# JURNAL 13 KONTRUKSI PERANGKAT LUNNAK

Nama : Rifqi M Ramdani

NIM : 2311104044

Kelas : SE-07-02

Github : https://github.com/Ramdaniiiii/KPL_RifqiMRamdani_2311104044_SE0702.git

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.

Program.cs

modul13_2311104044 — PusatDataSing

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

6 references
public class PusatDataSingleton
{
    private static PusatDataSingleton _instance;
    public List<string> DataTersimpan;

    1 reference
    private PusatDataSingleton()
    {
        DataTersimpan = new List<string>();
    }

    2 references
    public static PusatDataSingleton GetDataSingleton()
    {
        if (_instance == null)
        {
            _instance = new PusatDataSingleton();
        }
        return _instance;
    }

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

    2 references
    public void PrintSemuaData()
    {
        foreach (var data in DataTersimpan)
        {
            Console.WriteLine(data);
        }
    }

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

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

0 references
class Program
{
```

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        // Inisialisasi Singleton
        var data1 = PusatDataSingleton.GetDataSingleton();
        var data2 = PusatDataSingleton.GetDataSingleton();

        // Tambah data anggota dan asprak
        data1.AddSebuahData("Ramdani");
        data1.AddSebuahData("Gideon");
        data1.AddSebuahData("Devrin");

        Console.WriteLine("Data2 sebelum penghapusan:");
        data2.PrintSemuaData(); // Menampilkan semua data

        // Hapus data asisten (misalnya hapus 'Gideon')
        data2.HapusSebuahData(1); // Gideon ada di index ke-1

        Console.WriteLine("\nData1 setelah penghapusan:");
        data1.PrintSemuaData(); // Gideon sudah tidak ada

        // Tampilkan jumlah data
        Console.WriteLine($"\nJumlah data di data1: {data1.GetSemuaData().Count}");
        Console.WriteLine($"Jumlah data di data2: {data2.GetSemuaData().Count}");
    }
}
```

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

```
Microsoft Visual Studio Debu
Data2 sebelum penghapusan:
Ramdani
Gideon
Devrin

Data1 setelah penghapusan:
Ramdani
Devrin

Jumlah data di data1: 2
Jumlah data di data2: 2

D:\KPL_RAMDANI_2311104044_SE-07-02\13_Design_Pattern_Implementation 14\modul13_2311104044\modul13_2311104044\bin\Debug\modul13_2311104044.exe (process 46220) 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 . . .
```