

TUGAS JURNAL
KONSTRUKSI PERANGKAT LUNAK

MODUL XIII
DESIGN PATTERN IMPLEMENTATION



Disusun Oleh:

Aorinka Anendya Chazanah / 2211104013

S1 SE-06-01

Dosen Pengampu:

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

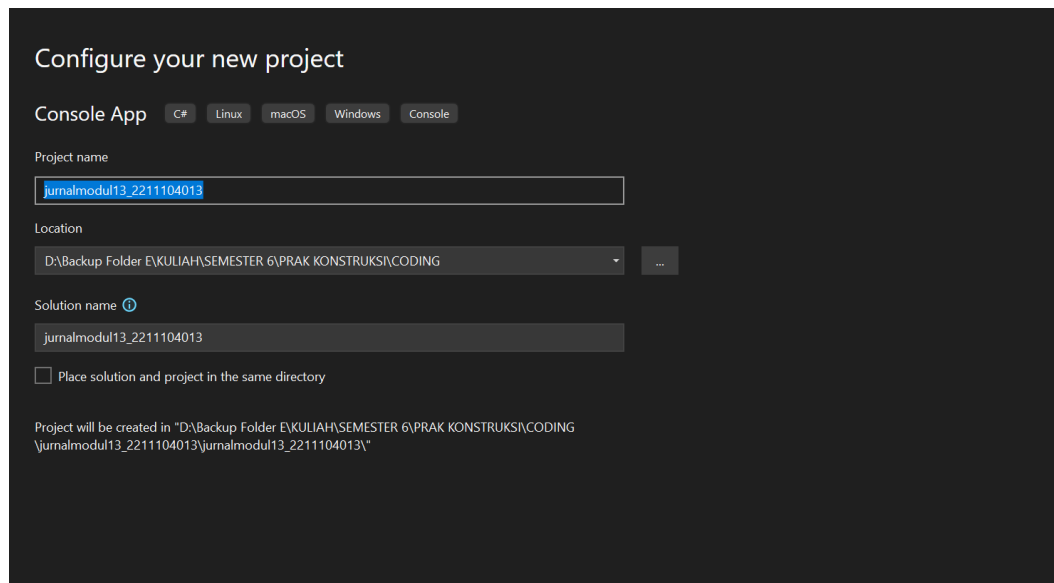
2025

TUGAS JURNAL

1. MEMBUAT PROJECT GUI BARU

Buka IDE misalnya dengan Visual Studio

- A. Misalnya menggunakan Visual Studio, buatlah project baru dengan nama modul13_NIM
- B. Project yang dibuat bisa berupa console atau sejenisnya



2. MENJELASKAN DESIGN PATTERN SINGLETON

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

- A. Berikan salah dua contoh kondisi dimana design pattern “Singleton” dapat digunakan.

- a. **Koneksi ke Database**

Singleton sangat berguna ketika aplikasi hanya membutuhkan satu koneksi database yang digunakan oleh berbagai bagian sistem. Dengan menggunakan Singleton, kita memastikan bahwa semua bagian aplikasi mengakses objek koneksi yang sama, menghindari overhead dari membuat koneksi baru berulang kali.

- b. **Logging Sistem**

Dalam sistem logging, kita biasanya hanya memerlukan satu instance logger yang digunakan oleh seluruh aplikasi. Singleton memastikan bahwa semua log yang dihasilkan akan ditangani oleh satu objek yang sama sehingga pencatatan lebih terstruktur dan konsisten.

- B. Berikan penjelasan singkat mengenai langkah-langkah dalam mengimplementasikan design pattern “Singleton”.
1. Buat konstruktor kelas menjadi private agar objek tidak bisa dibuat secara langsung dari luar kelas.
 2. Tambahkan field static privat di dalam kelas untuk menyimpan instance Singleton.
 3. Buat metode static publik (misalnya getInstance()) yang akan mengembalikan instance yang sama setiap kali dipanggil. Di dalam metode ini, buat objek hanya jika belum ada (lazy initialization).
 4. Ganti semua pemanggilan konstruktor di kode klien dengan pemanggilan metode getInstance() agar semua bagian aplikasi menggunakan instance yang sama.
- C. Berikan tiga kelebihan dan kekurangan dari design pattern “Singleton”.

Kelebihan:

1. Menjamin hanya ada satu instance dari kelas yang dibuat selama siklus hidup aplikasi.
2. Memberikan titik akses global yang konsisten terhadap instance tersebut.
3. Menghemat resource karena instance hanya dibuat saat dibutuhkan (lazy initialization).

