TUGAS JURNAL
KONSTRUKSI PERANGKAT LUNAK

MODUL XIII

DESIGN PATTERN IMPLEMENTATION

Disusun Oleh :

Atika Aji Hadiyani

2211104003

SE0601

Dosen Pengampu :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

TELKOM UNIVERSITY PURWOKERTO

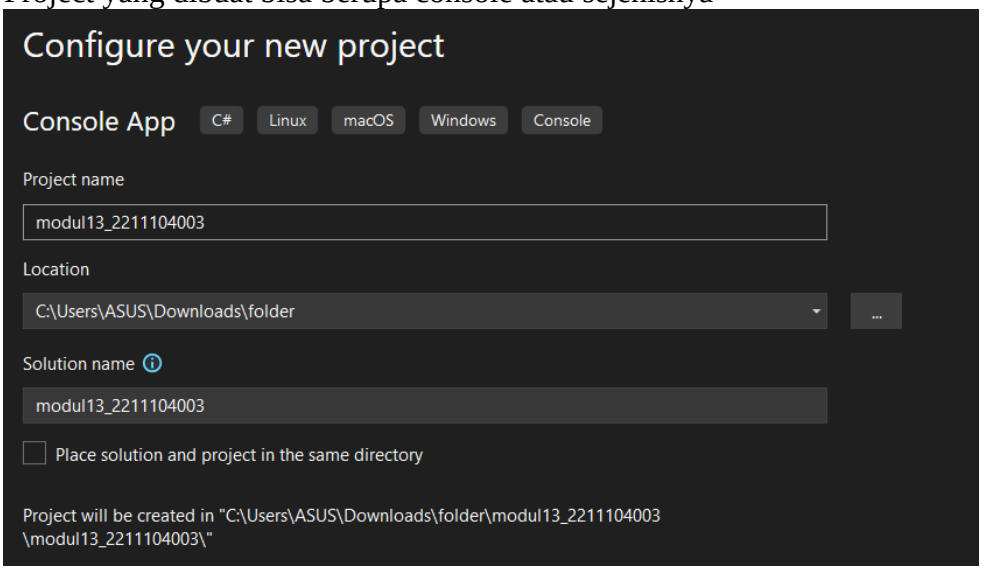
TUGAS JURNAL 13

1. MEMBUAT PROJECT GUI BARU

Buka IDE misalnya dengan Visual Studio

A. Misalnya menggunakan Visual Studio, buatlah project baru dengan nama modul113_NIM

B. Project yang dibuat bisa berupa console atau sejenisnya



2. MENJELASKAN SALAH SATU DESIGN PATTERN

Buka halaman web https://refactoring.guru/design-patterns/catalog kemudian baca design pattern dengan nama “Observer”, dan jawab pertanyaan berikut ini (dalam Bahasa Indonesia):

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

Jawab:

- Pengelolaan Koneksi ke Database

  Singleton digunakan untuk memastikan hanya ada satu objek koneksi yang aktif ke database selama program berjalan, sehingga menghindari overhead dari pembukaan koneksi berulang kali.

- Manajemen Konfigurasi Aplikasi

  Ketika sebuah aplikasi memiliki konfigurasi global (seperti pengaturan tema, bahasa, atau koneksi jaringan), Singleton dapat digunakan untuk menyimpan dan mengakses konfigurasi tersebut dari berbagai bagian aplikasi tanpa membuat duplikasi.

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

Jawab:

1. Buat constructor kelas bersifat private, agar objek tidak bisa dibuat dari luar kelas.
2. Tambahkan atribut statik privat di dalam kelas untuk menyimpan instance tunggal dari kelas tersebut.
3. Buat metode statik publik (misalnya GetInstance() atau Instance()) yang akan mengembalikan instance. Di dalam metode ini, jika instance belum dibuat, maka dibuatlah; jika sudah ada, kembalikan instance yang sama.
4. Pastikan akses ke instance bersifat global melalui metode tersebut.

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

Jawab:

Kelebihan

- **Menghemat resource**, karena hanya satu instance yang dibuat dan digunakan ulang.
- **Konsistensi data**, karena semua bagian program mengakses objek yang sama.
- **Kontrol akses terpusat**, memudahkan pengelolaan state dan logika bersama.

Kekurangan:

- Menyulitkan pengujian unit (unit testing) karena sulit membuat mock instance.
- Membuat tight coupling**,** karena banyak bagian kode bergantung pada instance tunggal tersebut.
- Tidak cocok untuk lingkungan multithread tanpa sinkronisasi**,** karena bisa menyebabkan race condition jika tidak diatur dengan benar.

