

Jurnal 14

Nama:Fahmi Hasan Asagaf

Nim:2311104074

Link repo:

[https://github.com/fahmihasanasagaf/KPL\\_2311104074\\_SE0702/tree/main/14\\_Clean\\_Code](https://github.com/fahmihasanasagaf/KPL_2311104074_SE0702/tree/main/14_Clean_Code)

### 1. Naming Convention (Aturan Penamaan)

Tujuannya: nama variabel, method, dan atribut harus rapi dan sesuai standar bahasa pemrograman (C#) agar kode mudah dibaca oleh orang lain.

| Jenis Penamaan | Contoh SALAH       | Contoh BENAR       | Penjelasan   |
|----------------|--------------------|--------------------|--|
| Variabel       | dataTersimpan      | _dataTersimpan     | Untuk variabel private di C#, tambahkan underscore di depan.         |
| Method         | getDataSingleton() | GetDataSingleton() | Nama method pakai PascalCase, yaitu huruf kapital di awal tiap kata. |
| Property       | instance           | Instance           | Gunakan huruf kapital awal (PascalCase) untuk properti publik.       |

## 2. White Space dan Indentation (Spasi dan Rapi Kode)

Tujuannya: agar kode **tidak berantakan** dan **mudah dibaca**.

### Sebelum Refactor (Salah):

```
public void PrintSemuaData(){  
foreach(string data in _dataTersimpan){  
Console.WriteLine(data);}}
```

### Sesudah Refactor (Benar):

```
public void PrintSemuaData()  
{  
    foreach (string data in _dataTersimpan)  
    {  
        Console.WriteLine(data);  
    }  
}
```

Gunakan:

- **4 spasi** untuk indentasi dalam { ... }
- Spasi di antara keyword dan tanda kurung: if (...) bukan if(...)

### 3. Variable / Attribute Declarations (Deklarasi yang Jelas)

Tujuannya: menuliskan variabel **dengan tipe data dan akses yang jelas**.

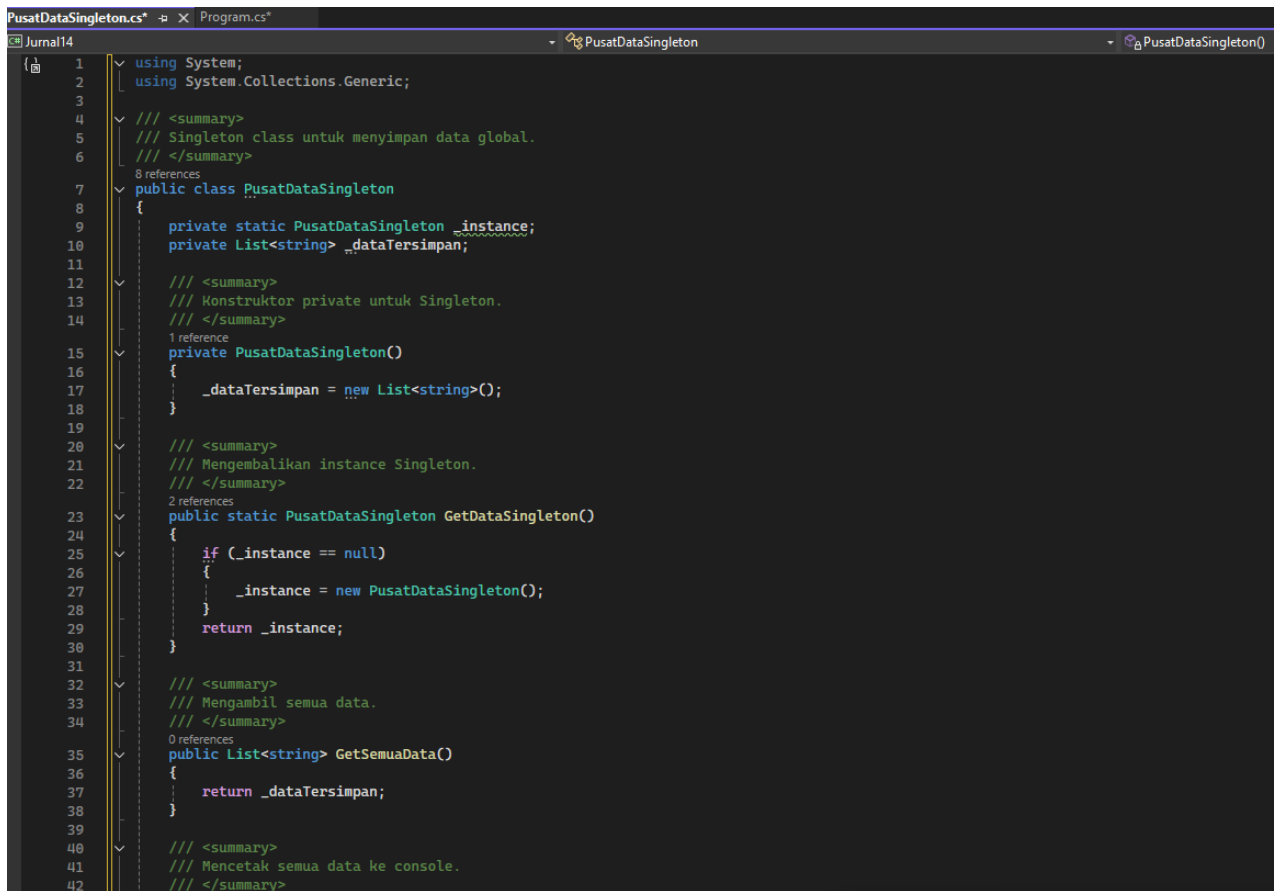
#### Contoh Sebelum Refactor (Kurang Jelas):

```
var list = new List<string>();
```