Kekurangan:

1. Melanggar *Single Responsibility Principle* karena menangani kontrol instance sekaligus menyediakan akses global.
2. Sulit untuk diuji secara unit test karena konstruktor privat dan metode statis tidak mudah di-*mock*.
3. Rentan menyebabkan desain buruk jika digunakan secara sembarangan karena membuat bagian-bagian program terlalu bergantung pada satu objek global.

3. IMPLEMENTASI DAN PEMAHAMAN DESIGN PATTERN SINGLETON

Buka halaman web berikut <https://refactoring.guru/design-patterns/singleton> 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:

- i. Kontruktor dari kelas tersebut yang mengisi atribut “DataTersimpan” dengan list kosong.
- ii. GetDataSingleton() yang mengembalikan “_instance” jika tidak null dan memanggil konstruktor terlebih dahulu apabila nilainya masih null.
- iii. GetSemuaData() yang mengembalikan list dari property “DataTersimpan”.

Jawaban

File source code PusatDataSingleton.cs

```
1  using System;
2  using System.Collections.Generic;
3
4  public class PusatDataSingleton
5  {
6      private static PusatDataSingleton _instance;
7      public List<string> DataTersimpan;
8
9      // Konstruktor private agar tidak bisa dibuat instance langsung dari luar class
10     private PusatDataSingleton()
11     {
12         DataTersimpan = new List<string>();
13     }
14
15     // Method untuk mendapatkan instance singleton
16     public static PusatDataSingleton GetDataSingleton()
17     {
18         if (_instance == null)
19         {
20             _instance = new PusatDataSingleton();
21         }
22         return _instance;
23     }
24
25     // Mengembalikan semua data yang tersimpan
26     public List<string> GetSemuaData()
27     {
28         return DataTersimpan;
29     }
30
31     // Menampilkan semua data satu per satu
```

```

31 // Menampilkan semua data satu per satu
32 2 references
33 public void PrintSemuaData()
34 {
35     foreach (var data in DataTersimpan)
36     {
37         Console.WriteLine(data);
38     }
39
40 // Menambahkan sebuah data
41 5 references
42 public void AddSebuahData(string input)
43 {
44     DataTersimpan.Add(input);
45
46 // Menghapus data berdasarkan index
47 1 reference
48 public void HapusSebuahData(int index)
49 {
50     if (index >= 0 && index < DataTersimpan.Count)
51     {
52         DataTersimpan.RemoveAt(index);
53     }
54 }
55

```

Penjelasan kode program

Kode di atas merupakan implementasi dari design pattern Singleton dalam bahasa C#. Kelas PusatDataSingleton dirancang agar hanya memiliki satu instance selama aplikasi berjalan. Ini dicapai dengan membuat konstruktor private, sehingga objek tidak bisa dibuat langsung dari luar kelas, dan menyediakan metode GetDataSingleton() sebagai satu-satunya cara untuk mendapatkan instance tersebut. Dalam metode ini, objek hanya akan dibuat satu kali saat pertama kali diminta, dan selanjutnya akan mengembalikan instance yang sama (*lazy initialization*).

Kelas ini juga memiliki sebuah properti DataTersimpan, yaitu daftar string yang berfungsi sebagai penyimpanan data. Beberapa metode disediakan untuk mengelola data tersebut, seperti AddSebuahData() untuk menambahkan data, HapusSebuahData() untuk menghapus data berdasarkan indeks, GetSemuaData() untuk mengembalikan seluruh data, dan PrintSemuaData() untuk mencetak semua data ke konsol. Dengan pola Singleton, semua bagian program yang memanggil GetDataSingleton() akan mengakses objek dan data yang sama, sehingga data yang tersimpan tetap konsisten di seluruh aplikasi.

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.

