Muhammad Ikhsan Al Hakim

2311104064

Laporan Pendahuluan Modul 4

1. DoorMachine.cs

```
    using System;

    public class DoorMachine
    {
        // Mendefinisikan semua kemungkinan state
        public enum State { Terkunci, Terbuka }
        // Mendefinisikan semua kemungkinan trigger/perintah
        public enum Trigger { KunciPintu, BukaPintu }

        // Menyimpan state saat ini
        public State CurrentState { get; private set; }

        public DoorMachine()
        {
            // State awal adalah Terkunci
            CurrentState = State.Terkunci;
            Console.WriteLine("Pintu terkunci"); // Output saat state
awal
        }

        public void ActivateTrigger(Trigger trigger)
        {
            // Logika transisi state
            switch (CurrentState)
            {
                case State.Terkunci:
                    if (trigger == Trigger.BukaPintu)
                    {
                        CurrentState = State.Terbuka;
                        Console.WriteLine("Pintu tidak terkunci"); //
Output saat masuk ke state Terbuka
                    }
                    // Jika trigger KunciPintu saat sudah Terkunci,
tidak ada perubahan state
                    break;
                case State.Terbuka:
                    if (trigger == Trigger.KunciPintu)
                    {
                        CurrentState = State.Terkunci;
                        Console.WriteLine("Pintu terkunci"); // Output
saat masuk ke state Terkunci
                    }
                    // Jika trigger BukaPintu saat sudah Terbuka, tidak
ada perubahan state
                    break;
            }
        }
    }
```

Penjelasan:

Kode C# ini mengimplementasikan sebuah **State Machine** sederhana menggunakan teknik desain **State-Based Construction**. Tujuannya adalah untuk mengelola dan merepresentasikan perubahan keadaan (state) dari sebuah objek pintu.

Poin-Poin Utama:

1. **Definisi State dan Trigger**:
   - Kelas ini mendefinisikan semua kemungkinan keadaan pintu (Terkunci, Terbuka) menggunakan sebuah enum bernama State.
   - Aksi atau input yang dapat mengubah keadaan tersebut (misalnya, KunciPintu, BukaPintu) juga didefinisikan dalam enum Trigger.
2. **Manajemen State**:
   - Kelas ini memiliki properti CurrentState untuk selalu melacak keadaan pintu saat ini.
   - Saat objek DoorMachine pertama kali dibuat (di dalam konstruktor), *state* awalnya langsung diatur ke Terkunci dan sebuah pesan dicetak ke konsol.
3. **Logika Transisi State**:
   - Metode ActivateTrigger adalah antarmuka utama untuk mengubah state pintu. Ketika dipanggil dengan sebuah Trigger, ia akan menggunakan struktur switch-case untuk memeriksa state saat ini.
   - Berdasarkan state saat ini dan *trigger* yang diberikan, program akan menentukan apakah transisi ke state lain diizinkan. Jika diizinkan, CurrentState akan diperbarui.
   - Jika sebuah *trigger* tidak valid untuk state saat ini (misalnya, mencoba mengunci pintu yang sudah terkunci), tidak ada perubahan state yang terjadi.
4. **Umpan Balik (Feedback)**:
   - Setiap kali pintu berhasil masuk ke sebuah state baru (termasuk state awal), sebuah pesan yang sesuai ("Pintu terkunci" atau "Pintu tidak terkunci") akan dicetak ke konsol untuk memberikan umpan balik yang jelas.

2. KodePos.cs

```
using System;
using System.Collections.Generic;

public class KodePos
{
    // Tabel data disimpan dalam Dictionary untuk pencarian cepat
    private Dictionary<string, string> kodePosTable = new
Dictionary<string, string>()
    {
        {"Batununggal", "40266"},
        {"Kujangsari",  "40287"},
        {"Mengger",     "40267"},
        {"Wates",       "40256"},
        {"Cijaura",     "40287"},
        {"Jatisari",    "40286"},
        {"Margasari",   "40286"},
        {"Sekejati",    "40286"},
        {"Kebonwaru",   "40272"},
```

```
        {"Maleer",      "40274"},
        {"Samoja",      "40273"}
    };

    public string getKodePos(string kelurahan)
    {
        if (kodePosTable.ContainsKey(kelurahan))
        {
            return kodePosTable[kelurahan];
        }
        return "Kode Pos tidak ditemukan";
    }
}
```

Penjelasan:

Kode C# ini mendefinisikan sebuah kelas bernama KodePos yang menerapkan teknik desain **Table-Driven**.

