

**LAPORAN PENDAHULUAN  
MODUL 14**



**Disusun Oleh :**

**Zaenarif Putra 'Ainurdin – 2311104049**

**Kelas :**

**SE-07-02**

**Dosen :**

**Yudha Islami Sulistya**

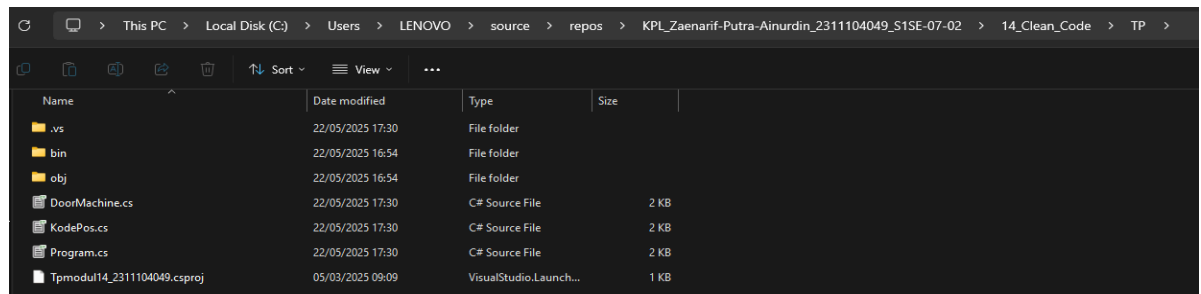
**PROGRAM STUDI SOFTWARE ENGINEERING  
DIREKTORAT KAMPUS PURWOKERTO  
TELKOM UNIVERSITY  
PURWOKERTO  
2025**

## I. Link Github

[https://github.com/zaenarifputra/KPL\\_Zaenarif-Putra-Ainurdin\\_2311104049\\_S1SE-07-02/tree/2015d87e65dd071a22b05dd4c4cb68b9d63bf8e6/14\\_Clean\\_Code/TP](https://github.com/zaenarifputra/KPL_Zaenarif-Putra-Ainurdin_2311104049_S1SE-07-02/tree/2015d87e65dd071a22b05dd4c4cb68b9d63bf8e6/14_Clean_Code/TP)

## II. Alur Pengerjaan

1. Memilih project yang sudah di buat saya memilih project pada modul 4 tp yang dimana menggunakan Console App yang dimana saya mengubah nama yang sudah dibuat menjadi "TpModul14\_2311104049", yang kemudian untuk menyelaraskan clean code dengan menerapkan konsep NET seperti berikut: Naming convention (PascalCase & camelCase), konsistensi method dan class, penggunaan access modifier yang tepat.



2. setelah itu mulai menyelaraskan agar clean code bisa berjalan yang pertama merapihkan class Kodepos.cs

```
using System;

namespace Kodepos
{
    /// <summary>
    /// Kelas untuk menyimpan dan mengelola data kode pos berdasarkan nama kelurahan.
    /// </summary>
    public class KodePos
    {
        // Array dua dimensi untuk menyimpan data kelurahan dan kode pos
        private readonly string[,] _dataKodePos =
        {
            { "Batununggal", "40266" },
            { "Kujangsari", "40287" },
            { "Mengger", "40267" },
            { "Mates", "Mates" },
            { "Cijaura", "40287" },
            { "Jatisari", "40286" },
            { "Margasari", "40286" },
            { "Sekejati", "40286" },
            { "Hebonmaru", "40272" },
            { "Maleer", "40274" },
            { "Sanoja", "40273" }
        };

        /// <summary>
        /// Menampilkan semua data kelurahan beserta kode pos-nya ke konsol.
        /// </summary>
        public void TampilkanSemuaKodePos()
        {
            for (int i = 0; i < _dataKodePos.GetLength(0); i++)
            {
                Console.WriteLine($"Kode Pos Kelurahan {_dataKodePos[i, 0]}: {_dataKodePos[i, 1]}");
            }
        }

        /// <summary>
        /// Mengembalikan kode pos berdasarkan nama kelurahan (case-insensitive).
        /// </summary>
        /// <param name="kelurahan">Nama kelurahan</param>
        /// <returns>Kode pos atau pesan error jika tidak ditemukan</returns>
        public string GetKodePos(string kelurahan)
        {
            for (int i = 0; i < _dataKodePos.GetLength(0); i++)
            {
                if (_dataKodePos[i, 0].Equals(kelurahan, StringComparison.OrdinalIgnoreCase))
                {
                    return _dataKodePos[i, 1];
                }
            }

            return "Data yang dicari tidak ditemukan!";
        }
    }
}
```

