TP MODUL 13

DESIGN PATTERN IMPLEMENTATION

Nama :

Dhiemas Tulus Ikhsan (2311104046)

Dosen :

Yudha Islami Sulistya, S.Kom., M.Cs

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

TELKOM UNIVERSITY PURWOKERTO

2025

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.
      1. Koneksi Database: Dalam aplikasi yang memerlukan akses ke database, kita hanya ingin memiliki satu koneksi database yang digunakan oleh seluruh bagian aplikasi. Dengan menggunakan pattern Singleton, kita dapat memastikan bahwa hanya ada satu instance dari koneksi database yang digunakan, sehingga menghindari overhead dari membuat banyak koneksi.
      2. Pengaturan Konfigurasi Aplikasi: Ketika aplikasi memerlukan pengaturan konfigurasi yang dibaca dari file atau sumber lain, kita bisa menggunakan Singleton untuk memastikan bahwa hanya ada satu instance dari objek konfigurasi yang diakses di seluruh aplikasi. Ini membantu menjaga konsistensi pengaturan dan menghindari konflik yang mungkin terjadi jika ada beberapa instance yang berbeda.

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

      Tambahkan Field Statis: Buatlah field statis di dalam kelas untuk menyimpan instance dari kelas Singleton tersebut.

      1. Deklarasikan Metode Statis: Buatlah metode statis publik yang akan digunakan untuk mendapatkan instance dari kelas Singleton. Metode ini akan memeriksa apakah instance sudah ada; jika belum, maka akan dibuat.
      2. Implementasikan "Lazy Initialization": Di dalam metode statis, jika instance belum ada, buatlah objek baru dan simpan di field statis. Pastikan metode ini selalu mengembalikan instance yang sama pada panggilan berikutnya.
      3. Buat Konstruktor Privat: Pastikan konstruktor kelas tersebut bersifat privat agar tidak bisa diakses dari luar kelas. Hanya metode statis yang dapat memanggil konstruktor ini.
      4. Ganti Panggilan Konstruktor: Periksa kode klien dan ganti semua panggilan langsung ke konstruktor dengan panggilan ke metode statis yang telah dibuat.

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

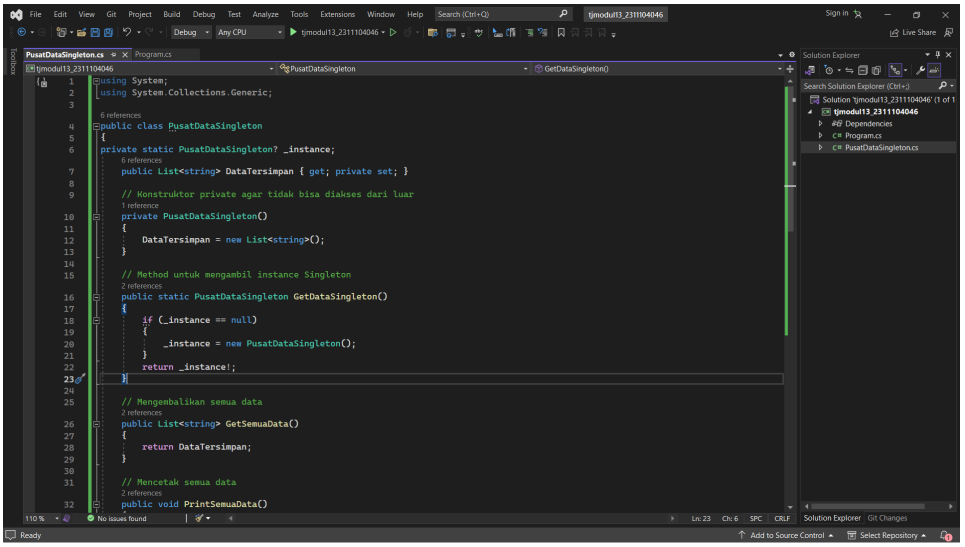
      Kelebihan:

1. Hanya Satu Instance: Memastikan bahwa hanya ada satu instance dari kelas, sehingga menghindari masalah yang mungkin timbul dari memiliki banyak instance.
2. Akses Global: Memberikan titik akses global ke instance tersebut, sehingga memudahkan penggunaan di seluruh aplikasi.
3. Inisialisasi Tertunda: Singleton hanya diinisialisasi saat pertama kali diminta, yang dapat menghemat sumber daya jika instance tidak selalu diperlukan.

