

## **JURNAL 13 KONTRUKSI PERANGKAT LUNNAK**

Nama : Dimas Abhipraya

NIM : 2311104069

Kelas : SE-07-02

### **MENJELASKAN SALAH SATU DESIGN PATTERN**

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

Jawab:

- Koneksi ke Database Dalam aplikasi besar, biasanya hanya dibutuhkan satu instance dari kelas koneksi database untuk menghindari konflik akses dan menjaga performa. Singleton memastikan semua bagian program menggunakan objek koneksi yang sama.
- Logger (Pencatat Log) Untuk mencatat aktivitas aplikasi (logging), kita bisa menggunakan satu objek logger global. Jika setiap bagian program membuat logger sendiri, maka file log bisa berantakan atau saling timpa. Singleton mencegah hal ini dengan satu titik kontrol log.

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

Jawab:

- Buat field static privat di dalam kelas untuk menyimpan instance Singleton.
- Privatkan konstruktor agar tidak bisa dibuat objek menggunakan new.
- Buat method static publik seperti getInstance() yang:
  - Mengecek apakah instance sudah dibuat.
  - Jika belum, membuat instance baru.
  - Jika sudah ada, mengembalikan instance lama. Gunakan method ini di seluruh program untuk mengakses instance, bukan membuat instance baru secara langsung.

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

Jawab:

Kelebihan:

- Menjamin hanya ada satu instance Cocok untuk objek penting yang tidak boleh diduplikasi, seperti koneksi database.
  - Memberikan akses global Semua bagian program dapat mengakses objek Singleton dari mana saja tanpa harus menyimpan referensi.
  - Inisialisasi hanya saat dibutuhkan (lazy initialization)
- Objek hanya dibuat saat pertama kali dibutuhkan, sehingga efisien dalam penggunaan memori.

Kekurangan:

- Melanggar Single Responsibility Principle Singleton menyelesaikan dua masalah sekaligus: pembatasan jumlah instance dan akses global, yang seharusnya ditangani secara terpisah.
- Sulit untuk unit testing Karena konstruktor privat dan method static sulit untuk di-mock dalam testing otomatis.
- Berisiko di lingkungan multithread Jika tidak hati-hati, beberapa thread bisa membuat instance ganda secara bersamaan.

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

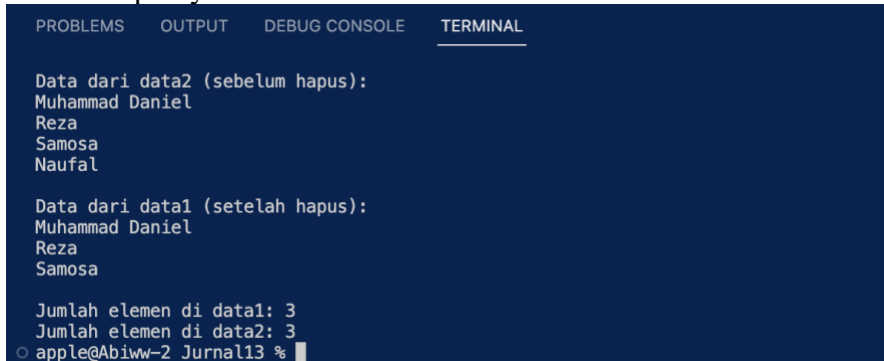
13\_Design\_Pattern\_Implementation > Jurnal13 > Jurnal13 > C# Program.cs

```
1  using System;
2
3  class Program
4  {
5      static void Main(string[] args)
6      {
7          // A & B
8          PusatDataSingleton data1 = PusatDataSingleton.GetDataSingleton();
9          PusatDataSingleton data2 = PusatDataSingleton.GetDataSingleton();
10
11         // C: Tambahkan data nama anggota dan asisten
12         data1.AddSebuahData("Muhammad Daniel");
13         data1.AddSebuahData("Reza");
14         data1.AddSebuahData("Samosa");
15         data1.AddSebuahData("Naufal"); // asisten
16
17         // D: Print semua data dari data2
18         Console.WriteLine("\nData dari data2 (sebelum hapus:");
19         data2.PrintSemuaData();
20
21         // E: Hapus nama asisten dari data2
22         data2.HapusSebuahData(3); // index ke-3 adalah nama asisten
23
24         // F: Print ulang dari data1
25         Console.WriteLine("\nData dari data1 (setelah hapus:");
26         data1.PrintSemuaData();
27
28         // G: Cetak jumlah data
29         Console.WriteLine($"Jumlah elemen di data1: {data1.GetSemuaData().Count}");
30         Console.WriteLine($"Jumlah elemen di data2: {data2.GetSemuaData().Count}");
31     }
32 }
33
```

Kode di atas merupakan implementasi design pattern Singleton pada C#. Kelas PusatDataSingleton hanya memiliki satu instance yang diakses melalui method

GetDataSingleton(). Data disimpan dalam list DataTersimpan, dengan method untuk menambah, menghapus, dan menampilkan data. Pada Main, dua variabel (data1 dan data2) mengambil instance yang sama. Data ditambahkan melalui data1, lalu ditampilkan dan dihapus melalui data2, membuktikan bahwa keduanya mengakses data yang sama. Singleton ini memastikan data tetap konsisten dan hanya ada satu instance yang digunakan di seluruh program.

### Maka Outputnya

A screenshot of a terminal window with a dark blue background. At the top, there are four tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL', with 'TERMINAL' being the active tab. The terminal displays the following text:

```
Data dari data2 (sebelum hapus):  
Muhammad Daniel  
Reza  
Samosa  
Naufal  
  
Data dari data1 (setelah hapus):  
Muhammad Daniel  
Reza  
Samosa  
  
Jumlah elemen di data1: 3  
Jumlah elemen di data2: 3  
apple@Abiww-2 Jurnal13 %
```