#### Contoh Sesudah Refactor (Lebih Jelas):

```
private List<string> _dataTersimpan = new List<string>();
```

code jurnal 13 yang sudah di refactoring



```
PusatDataSingleton.cs* X Program.cs*
Jurnal14 - PusatDataSingleton - PusatDataSingleton()
1 using System;
2 using System.Collections.Generic;
3
4 /// <summary>
5 /// Singleton class untuk menyimpan data global.
6 /// </summary>
7 public class PusatDataSingleton
8 {
9     private static PusatDataSingleton _instance;
10    private List<string> _dataTersimpan;
11
12    /// <summary>
13    /// Konstruktor private untuk Singleton.
14    /// </summary>
15    private PusatDataSingleton()
16    {
17        _dataTersimpan = new List<string>();
18    }
19
20    /// <summary>
21    /// Mengembalikan instance Singleton.
22    /// </summary>
23    public static PusatDataSingleton GetDataSingleton()
24    {
25        if (_instance == null)
26        {
27            _instance = new PusatDataSingleton();
28        }
29        return _instance;
30    }
31
32    /// <summary>
33    /// Mengambil semua data.
34    /// </summary>
35    public List<string> GetSemuaData()
36    {
37        return _dataTersimpan;
38    }
39
40    /// <summary>
41    /// Mencetak semua data ke console.
42    /// </summary>
```

## Program.cs

```
PusatDataSingleton.cs*  Program.cs* X
Jurnal14  Program
1  using System;
2
3  0 references
4  class Program
5  {
6      0 references
7      static void Main(string[] args)
8      {
9          // Ambil instance singleton
10         PusatDataSingleton data1 = PusatDataSingleton.GetDataSingleton();
11         PusatDataSingleton data2 = PusatDataSingleton.GetDataSingleton();
12
13         // Tambah data ke data1
14         data1.AddSebuahData("Fahmi Hasan Asagaf");
15         data1.AddSebuahData("Laila");
16         data1.AddSebuahData("Faishal");
17         data1.AddSebuahData("Rizky Naufal Alghifari");
18
19         // Print dari data2 (refleksi dari data1)
20         Console.WriteLine("Data sebelum penghapusan:");
21         data2.PrintSemuaData();
22
23         // Hapus asisten dari data2
24         data2.HapusSebuahData(3);
25
26         // Print ulang dari data1
27         Console.WriteLine("\nData setelah penghapusan:");
28         data1.PrintSemuaData();
29     }
30 }
```