Syntax pada kodepos.cs yang merupakan class yang dimana digunakan untuk menyimpan data kelurahan dan terdapat juga kode pos. Method TampilkanSemuaKodePos() digunakan untuk menampilkan semua data kelurahan beserta kode pos-nya. Method GetKodePos() digunakan untuk mencari kode pos yang dimana berdasarkan input kelurahan dengan case-insensitive comparison. Clean code yang diterapkan antara lain: penggunaan private field `_datakodepos`, naming method yang deskriptif (TampilkanSemuaKodePos), dan komentar XML sebagai dokumentasi.

3. Kemudian class DoorMachine.cs dengan menerapkan konsep clean code NET

```

DoorMachine.cs
using System;

namespace Doormachine
{
    /// <summary>
    /// Kelas simulasi mesin pintu sederhana menggunakan konsep state machine.
    /// </summary>
    public class DoorMachine
    {
        // State yang mungkin dimiliki pintu
        private enum DoorState
        {
            Terkunci,
            Terbuka
        }

        // Menyimpan status terkini pintu
        private DoorState _currentState;

        /// <summary>
        /// Konstruktor: Secara default, pintu dalam keadaan terkunci.
        /// </summary>
        public DoorMachine()
        {
            _currentState = DoorState.Terkunci;
            Console.WriteLine("Pintu terkunci");
        }

        /// <summary>
        /// Membuka pintu jika dalam keadaan terkunci.
        /// </summary>
        public void BukaPintu()
        {
            if (_currentState == DoorState.Terkunci)
            {
                _currentState = DoorState.Terbuka;
                Console.WriteLine("Pintu tidak terkunci");
            }
            else
            {
                Console.WriteLine("Pintu sudah terbuka");
            }
        }
    }
}