Kekurangan:

1. Melanggar Prinsip Tanggung Jawab Tunggal: Singleton menyelesaikan dua masalah sekaligus, yang dapat membuat kode menjadi lebih sulit untuk dipahami dan dikelola.
2. Kesulitan dalam Pengujian Unit: Karena konstruktor privat, sulit untuk membuat mock objek untuk pengujian, yang dapat menghambat pengujian unit yang efektif.
3. Masalah dalam Lingkungan Multithread: Memerlukan penanganan khusus dalam lingkungan multithread untuk mencegah beberapa thread membuat instance yang sama secara bersamaan, yang dapat menyebabkan perilaku tidak terduga.

2. Implementasi dan Pemahaman Design Pattern Singleton



a. 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.
b. Class tersebut juga memiliki beberapa method yaitu:
    1) Konstruktor mengisi DataTersimpan dengan list kosong

```
private PusatDataSingleton()
```

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

Konstruktor ini bersifat private agar hanya bisa dipanggil dari dalam class, dan inisialisasi DataTersimpan dengan list kosong (new List<string>()).

2) Method GetDataSingleton()

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

Method ini akan mengembalikan satu-satunya instance (_instance). Jika belum dibuat, maka akan dipanggil konstruktor untuk membuat instance tersebut.

3) Method GetSemuaData()

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

Method ini mengembalikan seluruh data yang disimpan di list DataTersimpan.

4) Method PrintSemuaData()

```
public void PrintSemuaData()
{
    foreach (string data in DataTersimpan)
    {
        Console.WriteLine(data);
    }
}
```

Method ini mencetak satu per satu elemen dalam list DataTersimpan ke layar menggunakan Console.WriteLine().

5) Method AddSebuahData(string input)

```
public void AddSebuahData(string input)
{
    DataTersimpan.Add(input);
}
```

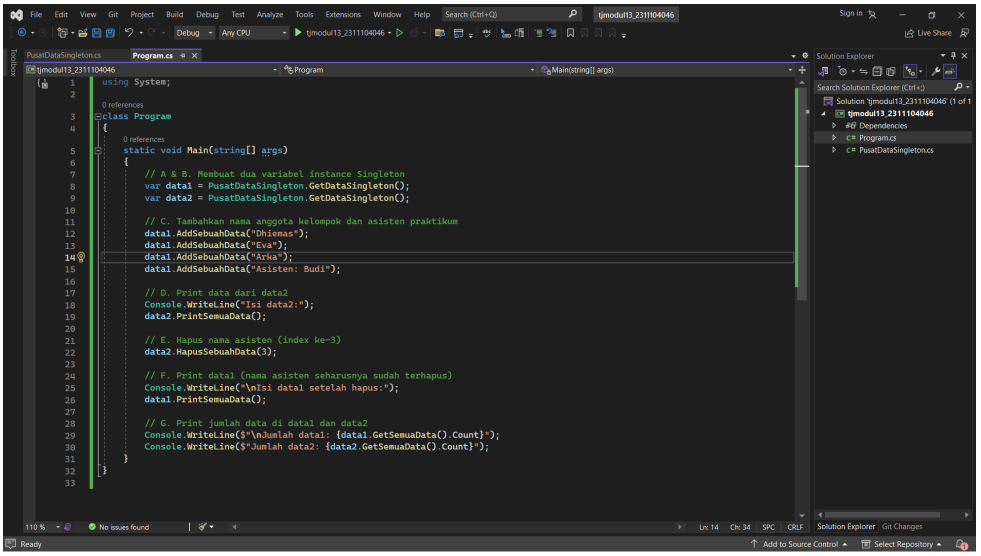
Method ini menambahkan elemen baru ke dalam list DataTersimpan.

6) Method HapusSebuahData(int index)

```
public void HapusSebuahData(int index)
{
    if (index >= 0 && index < DataTersimpan.Count)
    {
        DataTersimpan.RemoveAt(index);
    }
}
```

Method ini menghapus elemen dari list berdasarkan index tertentu. Dicek dulu agar tidak terjadi error jika index tidak valid.

