TUGAS JURNAL
KONSTRUKSI PERANGKAT LUNAK
MODUL 13

Disusun oleh :
Althafia Defiyandrea Laskanadya Wibowo

S1 REKAYASA PERANGKAT LUNAK
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

1. MENJELASKAN DESIGN PATTERN SINGLETON

a. Dua Contoh Kondisi Penggunaan Singleton Pattern

Jawaban : Design pattern Singleton digunakan saat kita ingin memastikan hanya ada satu instance dari suatu kelas yang dapat diakses secara global.

- Contoh 1: Logger / Sistem Logging Sebuah aplikasi besar hanya memerlukan satu objek Logger untuk mencatat semua aktivitas ke file log, agar konsisten dan efisien.
- Contoh 2: Database Connection Manager Aplikasi yang terhubung ke database sebaiknya hanya menggunakan satu instance koneksi utama agar tidak membanjiri resource server dan menghindari konflik antar koneksi.

b. Langkah-langkah Mengimplementasikan Singleton Pattern

1. Buat constructor private
    - Agar objek tidak bisa dibuat dari luar kelas.
2. Buat field static yang menyimpan satu-satunya instance

    ```
    private static Singleton _instance;
    ```

3. Buat method static GetInstance()

    Method ini akan memeriksa apakah instance sudah dibuat. Jika belum, buatlah. Jika sudah, kembalikan yang sudah ada.

    ```
    public static Singleton GetInstance()
    {
        if (_instance == null)
        {
            _instance = new Singleton();
        }
        return _instance;
    }
    ```

    (Opsional) Gunakan mekanisme thread-safe untuk menghindari race condition pada aplikasi multi-thread.

c. Kelebihan dan Kekurangan Singleton Pattern

1. Kelebihan :
    - Menghemat memori : Hanya satu instance dibuat selama aplikasi berjalan.
    - Global Access : Dapat diakses dari bagian mana pun dalam program.
    - Konsistensi data : Cocok untuk state atau konfigurasi global yang tidak boleh duplikat.
2. Kekurangan :

- Menyulitkan unit testing : Karena bergantung pada instance global, sulit mengganti dengan mock object.
- Menyebabkan hidden dependency : Kelas yang menggunakan singleton jadi tergantung pada objek global, menurunkan fleksibilitas.
- Tidak cocok untuk aplikasi paralel / multi-thread tanpa penanganan ekstra : Bisa menyebabkan bug seperti race condition jika tidak dibuat thread-safe.

IMPLEMENTASI KODE

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

public class PusatDataSingleton
{
    // Property Singleton
    private static PusatDataSingleton? _instance;

    // Atribut untuk menyimpan data
    private List<string> DataTersimpan;

    // Konstruktor private (agar tidak bisa dibuat objek dari luar)
    private PusatDataSingleton()
    {
        DataTersimpan = new List<string>();
    }

    // Method untuk mengakses instance Singleton
    public static PusatDataSingleton GetDataSingleton()
    {
        if (_instance == null)
        {
            _instance = new PusatDataSingleton();
        }
        return _instance;
    }

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

    // Menghapus data berdasarkan index
    public void HapusSebuahData(int index)
    {
        if (index >= 0 && index < DataTersimpan.Count)
        {
            DataTersimpan.RemoveAt(index);
        }
        else
        {
            Console.WriteLine("Index tidak valid!");
        }
    }
```

```
46
47      // Mengembalikan list data
48      public List<string> GetSemuaData()
49      {
50          return DataTersimpan;
51      }
52
53      // Menampilkan semua data
54      public void PrintSemuaData()
55      {
56          if (DataTersimpan.Count == 0)
57          {
58              Console.WriteLine("Tidak ada data tersimpan.");
59              return;
60          }
61
62          Console.WriteLine("Isi Data yang Tersimpan:");
63          for (int i = 0; i < DataTersimpan.Count; i++)
64          {
65              Console.WriteLine($"{i}. {DataTersimpan[i]}");
66          }
67      }
68  }
69
70  class Program
71  {
72      static void Main(string[] args)
73      {
74          var data1 = PusatDataSingleton.GetDataSingleton();
75          var data2 = PusatDataSingleton.GetDataSingleton();
76
77          data1.AddSebuahData("Althafia Defiyandrea");
78          data1.AddSebuahData("Maria Nathasya");
79          data1.AddSebuahData("Lintang Suminar");
80
81          Console.WriteLine("\nPrint dari data-2:");
82          data2.PrintSemuaData();
83
84          Console.WriteLine("\nMenghapus Sesuatu Entitas Hitam");
85          data2.HapusSebuahData(data2.GetSemuaData().IndexOf("Sesuatu Entitas Hitam"));
86
87          Console.WriteLine("\nPrint dari data-1 setelah penghapusan:");
88          data1.PrintSemuaData();
89
90          Console.WriteLine($"\nJumlah data di data-1: {data1.GetSemuaData().Count}");
91          Console.WriteLine($"Jumlah data di data-2: {data2.GetSemuaData().Count}");
92      }
93  }
```