TUGAS PENDAHULUAN
KONSTRUKSI PERANGKAT LUNAK

MODUL XIV
CLEAN CODE

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 PENDAHULUAN

1. MEMBUAT PROJECT MODUL

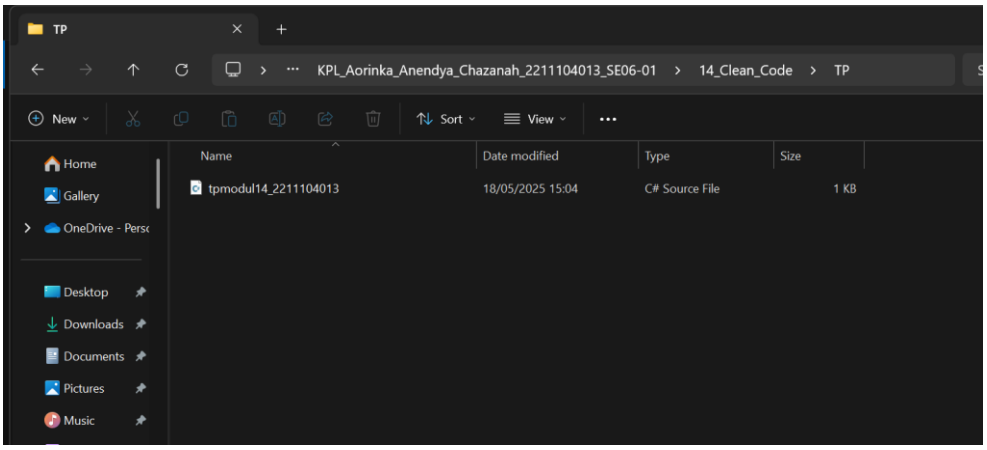
Buka IDE misalnya dengan Visual Studio

A. Copy salah satu folder tugas pendahuluan yang dimiliki sebelumnya (dari modul 2 sampai modul10), kemudian rename folder hasil copy-paste tersebut dengan tpmodul14_NIM (coba pilih tugas pendahuluan yang paling sederhana)

B. Misalnya menggunakan Visual Studio, bukalah project/folder yang di-copy sebelumnya.

Jawaban

Pada contoh berikut telah dilakukan copy paste folder TP dari modul 5 yaitu Generics.



2. REFACTORING DENGAN STANDAR CODE

Dengan mengikuti standard code yang digunakan (misal C# dengan standar dari .NET), pastikan kode yang dikumpulkan memenuhi faktor-faktor berikut:

A. Naming convention
   i. Variable / Property / Attribute
   ii. Method / Function / Procedure

B. White space dan indentation

C. Variable / attribute declarations

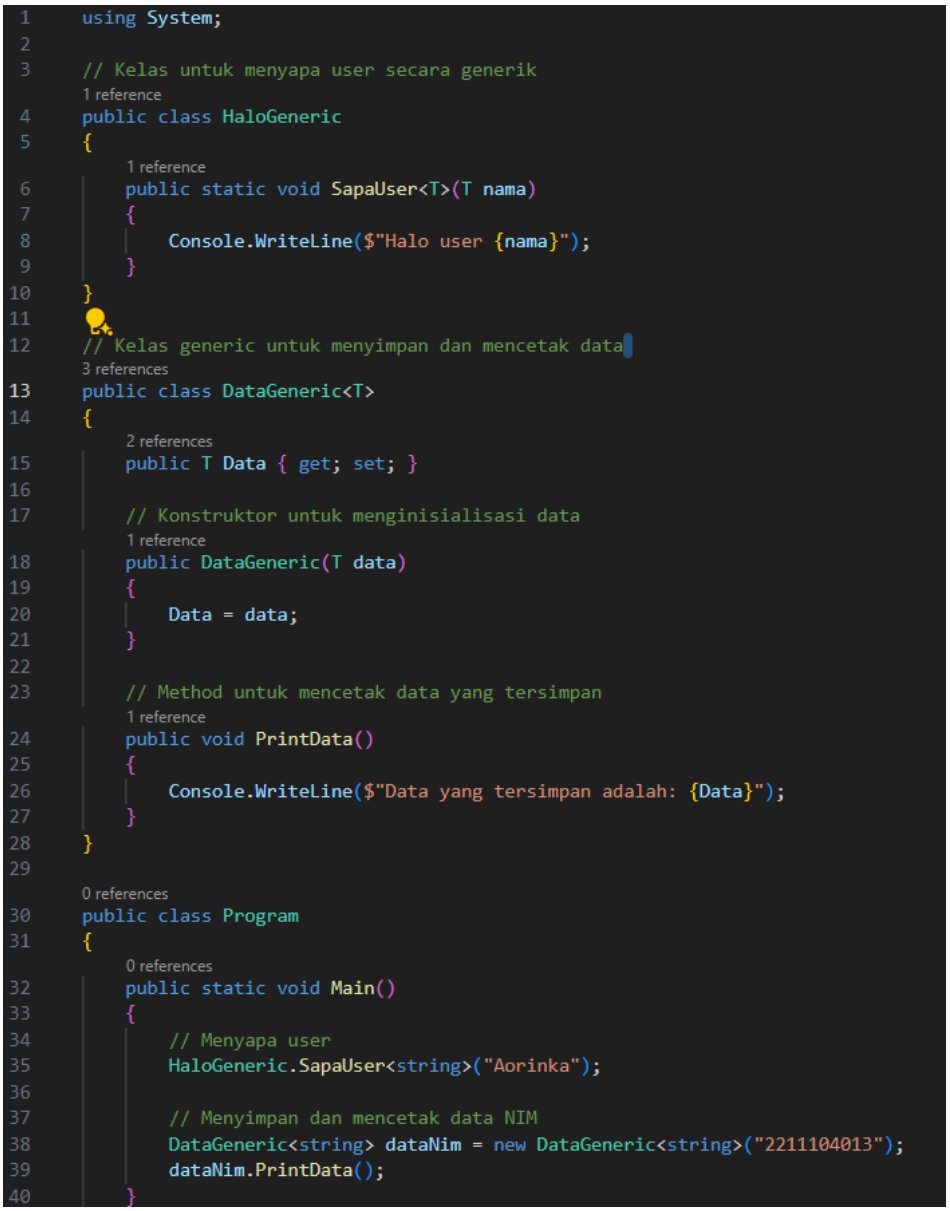
D. Comments

Jawaban

Kode sebelum dilakukan refactoring

```
using System;

1 reference
public class HaloGeneric
{
    1 reference
    public static void SapaUser<T>(T nama)
    {
        Console.WriteLine($"Halo user {nama}");
    }
}


3 references
public class DataGeneric<T>
{
    2 references
    public T Data { get; set; }

    1 reference
    public DataGeneric(T data)
    {
        Data = data;
    }

    1 reference
    public void PrintData()
    {
        Console.WriteLine($"Data yang tersimpan adalah: {Data}");
    }
}

0 references
class Program
{
    0 references
    static void Main()
    {
        HaloGeneric.SapaUser<string>("Aorinka");
        DataGeneric<string> dataNIM = new DataGeneric<string>("2211104013");
        dataNIM.PrintData();
    }
}

```

Kode setelah dilakukan refactoring



```
using System;

// Kelas untuk menyapa user secara generik
1 reference
public class HaloGeneric
{
    1 reference
    public static void SapaUser<T>(T nama)
    {
        Console.WriteLine($"Halo user {nama}");
    }
}

// Kelas generic untuk menyimpan dan mencetak data
3 references
public class DataGeneric<T>
{
    2 references
    public T Data { get; set; }

    // Konstruktor untuk menginisialisasi data
    1 reference
    public DataGeneric(T data)
    {
        Data = data;
    }

    // Method untuk mencetak data yang tersimpan
    1 reference
    public void PrintData()
    {
        Console.WriteLine($"Data yang tersimpan adalah: {Data}");
    }
}

0 references
public class Program
{
    0 references
    public static void Main()
    {
        // Menyapa user
        HaloGeneric.SapaUser<string>("Aorinka");

        // Menyimpan dan mencetak data NIM
        DataGeneric<string> dataNim = new DataGeneric<string>("2211104013");
        dataNim.PrintData();
    }
```

Penjelasan refactoring kode

Kode yang telah ditulis sudah mengalami beberapa proses refactoring untuk mengikuti standar penulisan kode C# (terutama dari segi *naming convention*, whitespace, deklarasi variabel, dan komentar). Refactoring ini bertujuan untuk meningkatkan keterbacaan, konsistensi, dan pemeliharaan kode dalam jangka panjang. Berikut adalah bagian-bagian spesifik yang telah diperbaiki:

1. **Naming Convention**
    - Nama properti Data menggunakan PascalCase, sesuai standar penulisan properti publik di C#.
    - Parameter nama dan data pada method serta konstruktor menggunakan camelCase, yang umum digunakan untuk parameter dan variabel lokal.
    - Nama variabel lokal dataNim di method Main() juga mengikuti camelCase, yang konsisten dengan praktik C#.
2. **Whitespace dan Indentation**
    - Indentasi 4 spasi digunakan secara konsisten di seluruh kode untuk setiap blok kode baru (seperti dalam class, method, dan constructor).
    - Disisipkan baris kosong antar method dan class untuk memberikan ruang visual dan meningkatkan keterbacaan kode.
3. **Deklarasi Variabel / Atribut**
    - Properti generik Data dideklarasikan dengan akses modifier public, lengkap dengan get dan set, mencerminkan prinsip enkapsulasi.
    - Variabel dataNim dideklarasikan secara eksplisit dengan tipe data generik DataGeneric<string>, memberikan kejelasan tipe data yang digunakan.
4. **Comments**
    - Komentar dalam bentuk // digunakan secara ringkas dan jelas untuk menjelaskan maksud dari method, seperti pada PrintData(), menggantikan <summary> XML yang lebih formal dan kurang umum di kode sederhana atau pembelajaran awal.