

**TUGAS PENDAHULUAN**  
**KONSTRUKSI PERANGKAT LUNAK**  
**MODUL XIV**  
**CLEAN CODE**



**Disusun Oleh:**  
**Maria Nathasya Desfera Pangestu**  
**2211104008**  
**SE0601**

**Dosen Pengampu:**  
**Yudha Islami Sulistya, S.Kom., M.Cs.**

**PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK**  
**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 modul 13), kemudian rename folder hasil copy-paste tersebut dengan modul14\_NIM (coba pilih tugas pendahuluan yang paling sederhana)

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

### 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 - Variable / Property / Attribute - Method / Function / Procedure

B. White space dan indentation

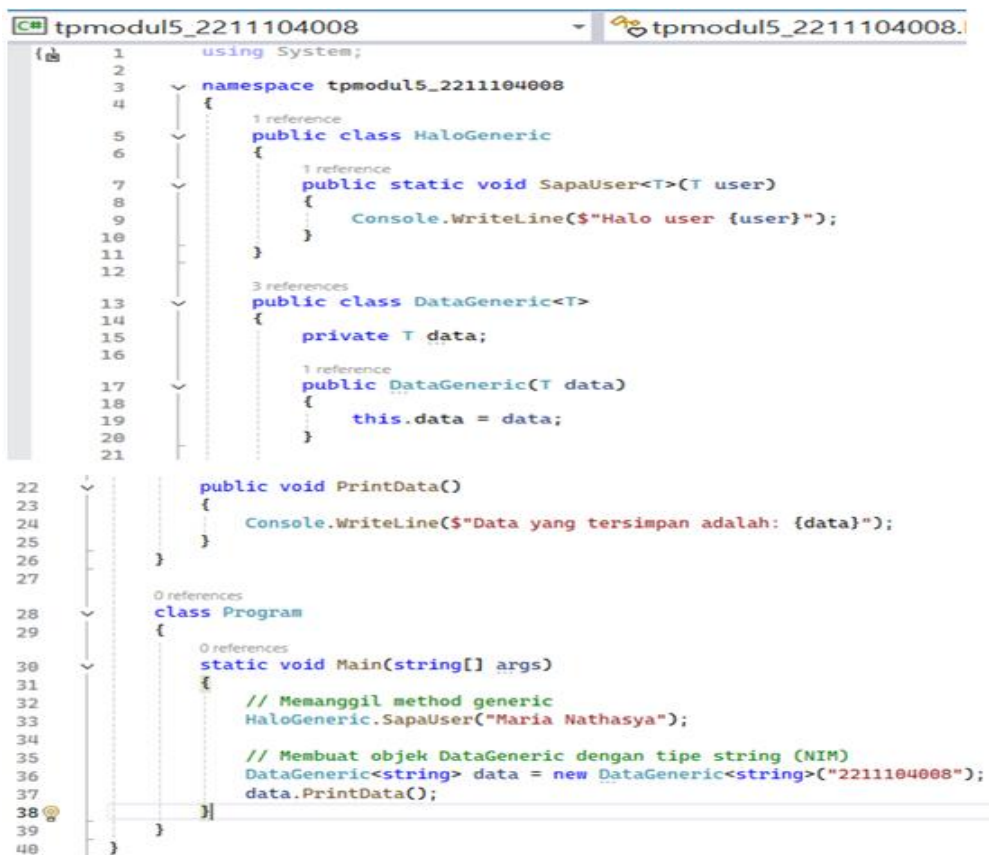
C. Variable / attribute declarations

D. Comments

Jawab:

Saya mencopy tugas pendahuluan modul 5 Generic dan merename nama folder jadi tpmodul14\_2211104008.

