# TUGAS JURNAL
# KONSTRUKSI PERANGKAT LUNAK

## MODUL XIII

## DESIGN PATTERN IMPLEMENTATION


**Disusun Oleh :**

Nadia Putri Rahmaniar / 2211104012

S1 SE-06-01

**Dosen Pengampu :**

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



**PROGRAM STUDI S1 SOFTWARE ENGINEERING**

**TELKOM UNIVERSITY PURWOKERTO**

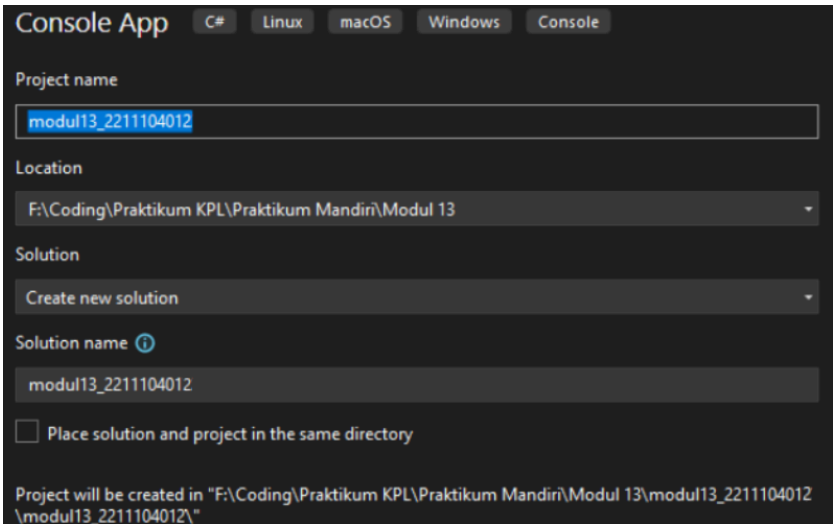
# TUGAS JURNAL 13

## 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?

- Pengelolaan Sumber Daya Tunggal (Single Resource Management)

  Digunakan ketika aplikasi memerlukan satu objek sentral untuk mengelola sumber daya eksklusif, seperti printer spooler atau thread pool. Dengan hanya satu instance Singleton yang mengontrol akses ke sumber daya ini, kita dapat mencegah konflik dan memastikan koordinasi yang efisien dalam penggunaannya oleh berbagai bagian aplikasi.

- Manajemen Cache Aplikasi (Application Cache Management)

  Sangat cocok untuk implementasi sistem caching di mana hanya satu instance manajer cache yang diperlukan untuk seluruh aplikasi. Manajer cache ini akan bertanggung jawab untuk menyimpan data yang sering diakses dan memastikan konsistensi data serta efisiensi penggunaan memori secara global, sehingga semua komponen aplikasi dapat mengakses data yang sama dari cache terpusat.

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

- Buat Konstruktor Privat: Langkah pertama adalah membuat konstruktor kelas menjadi privat. Ini adalah kunci utama untuk mencegah kelas di-instantiate (dibuat objeknya) dari luar kelas itu sendiri. Jika konstruktornya tidak privat, siapa pun bisa membuat instance baru, dan tujuan singleton pun gagal.
- Deklarasikan Atribut Statis untuk Instance: Anda perlu mendeklarasikan atribut statis (terkadang disebut juga variabel kelas) di dalam kelas Singleton. Atribut ini akan bertugas untuk menyimpan satu-satunya instance dari kelas tersebut begitu dibuat. Karena statis, atribut ini akan dibagikan oleh semua instance (dalam hal ini, hanya satu) dan tidak terikat pada objek tertentu.
- Sediakan Metode Statis untuk Mendapatkan Instance: Buatlah metode statis publik (sering dinamakan getInstance()) yang akan menjadi satu-satunya cara untuk mengakses instance Singleton. Metode ini berisi logika untuk memeriksa apakah instance sudah ada dalam atribut statis. Jika belum, metode ini akan membuat instance baru dan menyimpannya di atribut statis tersebut, lalu mengembalikannya. Jika instance sudah ada, metode ini hanya akan mengembalikan instance yang sudah ada.
- Opsional: Cegah Pewarisan Kelas: Untuk bahasa pemrograman tertentu (seperti Java dengan final atau C# dengan sealed), Anda bisa mencegah kelas Singleton diwarisi oleh kelas lain. Ini memastikan bahwa tidak ada kelas turunan yang bisa mengubah atau memecah fungsionalitas singleton, menjaga integritas pola tersebut.