3. Implementasi Program Utama



a. Membuat & mengisi dua variabel Singleton

```
var data1 = PusatDataSingleton.GetDataSingleton();
var data2 = PusatDataSingleton.GetDataSingleton();
```

data1 dan data2 keduanya mengambil instance dari Singleton. Harusnya instance-nya sama.

b. Menambahkan data ke data1

```
data1.AddSebuahData("Dhiemas");
data1.AddSebuahData("Eva");
data1.AddSebuahData("Arka");
data1.AddSebuahData("Asisten: Budi");
```

Menambahkan nama anggota kelompok dan asisten ke data1.

c. Mencetak data dari data2

```
Console.WriteLine("Isi data2:");
data2.PrintSemuaData();
```

Walaupun ditambahkan lewat data1, karena ini Singleton, data2 mencetak isi yang sama.

d. Menghapus data (asisten) lewat data2

```
data2.HapusSebuahData(3); // Menghapus "Asisten: Budi"
```

Menghapus nama asisten dari list menggunakan index ke-3 (sesuai urutan yang ditambahkan).

e. Mencetak data dari data1 setelah penghapusan

```
Console.WriteLine("\nIsi data1 setelah hapus:");
data1.PrintSemuaData();
```

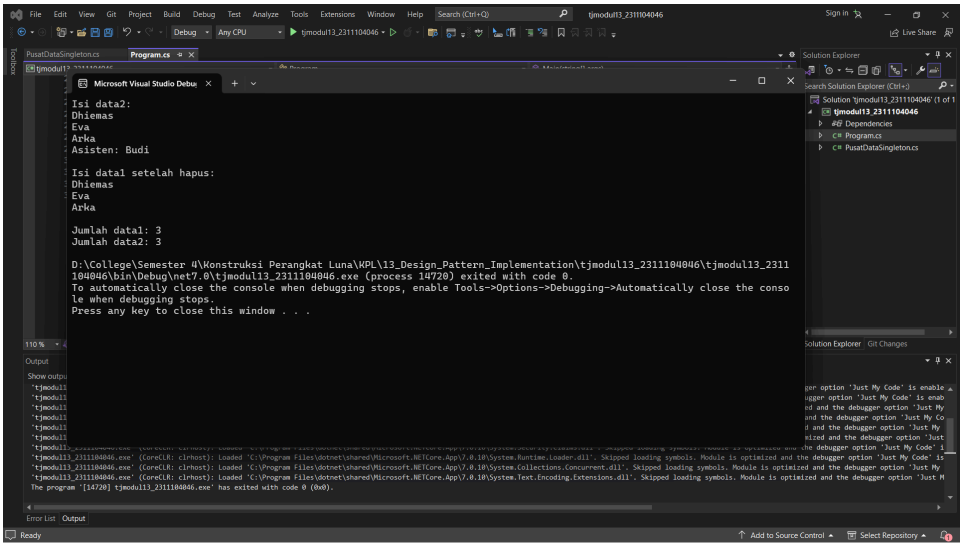
Setelah penghapusan via data2, isi data1 juga ikut berubah karena mereka adalah instance yang sama.

f. Menampilkan jumlah data

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

Menampilkan jumlah data dari data1 dan data2, yang harusnya sama karena data disimpan di satu tempat (Singleton)

4. Hasil Program



5. Kesimpulan

Berdasarkan implementasi dan pengujian pada program ini, dapat

disimpulkan bahwa Design Pattern Singleton berhasil diterapkan dengan baik dan sesuai dengan konsep dasarnya. Singleton merupakan pola desain yang memastikan bahwa sebuah kelas hanya memiliki satu instance saja selama siklus hidup program dan menyediakan titik akses global terhadap instance tersebut.

Hal ini terbukti ketika dua variabel objek (data1 dan data2) yang dipanggil dari method GetDataSingleton() tetap menunjuk ke instance yang sama. Setiap perubahan yang dilakukan pada objek data1, seperti penambahan dan penghapusan data, juga langsung terlihat pada objek data2. Ini menunjukkan bahwa tidak terjadi pembuatan instance baru, melainkan hanya satu objek yang digunakan bersama.

Implementasi ini sangat berguna dalam kasus-kasus di mana dibutuhkan kontrol global terhadap data atau objek, seperti dalam pengelolaan konfigurasi sistem, koneksi database, ataupun log sistem. Namun, penerapan Singleton juga harus dilakukan secara bijak, karena dapat menimbulkan masalah seperti kesulitan dalam pengujian unit dan pelanggaran terhadap prinsip SOLID jika tidak dikelola dengan baik.

Dengan menyelesaikan tugas ini, pemahaman terhadap prinsip kerja dan penerapan pola Singleton dalam pemrograman C# menjadi lebih kuat, terutama dalam konteks pengelolaan data terpusat dan keterbatasan jumlah instance objek yang diizinkan.