dan mencegah pembuatan instance baru secara bebas. Akses ke instance tunggal disediakan melalui method static `GetDataSingleton()`, yang akan membuat instance baru hanya jika belum ada sebelumnya (lazily initialized). Class ini juga memiliki beberapa method publik seperti `AddSebuahData(string input)` untuk menambahkan data ke list, `HapusSebuahData(int index)` untuk menghapus data berdasarkan indeks, `PrintSemuaData()` untuk mencetak seluruh isi list ke console,

dan GetSemuaData() untuk mengembalikan seluruh data yang tersimpan dalam list. Karena instance yang digunakan bersifat tunggal, setiap perubahan data melalui objek apapun yang mengakses Singleton ini akan saling terhubung dan konsisten.

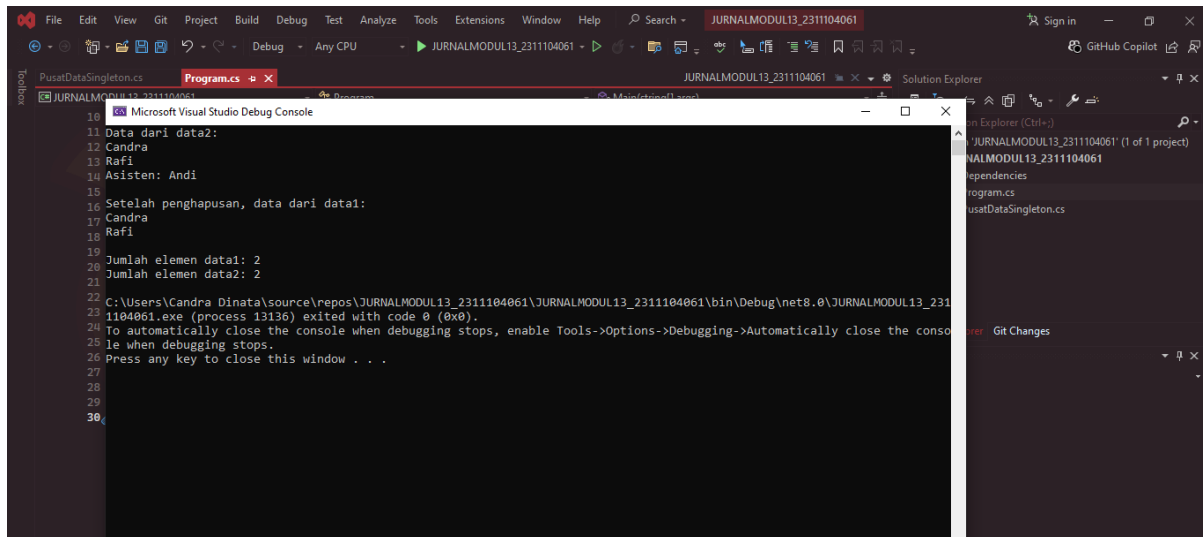
#### Program.cs

```
1  class Program
2  {
3      static void Main(string[] args)
4      {
5          // A & B
6          var data1 = PusatDataSingleton.GetDataSingleton();
7          var data2 = PusatDataSingleton.GetDataSingleton();
8
9          // C - Tambah anggota kelompok dan asisten
10         data1.AddSebuahData("Candra");
11         data1.AddSebuahData("Rafi");
12         data1.AddSebuahData("Asisten: Andi");
13
14         // D - Print dari data2
15         Console.WriteLine("Data dari data2:");
16         data2.PrintSemuaData();
17
18         // E - Hapus nama asisten
19         data2.HapusSebuahData(2); // Index ke-2 adalah "Asisten: Andi"
20
21         // F - Print ulang dari data1 (cek apakah asisten terhapus)
22         Console.WriteLine("\nSetelah penghapusan, data dari data1:");
23         data1.PrintSemuaData();
24
25         // G - Print jumlah data
26         Console.WriteLine($"Jumlah elemen data1: {data1.GetSemuaData().Count}");
27         Console.WriteLine($"Jumlah elemen data2: {data2.GetSemuaData().Count}");
28     }
29 }
30
```

Kode di atas merupakan method Main() dalam class Program yang berfungsi sebagai titik awal eksekusi program. Kode ini mendemonstrasikan penggunaan pola desain **Singleton** melalui class PusatDataSingleton. Pertama, dibuat dua variabel data1 dan data2 yang keduanya diisi menggunakan method GetDataSingleton(), sehingga meskipun tampak seperti dua objek berbeda, keduanya sebenarnya merujuk pada instance yang sama. Selanjutnya, melalui data1, ditambahkan tiga string ke dalam list DataTersimpan, yaitu dua nama anggota kelompok dan satu nama asisten. Kemudian, data ditampilkan melalui data2, yang menunjukkan bahwa kedua variabel mengakses sumber data yang sama. Setelah itu, elemen dengan indeks ke-2 (nama asisten) dihapus menggunakan data2, dan dilakukan pengecekan dengan mencetak ulang isi data menggunakan data1, yang membuktikan bahwa perubahan pada satu variabel berdampak pada yang lain. Terakhir, program mencetak jumlah elemen dalam list dari kedua variabel, yang menunjukkan jumlah yang

sama, mengonfirmasi bahwa Singleton bekerja sebagaimana mestinya dalam menjaga satu-satunya instance yang konsisten di seluruh program.

## Output



```
10
11 Data dari data2:
12 Candra
13 Rafi
14 Asisten: Andi
15
16 Setelah penghapusan, data dari data1:
17 Candra
18 Rafi
19
20 Jumlah elemen data1: 2
21 Jumlah elemen data2: 2
22
23 C:\Users\Candra Dinata\source\repos\JURNALMODUL13_2311104061\JURNALMODUL13_2311104061\bin\Debug\net8.0\JURNALMODUL13_2311104061.exe (process 13136) exited with code 0 (0x0).
24 To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
25 Press any key to close this window . . .
26
27
28
29
30
```