```

```

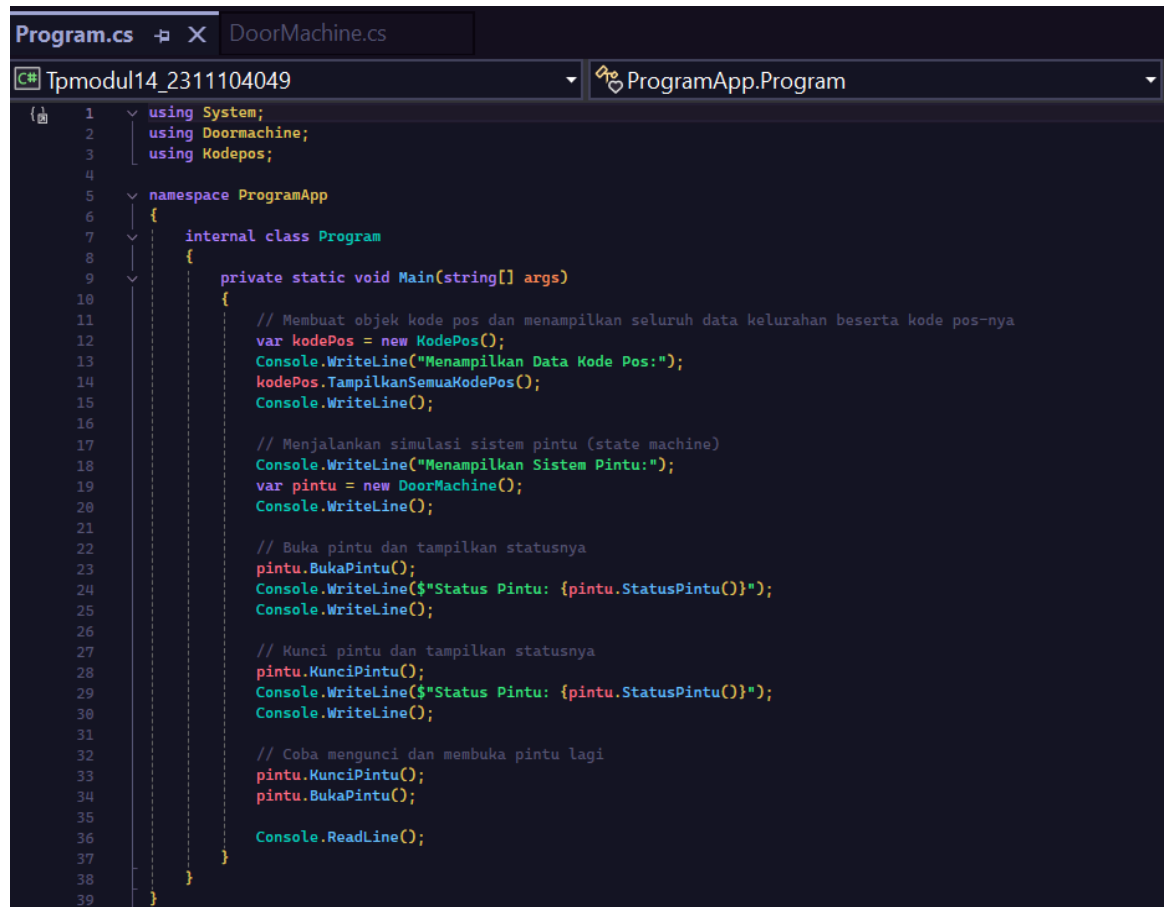
    /// <summary>
    /// Mengunci pintu jika dalam keadaan terbuka.
    /// </summary>
    public void KunciPintu()
    {
        if (_currentState == DoorState.Terbuka)
        {
            _currentState = DoorState.Terkunci;
            Console.WriteLine("Pintu terkunci");
        }
        else
        {
            Console.WriteLine("Pintu sudah terkunci");
        }
    }

    /// <summary>
    /// Mengembalikan status pintu saat ini dalam bentuk string.
    /// </summary>
    /// <returns>Status pintu (terkunci atau tidak)</returns>
    public string StatusPintu()
    {
        return _currentState == DoorState.Terkunci
            ? "Pintu terkunci"
            : "Pintu tidak terkunci";
    }
}

```

Syntax pada class DoorMachine.cs merupakan implementasi state machine sederhana yang menyimulasikan perilaku pintu. Yang dimana pintu memiliki dua method yaitu Terkunci dan Terbuka, class ini juga memungkinkan untuk interaksi seperti membuka atau mengunci pintu, serta melihat status dari method StatusPintu(). Clean code yang digunakan Enum DoorState sebagai representasi state internal, Penamaan yang jelas dan fungsi yang lebih modular, dan komentar yang menjelaskan tiap method dan fungsinya.