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

**Kelebihan**

- Penghematan Sumber Daya: Salah satu keuntungan utama adalah efisiensi memori. Karena Singleton hanya menjamin satu instance dari sebuah kelas, alokasi memori menjadi lebih optimal, menghindari duplikasi objek yang tidak perlu.
- Akses Global yang Disederhanakan: Memberikan titik akses global yang mudah ke instance tunggal. Ini menghilangkan kebutuhan untuk meneruskan objek antar komponen, menyederhanakan arsitektur dan interaksi antarmuka dalam aplikasi.
- Integritas Data Terjamin: Dengan hanya satu instance yang beroperasi, konsistensi data yang dikelolanya menjadi lebih terjamin. Ini meminimalisir risiko inkonsistensi atau konflik data yang bisa terjadi jika ada banyak instance yang mencoba memodifikasi data yang sama secara bersamaan.

**Kekurangan**:

- Tantangan dalam Pengujian Unit: Salah satu kerugian terbesar adalah kesulitan dalam pengujian (testing). Karena sifatnya yang global, instance Singleton sulit diisolasi dan diatur ulang antara test case, yang dapat menyulitkan penulisan unit test yang bersih dan independen.
- Potensi Menjadi Hambatan Kinerja: Jika Singleton digunakan untuk operasi yang sering diakses atau memerlukan banyak sumber daya, ia berpotensi menjadi bottleneck kinerja. Semua permintaan harus melalui satu instance ini, yang bisa memperlambat aplikasi jika instance tersebut kewalahan.
- Fleksibilitas yang Terbatas: Pola ini kurang fleksibel untuk skenario di mana aplikasi mungkin membutuhkan beberapa instance dari kelas yang sama di kemudian hari. Menerapkan Singleton berarti mengunci desain pada satu instance saja, membuat perubahan di masa depan menjadi lebih sulit.

## 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.

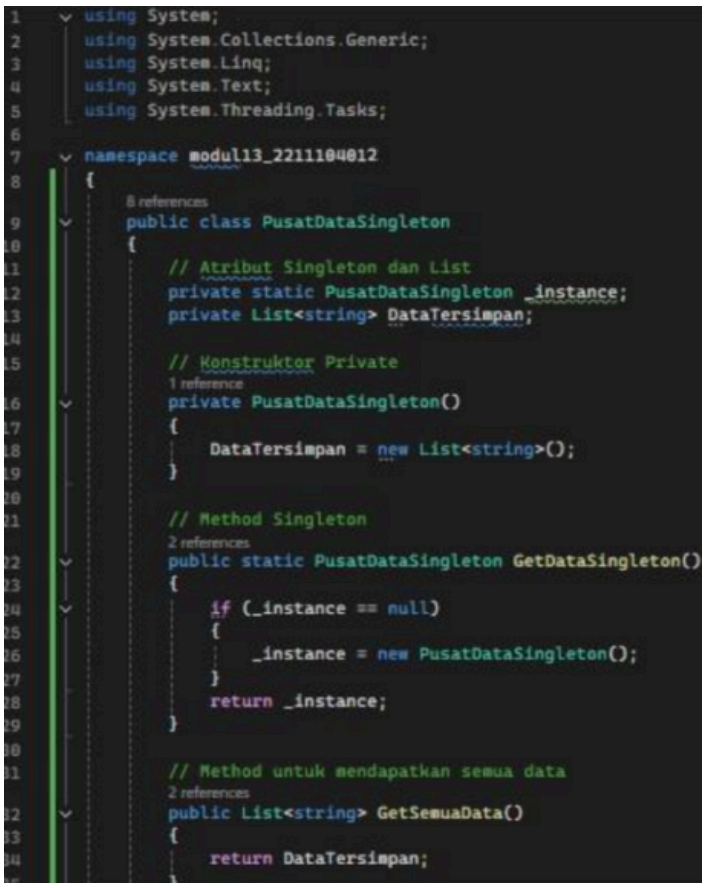
E. Pada data2 panggil HapusSebuahData() untuk menghapus nama asisten praktikum anda sekarang.

