

Nama : abdul roni_ SE063 _2211104080

1. MENJELASKAN DESIGN PATTERN SINGLETON

Buka halaman web <https://refactoring.guru/design-patterns/catalog> kemudian baca design pattern dengan nama “Singleton”, dan jawab pertanyaan berikut ini (dalam Bahasa Indonesia):

A. Berikan salah dua contoh kondisi dimana design pattern “Singleton” dapat digunakan.

Jawab :

- Koneksi database — Dalam sebuah aplikasi, hanya dibutuhkan satu koneksi database yang dapat diakses dari mana saja tanpa membuat koneksi baru berkali-kali.
- Logger (Pencatat log) — Sistem pencatatan log biasanya hanya memerlukan satu instance logger untuk mengelola dan menyimpan catatan aktivitas aplikasi secara terpusat.

B. Berikan penjelasan singkat mengenai langkah-langkah dalam mengimplementasikan design pattern “Singleton”.

Jawab :

- Buat konstruktor kelas menjadi private agar objek dari luar kelas tidak bisa dibuat langsung.
- Buat sebuah atribut statis private yang menyimpan instance tunggal dari kelas tersebut.
- Buat method statis public, misal `GetInstance()`, yang berfungsi untuk mengembalikan instance tunggal, jika instance belum ada maka method ini akan membuatnya terlebih dahulu (lazy initialization).
- Semua akses ke instance Singleton harus melalui method ini sehingga memastikan hanya satu instance yang dibuat dan digunakan di seluruh aplikasi.

C. Berikan tiga kelebihan dan kekurangan dari design pattern “Singleton”.

Jawab :

Kelebihan:

- Menjamin hanya ada satu instance objek yang dibuat di seluruh aplikasi.
- Memberikan titik akses global ke instance tersebut.
- Menghemat sumber daya karena instance dapat digunakan kembali tanpa membuat objek baru.

Kekurangan:

- Melanggar prinsip tanggung jawab tunggal (single responsibility principle) karena mengelola instance sekaligus logika bisnis.
- Bisa menyebabkan kesulitan pada pengujian unit (unit testing) karena ketergantungan global.
- Dalam lingkungan multithread, perlu penanganan khusus agar instance tidak dibuat ganda (konkurensi/locking).

2. IMPLEMENTASI DAN PEMAHAMAN DESIGN PATTERN SINGLETON

Buka halaman web berikut <https://refactoring.guru/design-patterns/singleton> dan scroll ke bagian “Code

Examples”, pilih kode yang akan dilihat misalnya C# dan ikuti langkah-langkah berikut:

A. Dengan contoh yang sudah diberikan, buatlah sebuah class dengan design pattern singleton dengan nama “PusatDataSingleton”.

B. Class “PusatDataSingleton” mempunyai dua atribut yaitu “DataTersimpan” yang mempunyai tipe berupa List<string> dan property singleton dengan nama “_instance” dengan tipe data “PusatDataSingleton” itu sendiri.

C. Class tersebut juga memiliki beberapa method yaitu:

- i. Kontruktor dari kelas tersebut yang mengisi atribut “DataTersimpan” dengan list kosong.
- ii. GetDataSingleton() yang mengembalikan “_instance” jika tidak null dan memanggil konstruktor terlebih dahulu apabila nilainya masih null.
- iii. GetSemuaData() yang mengembalikan list dari property “DataTersimpan”.
- iv. PrintSemuaData() yang melakukan print satu per satu dari string yang ada di list

“DataTersimpan”.

v. AddSebuahData(string input) yang menambahkan satu data baru “input” ke dalam list “DataTersimpan”.

vi. HapusSebuahData(int index) yang menghapus sebuah data berdasarkan index tertentu.

Jawab :

Program.cs

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

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

    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 (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);  
    }  
}  
}
```

Penjelasan kode

Kode di atas adalah implementasi pola desain **Singleton** dalam bahasa C#. Kelas `PusatDataSingleton` memastikan hanya ada satu instance (objek) yang dibuat selama program berjalan. Ini dilakukan dengan:

- Membuat konstruktor `private`, sehingga objek tidak bisa dibuat dari luar kelas.
- Menyediakan method statis `GetDataSingleton()` yang mengembalikan instance tunggal (`_instance`), dan membuatnya hanya jika belum ada.

Kelas ini menyimpan data dalam list `DataTersimpan` dan menyediakan beberapa method:

- `AddSebuahData(string)` untuk menambahkan data.
- `HapusSebuahData(int)` untuk menghapus data berdasarkan indeks.
- `GetSemuaData()` untuk mengambil seluruh data dalam bentuk list.
- `PrintSemuaData()` untuk mencetak semua data ke konsol.

Dengan menggunakan pola `Singleton`, semua bagian program yang memanggil `GetDataSingleton()` akan mengakses dan memodifikasi objek yang sama, menjaga konsistensi data pusat.

3. IMPLEMENTASI PROGRAM UTAMA

Tambahkan beberapa implementasi di program/method utama atau “main”:

- A. Buatlah dua variable dengan tipe “PusatDataSingleton” bernama data1 dan data2.
- B. Isi kedua variable tersebut dengan hasil keluaran dari GetDataSingleton().
- C. Pada data1 lakukan pemanggilan method AddSebuahData() beberapa kali dengan input nama anggota kelompok dan asisten praktikum.
- D. Pada data2 panggil method PrintSemuaData(), pastikan keluaran dari hasil print data2 menampilkan nama-nama anggota kelompok dan asisten praktikum.
- E. Pada data2 panggil HapusSebuahData() untuk menghapus nama asisten praktikum anda sekarang.
- F. Pada data1 panggil PrintSemuaData(), dan seharusnya nama asisten praktikum anda tidak muncul di hasil print tersebut.
- G. Langkah terakhir, pada data1 dan data2 panggil GetSemuaData() dan lakukan print dari jumlah
“Count” atau elemen yang ada di list pada data1 dan data2.

Jawab:

Program.cs

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

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

    // Konstruktor privat untuk mencegah pembuatan instance secara langsung
    private PusatDataSingleton()
    {
        DataTersimpan = new List<string>();
    }

    // Mendapatkan instance tunggal dari kelas PusatDataSingleton
    public static PusatDataSingleton GetDataSingleton()
    {
        if (_instance == null)
        {
            _instance = new PusatDataSingleton();
        }
    }
}
```

```

        return _instance;
    }

    // Mengembalikan seluruh data yang tersimpan
    public List<string> GetSemuaData()
    {
        return DataTersimpan;
    }

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

    // Menambahkan data baru ke dalam list
    public void AddSebuahData(string input)
    {
        DataTersimpan.Add(input);
    }

    // Menghapus data berdasarkan index yang diberikan
    public void HapusSebuahData(int index)
    {
        if (index >= 0 && index < DataTersimpan.Count)
        {
            DataTersimpan.RemoveAt(index);
        }
    }
}

class Program
{
    static void Main(string[] args)
    {
        // Membuat dua variabel dengan tipe PusatDataSingleton
        PusatDataSingleton data1 = PusatDataSingleton.GetDataSingleton();
        PusatDataSingleton data2 = PusatDataSingleton.GetDataSingleton();

        // Menambahkan data anggota kelompok dan asisten praktikum ke
        data1.AddSebuahData("Anggota 1");
        data1.AddSebuahData("Anggota 2");
        data1.AddSebuahData("Asisten Praktikum");
    }
}