3. IMPLEMENTASI DAN PEMAHAMAN DESIGN PATTERN SINGLETON

Buka halaman web berikut https://refactoring.guru/design-patterns/observer dan scroll ke bagian “Code Examples”, pilih kode yang akan dilihat misalnya C# dan ikuti langkah-langkah berikut:

A. Dengan contoh yang sudah diberikan, buatlah sebuah class dengan design pattern singleton dengan nama “PusatDataSingleton”.
B. 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.
C. Class tersebut juga memiliki beberapa method yaitu:
    - Konstruktor dari kelas tersebut yang mengisi atribut “DataTersimpan” dengan list kosong.
    - GetDataSingleton() yang mengembalikan “_instance” jika tidak null dan memanggil konstruktor terlebih dahulu apabila nilainya masih null.
    - GetSemuaData() yang mengembalikan list dari property “DataTersimpan”.
    - PrintSemuaData() yang melakukan print satu per satu dari string yang ada di list “DataTersimpan”.
    - AddSebuahData(string input) yang menambahkan satu data baru “input” ke dalam list “DataTersimpan”.
    - HapusSebuahData(int index) yang menghapus sebuah data berdasarkan index tertentu.

4. IMPLEMENTASI PROGRAM UTAMA

Tambahkan beberapa implementasi di program/method utama atau “main”:

A. Buatlah dua variable dengan tipe “PusatDataSingleton” bernama data1 dan data2.
B. Isi kedua variable tersebut dengan hasil keluaran dari GetDataSingleton().
C. Pada data1 lakukan pemanggilan method AddSebuahData() beberapa kali dengan input nama anggota kelompok dan asisten praktikum.

D. Pada data2 panggil method PrintSemuaData(), pastikan keluaran dari hasil print data2 menampilkan nama-nama anggota kelompok dan asisten praktikum.
E. Pada data2 panggil HapusSebuahData() untuk menghapus nama asisten praktikum anda sekarang.
F. Pada data1 panggil PrintSemuaData(), dan seharusnya nama asisten praktikum anda tidak muncul di hasil print tersebut.
G. Langkah terakhir, pada data1 dan data2 panggil GetSemuaData() dan lakukan print dari jumlah “Count” atau elemen yang ada di list pada data1 dan data2.

Source Code:

- File PusatDataSingleton.cs

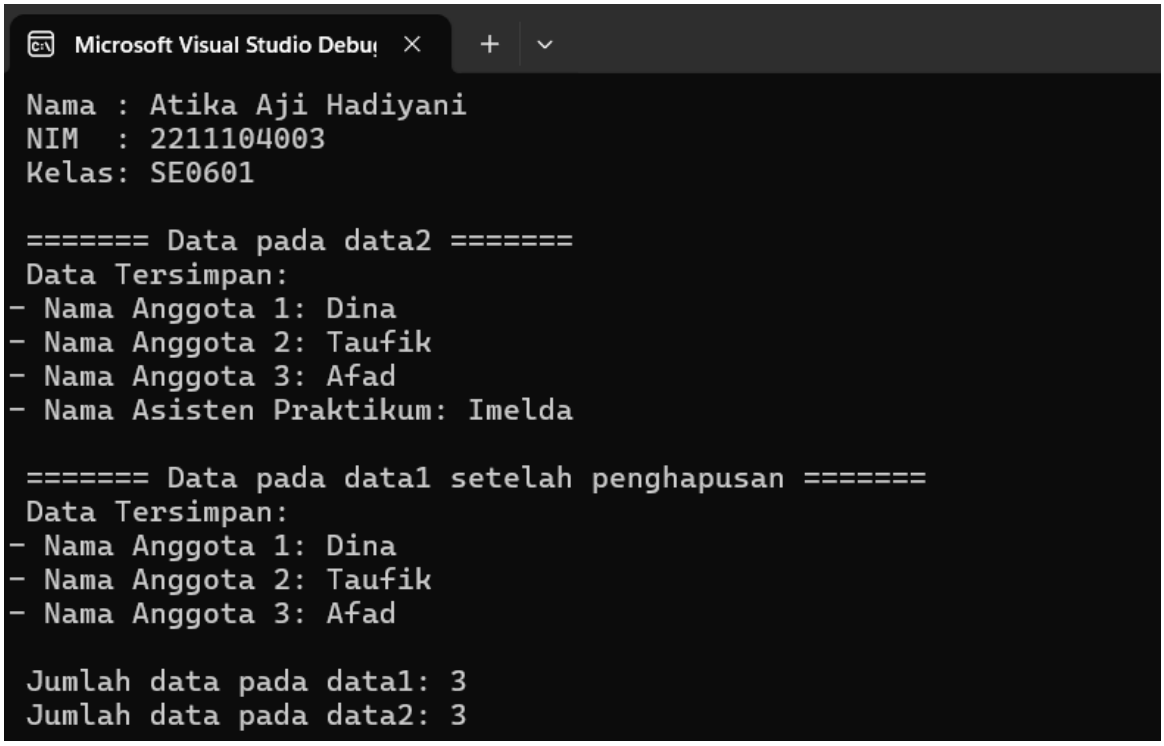
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace modul13_2211104003
{
    8 references
    public class PusatDataSingleton
    {
        // Atribut Singleton dan List
        private static PusatDataSingleton _instance;
        private List<string> DataTersimpan;