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_2211104012
{
    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 System;

namespace modul13_2211104012
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("--- Nama: Nadia Putri Rahmaniar ---");
            Console.WriteLine("--- NIM: 2211104012 ---");
            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: Nita");
            data1.AddSebuahData("Nama Anggota 2: Alfian");
            data1.AddSebuahData("Nama Anggota 3: Edgar");
            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);
        }
    }
```

**Hasil:**

```
Nama : Nadia Putri Rahmaniar
NIM  : 2211104012
Kelas: SE0601
========== Data pada data2 ==========
Data Tersimpan:
- Nama Anggota 1: Nita
- Nama Anggota 2: Alfian
- Nama Anggota 3: Edgar
- Nama Asisten Praktikum: Imelda

========== Data pada data1 setelah penghapusan ==========
Data Tersimpan:
- Nama Anggota 1: Nita
- Nama Anggota 2: Alfian
- Nama Anggota 3: Edgar

Jumlah data pada data1: 3
Jumlah data pada data2: 3
```

**Penjelasan**

Di dalam file PusatDataSingleton.cs, kita punya kelas bernama PusatDataSingleton yang didesain menggunakan pola Singleton. Inti dari pola ini adalah memastikan hanya ada satu objek (atau instance) dari kelas ini yang bisa dibuat dan digunakan di seluruh aplikasi.Kelas ini memiliki beberapa komponen utama:

- DataTersimpan: Ini adalah sebuah list bertipe string yang berfungsi untuk menyimpan semua data yang kita masukkan.
- _instance: Ini adalah variabel khusus yang akan menyimpan satu-satunya instance dari PusatDataSingleton setelah dibuat.
- GetDataSingleton(): Metode ini sangat penting. Fungsinya adalah memastikan instance PusatDataSingleton hanya dibuat satu kali saja. Setiap kali kita memanggilnya, ia akan mengembalikan instance yang sama.
- Metode Lainnya: Ada juga metode seperti AddSebuahData(), HapusSebuahData(), dan PrintSemuaData(). Sesuai namanya, metode-metode ini bertanggung jawab untuk menambah, menghapus, dan menampilkan data dari list DataTersimpan tersebut.

File Program.cs dibuat khusus untuk menguji dan mendemonstrasikan cara kerja pola Singleton. Dalam file ini, kita melakukan beberapa hal:

- Membuat Dua Variabel, Satu Instance: Kita membuat dua variabel bernama data1 dan data2. Meskipun namanya berbeda, keduanya sebenarnya menunjuk ke

instance PusatDataSingleton yang sama. Ini adalah bukti utama dari pola Singleton.

- Perubahan Data Terlihat di Kedua Sisi: Kita menambahkan data menggunakan variabel data1. Kemudian, ketika kita memeriksa data melalui data2, data yang sama sudah terlihat. Ini menunjukkan bahwa perubahan yang dilakukan melalui satu variabel akan tercermin pada variabel lain karena keduanya mengakses objek yang sama.
- Konsistensi Saat Penghapusan: Lebih lanjut, ketika kita menghapus data melalui data2, perubahan ini juga langsung terlihat saat kita memeriksa data melalui data1. Hal ini semakin memperkuat fakta bahwa data1 dan data2 adalah satu instance yang sama.
- Konfirmasi Jumlah Data: Di akhir program, kita mencetak jumlah total data (atau "Count") dari kedua variabel. Hasilnya selalu sama, yang menjadi konfirmasi terakhir bahwa kedua variabel tersebut adalah referensi untuk satu-satunya instance PusatDataSingleton yang ada.