4. setelah sudah menyelesaikan clean code pada ke dua class sekarang untuk class Program.cs yang dimana merupakan class yang digunakan sebagai class yang membuat fungsi semua class berjalan dengan baik dan menampilkan sesuai dengan class yang sudah ada dan berhasil di implementasikan.



```
1 using System;
2 using Doormachine;
3 using Kodepos;
4
5 namespace ProgramApp
6 {
7     internal class Program
8     {
9         private static void Main(string[] args)
10        {
11            // Membuat objek kode pos dan menampilkan seluruh data kelurahan beserta kode pos-nya
12            var kodePos = new KodePos();
13            Console.WriteLine("Menampilkan Data Kode Pos:");
14            kodePos.TampilkanSemuaKodePos();
15            Console.WriteLine();
16
17            // Menjalankan simulasi sistem pintu (state machine)
18            Console.WriteLine("Menampilkan Sistem Pintu:");
19            var pintu = new DoorMachine();
20            Console.WriteLine();
21
22            // Buka pintu dan tampilkan statusnya
23            pintu.BukaPintu();
24            Console.WriteLine($"Status Pintu: {pintu.StatusPintu()}");
25            Console.WriteLine();
26
27            // Kunci pintu dan tampilkan statusnya
28            pintu.KunciPintu();
29            Console.WriteLine($"Status Pintu: {pintu.StatusPintu()}");
30            Console.WriteLine();
31
32            // Coba mengunci dan membuka pintu lagi
33            pintu.KunciPintu();
34            pintu.BukaPintu();
35
36            Console.ReadLine();
37        }
38    }
39 }
```

Class Program.cs merupakan sebuah class entri utama pada aplikasi, mengatur alur utama pada program, memanggil fitur fitur dari KodePos, dan DoorMachine, setelah objek sudah di buat seluruh metode dipanggil dengan dengan output yang sesuai pada console. Clean Code yang diterapkan namespace ProgramApp yang konsisten, Penggunaan Var untuk deklarasi objek, Komentar singkat untuk setiap tahapan pada program

5. Setelah semua project sudah diselesaikan selanjutnya upload ke cloud github yang dimana berikut syntax untuk push ke cloud git add . git commit -m "Feat: mengimplementasikan clean code for Modul 14" git push -u origin master.

KPL\_Zaenarif-Putra-Ainurdin\_2311104049\_S1SE-07-02 / 14\_Clean\_Code / TP /

zaenarifputra feat: mengimplementasikan clean code TP14

Name	Last commit message
..	
DoorMachine.cs	feat: mengimplementasikan clean code TP14
KodePos.cs	feat: mengimplementasikan clean code TP14
Program.cs	feat: mengimplementasikan clean code TP14
Tpmodul14_2311104049.csproj	feat: mengimplementasikan clean code TP14

### III. Hasil Running & Kesimpulan

```
C:\Users\LENOVO\source\rep x + v
Menampilkan Data Kode Pos:
Kode Pos Kelurahan Batununggal: 40266
Kode Pos Kelurahan Kujangsari: 40287
Kode Pos Kelurahan Mengger: 40267
Kode Pos Kelurahan Wates: Wates
Kode Pos Kelurahan Cijaura: 40287
Kode Pos Kelurahan Jatisari: 40286
Kode Pos Kelurahan Margasari: 40286
Kode Pos Kelurahan Sekejati: 40286
Kode Pos Kelurahan Kebonwaru: 40272
Kode Pos Kelurahan Maleer: 40274
Kode Pos Kelurahan Samoja: 40273

Menampilkan Sistem Pintu:
Pintu terkunci

Pintu tidak terkunci
Status Pintu: Pintu tidak terkunci

Pintu terkunci
Status Pintu: Pintu terkunci

Pintu sudah terkunci
Pintu tidak terkunci
```

Kesimpulannya adalah refactoring dengan prinsip clean code sangat penting dalam pengembangan perangkat lunak. Dengan mengikuti standar coding seperti: Penamaan yang jelas Struktur kode yang rapi Penggunaan akses yang sesuai (public/private) Pemisahan tanggung jawab antar class maka program menjadi lebih mudah dibaca, dipelihara, dan dikembangkan. Praktikum ini memperkuat pemahaman tentang desain modular dan pentingnya keterbacaan dalam kode. Refactoring juga membantu menyiapkan kode agar siap diintegrasikan ke sistem lebih besar di masa mendatang.