Jawaban

File source code Program.cs

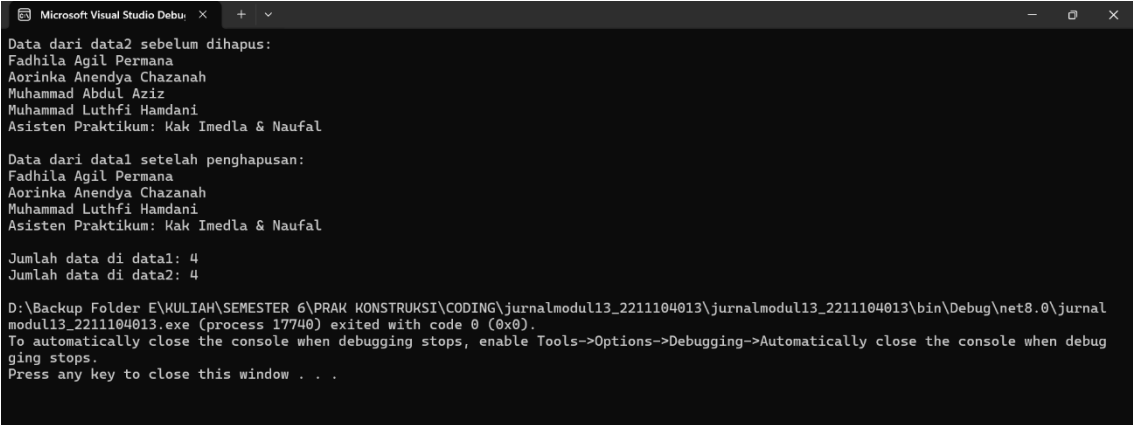
```
1  ic class Program
2
3  public static void Main(string[] args)
4  {
5      // A. Buat dua variable dengan tipe PusatDataSingleton
6      PusatDataSingleton data1 = PusatDataSingleton.GetDataSingleton();
7      PusatDataSingleton data2 = PusatDataSingleton.GetDataSingleton();
8
9      // B. Tambah data pada data1
10     data1.AddSebuahData("Fadhila Agil Permana");
11     data1.AddSebuahData("Aorinka Anendya Chazanah");
12     data1.AddSebuahData("Muhammad Abdul Aziz");
13     data1.AddSebuahData("Muhammad Luthfi Hamdani");
14     data1.AddSebuahData("Asisten Praktikum: Kak Imedla & Naufal");
15
16     // C. Print semua data dari data2
17     Console.WriteLine("Data dari data2 sebelum dihapus:");
18     data2.PrintSemuaData();
19
20     // D. Hapus data asisten praktikum (index ke-2)
21     data2.HapusSebuahData(2);
22
23     // E. Print data lagi dari data1
24     Console.WriteLine("\nData dari data1 setelah penghapusan:");
25     data1.PrintSemuaData();
26
27     // F. Print jumlah elemen di data1 dan data2
28     Console.WriteLine($"Jumlah data di data1: {data1.GetSemuaData().Count}");
29     Console.WriteLine($"Jumlah data di data2: {data2.GetSemuaData().Count}");
30 }
31
32
```

Penjelasan kode program

Program di atas mendemonstrasikan penggunaan design pattern Singleton melalui kelas PusatDataSingleton. Pada bagian awal, dua variabel data1 dan data2 dibuat menggunakan metode GetDataSingleton(). Karena menggunakan pola Singleton, kedua variabel tersebut sebenarnya menunjuk ke instance yang sama, sehingga perubahan pada salah satu variabel akan langsung berdampak pada yang lain. Kemudian, beberapa data ditambahkan melalui data1, dan data tersebut langsung bisa diakses serta ditampilkan melalui data2.

Selanjutnya, program menghapus data pada indeks ke-2 dari data2, lalu mencetak ulang isi data melalui data1. Hasilnya menunjukkan bahwa penghapusan dari data2 juga mempengaruhi isi data1, karena mereka berbagi instance yang sama. Terakhir, program mencetak jumlah data dari data1 dan data2, yang nilainya sama, memperkuat bahwa hanya satu objek yang digunakan.

Output program



```
Microsoft Visual Studio Debu: X + v
Data dari data2 sebelum dihapus:
Fadhila Agil Permana
Aorinka Anendya Chazanah
Muhammad Abdul Aziz
Muhammad Luthfi Hamdani
Asisten Praktikum: Kak Imedla & Naufal

Data dari data1 setelah penghapusan:
Fadhila Agil Permana
Aorinka Anendya Chazanah
Muhammad Luthfi Hamdani
Asisten Praktikum: Kak Imedla & Naufal

Jumlah data di data1: 4
Jumlah data di data2: 4

D:\Backup Folder E\KULIAH\SEMESTER 6\PRAK KONSTRUKSI\CODING\jurnalmodul13_2211104013\jurnalmodul13_2211104013\bin\Debug\net8.0\jurnalmodul13_2211104013.exe (process 17740) 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 . . .
```