Source code sebelum refactoring



```
1 using System;
2
3 namespace tpmodul5_2211104008
4 {
5     1 reference
6     public class HaloGeneric
7     {
8         1 reference
9         public static void SapaUser<T>(T user)
10        {
11            Console.WriteLine($"Halo user {user}");
12        }
13
14    3 references
15    public class DataGeneric<T>
16    {
17        private T data;
18
19        1 reference
20        public DataGeneric(T data)
21        {
22            this.data = data;
23        }
24
25        public void PrintData()
26        {
27            Console.WriteLine($"Data yang tersimpan adalah: {data}");
28        }
29    }
30
31    0 references
32    class Program
33    {
34        0 references
35        static void Main(string[] args)
36        {
37            // Memanggil method generic
38            HaloGeneric.SapaUser("Maria Nathasya");
39
40            // Membuat objek DataGeneric dengan tipe string (NIM)
41            DataGeneric<string> data = new DataGeneric<string>("2211104008");
42            data.PrintData();
43        }
44    }
```

Source code setelah refactoring

```
1  using System;
2
3  1 reference
4  ▼ public class GreetingService
5  {
6      1 reference
7      public static void GreetUser<T>(T user)
8      {
9          Console.WriteLine($"Halo user {user}");
10     }
11 }
12
13 2 references
14 ▼ public class GenericData<T>
15 {
16     private readonly T _data;
17
18     1 reference
19     public GenericData(T data)
20     {
21         _data = data;
22     }
23
24     1 reference
25     public void PrintData()
26     {
27         Console.WriteLine($"Data yang tersimpan adalah: {_data}");
28     }
29 }
30
31 0 references
32 ▼ class Program
33 {
34     0 references
35     static void Main(string[] args)
36     {
37         // Memanggil method untuk menyapa pengguna
38         GreetingService.GreetUser("Maria Nathasya");
39
40         // Membuat objek GenericData dengan tipe string (NIM)
41         var nimData = new GenericData<string>("2211104008");
42         nimData.PrintData();
43     }
44 }
```

Output

```
Microsoft Visual Studio Debu  X + v
Halo user Maria Nathasya
Data yang tersimpan adalah: 2211104008
C:\KPL_MARIA-NATHASYA-DESFERA-PANGESTU_2211104008
```

## Penjelasan

Setelah direfactoring hasilnya yaitu:

- Perubahan Nama Kelas HaloGeneric Menjadi GreetingService  
Nama kelas diubah agar lebih deskriptif dan sesuai dengan fungsinya, yaitu menyediakan layanan untuk menyapa pengguna. Penggunaan bahasa Inggris juga meningkatkan keterbacaan secara internasional.
- Perubahan Nama Method SapaUser<T> Menjadi GreetUser<T>  
Nama method diubah ke bahasa Inggris agar konsisten dengan standar penulisan kode profesional, serta lebih mudah dipahami secara umum. Kata kerja "Greet" menunjukkan bahwa method ini berfungsi untuk menyapa.
- Perubahan Nama Kelas DataGeneric<T> Menjadi GenericData<T>  
Nama kelas diubah untuk mengikuti konvensi penamaan generik dalam bahasa C#, di mana tipe data umumnya diawali dengan kata "Generic" dan diikuti dengan jenis atau fungsinya.
- Penggunaan Nama Variabel \_data untuk Field Privat  
Field privat data diubah menjadi \_data untuk mengikuti konvensi penamaan C# yang umum digunakan pada variabel privat, sehingga membedakannya dengan parameter atau properti lain.
- Penambahan Modifier readonly pada Field \_data  
Penambahan kata kunci readonly digunakan untuk memastikan bahwa nilai field \_data hanya bisa diatur satu kali, yaitu melalui konstruktor. Hal ini meningkatkan keamanan data (immutability) dan mencegah perubahan yang tidak disengaja.
- Penggunaan var dalam Deklarasi Objek  
Pada saat membuat objek GenericData, digunakan keyword var untuk membuat deklarasi lebih ringkas dan modern, tanpa mengorbankan tipe data karena sudah ditentukan oleh nilai inisialisasi.
- Perubahan Struktur Kode yang Lebih Rapi dan Konsisten  
Secara keseluruhan, struktur kode setelah refactoring menjadi lebih konsisten, lebih bersih, dan mengikuti standar penulisan C# yang baik, tanpa mengubah logika atau fungsi utama dari program.