```

```
// Mencetak semua data dari data2  
Console.WriteLine("Data di data2:");
```



```

    data2.PrintSemuaData();

    // Menghapus nama asisten praktikum
    data2.HapusSebuahData(2); // Indeks asisten praktikum adalah 2

    // Mencetak data setelah penghapusan dari data1
    Console.WriteLine("\nData di data1 setelah penghapusan:");
    data1.PrintSemuaData();

    // Mencetak jumlah data dari data1 dan data2
    Console.WriteLine("\nJumlah data di data1: " + data1.GetSemuaData().Count);
    Console.WriteLine("Jumlah data di data2: " + data2.GetSemuaData().Count);
}
}

```

Penjelasan Kode

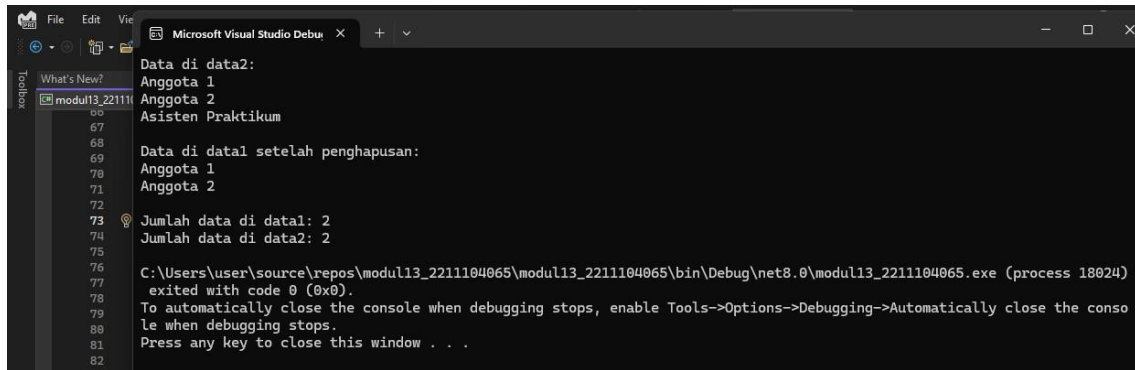
Kode di atas merupakan implementasi pola desain **Singleton** dalam C# melalui kelas PusatDataSingleton. Pola ini memastikan hanya **satu instance** objek yang digunakan di seluruh program.

Penjelasan Singkat:

- Objek data1 dan data2 sama-sama mengakses instance tunggal dari PusatDataSingleton melalui GetDataSingleton(), sehingga **semua perubahan pada satu objek akan terlihat di objek lainnya**.
- Program menambahkan tiga data ke data1, lalu mencetak isinya melalui data2 (hasilnya tetap terlihat karena mereka berbagi instance).
- Data “Asisten Praktikum” dihapus melalui data2, lalu hasilnya dicek kembali melalui data1, dan data sudah terhapus.
- Di akhir, program mencetak jumlah data di data1 dan data2, dan hasilnya sama, menunjukkan bahwa keduanya adalah instance yang sama.

Kesimpulan: Kode ini menunjukkan bagaimana Singleton menjaga **konsistensi data global** dengan hanya satu objek yang dipakai bersama di seluruh program.

Output



The screenshot shows the Microsoft Visual Studio Debug Console window. On the left, the 'Toolbox' pane is visible with 'What's New?' and 'modul13_22111' selected. The main console area displays the following output:

```
Data di data2:  
Anggota 1  
Anggota 2  
Asisten Praktikum  
  
Data di data1 setelah penghapusan:  
Anggota 1  
Anggota 2  
  
Jumlah data di data1: 2  
Jumlah data di data2: 2  
  
C:\Users\user\source\repos\modul13_2211104065\modul13_2211104065\bin\Debug\net8.0\modul13_2211104065.exe (process 18024)  
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 . . .
```

