Nama:Faishal Arif

Setiawan

Nim:2311104066

Kelas:SE-07-02

JURNAL MODUL 13

I.Link Github

https://github.com/faishalstwn/KPL_FaishalArifSetiawan_2311104066_SE_07_02

II. MENJELASKAN DESIGN PATTERN SINGLETON

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.

III.Source Code

PusatDataSingleton.cs

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

namespace Jurnalmodul13_2311104066
{
    6 references
    public class PusatDataSingleton
    {
        private static PusatDataSingleton _instance;
        6 references
        public List<string> DataTersimpan { get; private set; }

        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);
            }
        }
    }
}
```

```
public class PusatDataSingleton
{
```

- public class: bisa diakses dari file manapun.
- Kelas ini menggunakan pola **Singleton** untuk memastikan hanya **satu instance** objek yang bisa digunakan di seluruh program.

```
private static PusatDataSingleton _instance;
6 references
```

- Menyimpan **satu-satunya instance** dari class ini.

- Bersifat static, jadi milik class dan bukan objek individual.

```
6 references
public List<string> DataTersimpan { get; private set; }
```

- List yang menyimpan data yang ditambahkan.
- get public, tapi set privat agar tidak bisa diubah langsung dari luar class.

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

- private: tidak bisa diakses dengan new dari luar.
- Membuat list saat instance pertama kali dibuat.

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

    return _instance;
}
```

- Memberikan akses global ke instance tunggal.
- Jika belum dibuat (_instance == null), akan dibuat dulu.
- Kalau sudah ada, akan mengembalikan instance yang sama.

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

Mengembalikan list berisi semua data yang sudah ditambahkan.

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

- Menampilkan semua data yang tersimpan di konsol.

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

Menambahkan data string ke list.

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

- Menghapus data berdasarkan index.
- Dicek agar tidak terjadi error jika index tidak valid.

Program.cs

```
using Jurnalmodul13_2311104066;
using System;

namespace tpmodul13_2311104066
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            var data1 = PusatDataSingleton.GetDataSingleton();
            var data2 = PusatDataSingleton.GetDataSingleton();

            // Tambahkan nama anggota kelompok dan asisten praktikum
            data1.AddSebuahData("stevan");
            data1.AddSebuahData("Stevani");
            data1.AddSebuahData("Asprak: Gideon");

            // Print data dari data2 (seharusnya sama dengan data1)
            Console.WriteLine("Data dari data2:");
            data2.PrintSemuaData();

            // Hapus data asisten dari data2
            data2.HapusSebuahData(2);

            // Print ulang dari data1 (asisten harus sudah hilang)
            Console.WriteLine("\nSetelah dihapus, data dari data1:");
            data1.PrintSemuaData();

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

```
using Jurnalmodul13_2311104066;
using System;
```

- Mengimpor namespace dari class PusatDataSingleton yang sebelumnya kamu buat.

- System diperlukan untuk fungsi dasar seperti Console.WriteLine().

```
namespace tpmodul13_2311104066
{
    0 references
    internal class Program
    {
        0 references
```

- Mendefinisikan method Main() sebagai titik masuk utama dari program.
- Namespace tpmodul13_2311104066 digunakan sesuai nama project.

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

- data1 dan data2 mengakses instance yang sama dari Singleton.
- Keduanya menunjuk ke objek yang sama, meskipun nama variabel berbeda.
-
- Menambahkan tiga data (nama anggota dan asisten) ke instance Singleton.

```
data2.PrintSemuaData();
```

Karena data1 dan data2 menunjuk ke objek yang sama, maka data yang ditambahkan lewat data1 juga muncul saat dicetak lewat data2.

```
data2.HapusSebuahData(2);
```

- Menghapus data ke-3, yaitu "Asprak: Gideon" dari list.
- Perubahan ini akan terlihat juga pada data1, karena instance-nya sama.

```
Console.WriteLine("\nSetelah dihapus, data dari data1:");
data1.PrintSemuaData();
```

- Menunjukkan bahwa data "Asprak: Gideon" juga tidak ada pada data1, membuktikan bahwa hanya satu instance digunakan bersama.

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

Menghitung dan mencetak jumlah data dari data1 dan data2

IV.Output


Data dari data2:
stevan
Stevani
Asprak: Gideon
Setelah dihapus, data dari data1:
stevan
Stevani
Jumlah data di data1: 2
Jumlah data di data2: 2
D:\KPL SMT 4\KPL_FaishalArifSetiawan_2311104066_SE_07_02\13_Clean_Code_Standard\Jurnalmodul13_2311104066\J
311104066\bin\Debug\net8.0\Jurnalmodul13_2311104066.exe (process 19784) exited with code 0 (0x0).