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

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

        // Method untuk mendapatkan semua data
        2 references
        public List<string> GetSemuaData()
        {
            return DataTersimpan;
        }

        // Method untuk mencetak semua data
        2 references
        public void PrintSemuaData()
        {
            Console.WriteLine(" Data Tersimpan:");
            foreach (string data in DataTersimpan)
            {
                Console.WriteLine("- " + data);
            }
        }
```

```
        // Method untuk menambahkan data
        4 references
        public void AddSebuahData(string input)
        {
            DataTersimpan.Add(input);
        }

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

- File Program.cs

```
using modul13_2211104003;
using System;

namespace modul13_2211104003
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine(" Nama : Atika Aji Hadiyani ");
            Console.WriteLine(" NIM  : 2211104003 ");
            Console.WriteLine(" Kelas: SE0601 ");

            // Membuat dua variable Singleton
            PusatDataSingleton data1 = PusatDataSingleton.GetDataSingleton();
            PusatDataSingleton data2 = PusatDataSingleton.GetDataSingleton();

            // Menambahkan data (nama anggota kelompok dan asisten)
            data1.AddSebuahData("Nama Anggota 1: Dina");
            data1.AddSebuahData("Nama Anggota 2: Taufik");
            data1.AddSebuahData("Nama Anggota 3: Afad");
            data1.AddSebuahData("Nama Asisten Praktikum: Imelda");

            // Mencetak semua data melalui data2 (harusnya sama dengan data1)
            Console.WriteLine("\n ======= Data pada data2 ======= ");
            data2.PrintSemuaData();

            // Menghapus nama asisten praktikum
            data2.HapusSebuahData(3); // Hapus asisten praktikum di index ke-3

            // Mencetak kembali data melalui data1 (asisten harus sudah terhapus)
            Console.WriteLine("\n ======= Data pada data1 setelah penghapusan ======= ");
            data1.PrintSemuaData();

            // Mencetak jumlah data pada kedua variabel
            Console.WriteLine("\n Jumlah data pada data1: " + data1.GetSemuaData().Count);
            Console.WriteLine(" Jumlah data pada data2: " + data2.GetSemuaData().Count);
        }
    }
}
```

Output:



Penjelasan

Dalam file PusatDataSingleton.cs, terdapat sebuah kelas bernama PusatDataSingleton yang mengimplementasikan *design pattern* Singleton. Tujuan utama dari pola ini adalah untuk menjamin bahwa hanya satu objek dari kelas tersebut yang akan dibuat dan digunakan di seluruh aplikasi. Kelas ini memiliki atribut DataTersimpan, yaitu sebuah daftar string yang berfungsi sebagai tempat penyimpanan data, serta atribut privat _instance yang menampung satu-satunya instance dari kelas ini. Metode GetDataSingleton() bertugas untuk menginisialisasi instance jika belum ada, sehingga tidak terjadi duplikasi objek. Selain itu, tersedia pula metode-metode seperti AddSebuahData, HapusSebuahData, dan PrintSemuaData, yang masing-masing digunakan untuk menambah, menghapus, dan menampilkan isi daftar data.

Sedangkan dalam file Program.cs, dilakukan pengujian terhadap penerapan pola Singleton. Dua variabel, yaitu data1 dan data2, dibuat untuk mereferensikan instance dari kelas PusatDataSingleton. Meski tampaknya berbeda, kedua variabel tersebut sebenarnya menunjuk pada objek yang sama. Saat data ditambahkan melalui data1, hasilnya juga dapat diakses melalui data2, dan sebaliknya. Bahkan, ketika data dihapus melalui data2, perubahan tersebut turut terlihat pada data1. Hal ini menunjukkan bahwa keduanya memang menggunakan satu instance yang sama. Program kemudian mencetak jumlah data dari kedua variabel tersebut, yang hasilnya selalu identik karena mereka berbagi instance yang sama.