Poin-Poin Utama:

1. **Struktur Data (Tabel)**: Kelas ini menggunakan Dictionary bernama kodePosTable untuk menyimpan pasangan data, yaitu nama kelurahan (sebagai *key*) dan kode pos (sebagai *value*). Dictionary ini berfungsi sebagai "tabel" data yang terpusat.
2. **Enkapsulasi Data**: Tabel data kodePosTable dideklarasikan sebagai private, artinya data tersebut hanya bisa diakses dari dalam kelas KodePos, sehingga menjaga integritas data.
3. **Metode Pencarian (getKodePos)**: Terdapat satu metode publik getKodePos yang menerima kelurahan sebagai input.
4. **Logika Pencarian**: Metode ini secara efisien memeriksa apakah kelurahan ada di dalam "tabel" (kodePosTable).
    - Jika nama kelurahan ditemukan, metode akan mengembalikan kode pos yang sesuai.
    - Jika tidak ditemukan, metode akan mengembalikan pesan "Kode Pos tidak ditemukan".

3. Program.cs

```
using System;

namespace tpmodul3_2311104064
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("--- Demo Table-Driven (KodePos) ---");
            KodePos tabelKodePos = new KodePos();

            // Contoh pemanggilan
            string kelurahan = "Batununggal";
            string kode = tabelKodePos.getKodePos(kelurahan);
            Console.WriteLine($"Kode Pos untuk kelurahan {kelurahan}
adalah: {kode}");
```

```
            kelurahan = "Samoja";
            kode = tabelKodePos.getKodePos(kelurahan);
            Console.WriteLine($"Kode Pos untuk kelurahan {kelurahan}
adalah: {kode}");

            // Bagian untuk State-Based akan ditambahkan nanti di bawah
ini
            Console.WriteLine("\n--- Tekan tombol apa saja untuk
melanjutkan ke demo State-Based ---");
            Console.ReadKey();

            Console.WriteLine("\n--- Demo State-Based (DoorMachine) ---
");
            DoorMachine door = new DoorMachine();
            Console.WriteLine($"State awal pintu: {door.CurrentState}");

            // Simulasi perubahan state
            Console.WriteLine("\n> Membuka pintu...");
            door.ActivateTrigger(DoorMachine.Trigger.BukaPintu);
            Console.WriteLine($"State sekarang: {door.CurrentState}");

            Console.WriteLine("\n> Mencoba membuka pintu lagi...");
            door.ActivateTrigger(DoorMachine.Trigger.BukaPintu); //
State tidak akan berubah
            Console.WriteLine($"State sekarang: {door.CurrentState}");

            Console.WriteLine("\n> Mengunci pintu...");
            door.ActivateTrigger(DoorMachine.Trigger.KunciPintu);
            Console.WriteLine($"State sekarang: {door.CurrentState}");

            Console.WriteLine("\n> Mencoba mengunci pintu lagi...");
            door.ActivateTrigger(DoorMachine.Trigger.KunciPintu); //
State tidak akan berubah
            Console.WriteLine($"State sekarang: {door.CurrentState}");

            Console.WriteLine("\n--- Simulasi Selesai ---");
        }
    }
}
```

Penjelasan:

Kode C# yang Anda berikan adalah program konsol utama (Main) yang berfungsi untuk mendemonstrasikan dua teknik desain yang berbeda, yaitu *Table-Driven* dan *State-Based Construction*.

Poin-Poin Utama:

1. **Demonstrasi Table-Driven (KodePos)**:
    - Bagian pertama program membuat sebuah objek dari kelas KodePos.
    - Kemudian, ia memanggil metode getKodePos beberapa kali dengan nama kelurahan yang berbeda ("Batununggal" dan "Samoja") untuk mengambil kode pos yang sesuai dari sebuah tabel data internal (kemungkinan sebuah *Dictionary*).

- Tujuannya adalah untuk menunjukkan bagaimana logika dapat disederhanakan dengan mengambil data dari sebuah struktur tabel daripada menggunakan banyak pernyataan if-else.

2. **Pemisah Demonstrasi**:
   - Console.ReadKey(); digunakan untuk menjeda program setelah demonstrasi pertama. Program akan menunggu pengguna menekan sebuah tombol sebelum melanjutkan ke demonstrasi kedua.
3. **Demonstrasi State-Based Construction (DoorMachine)**:
   - Bagian kedua membuat sebuah objek dari kelas DoorMachine.
   - Program kemudian mensimulasikan serangkaian "aksi" (BukaPintu, KunciPintu) dengan memanggil metode ActivateTrigger.
   - Simulasi ini dirancang secara spesifik untuk menjalankan semua transisi state yang mungkin (Terkunci ke Terbuka, dan sebaliknya) serta menunjukkan bahwa state tidak akan berubah jika aksi yang diberikan tidak valid untuk state saat ini (misalnya, mencoba membuka pintu yang sudah terbuka).

4. Hasil Run:

```
Kode Pos untuk kelurahan Batununggal adalah: 40266
Kode Pos untuk kelurahan Samoja adalah: 40273

--- Tekan tombol apa saja untuk melanjutkan ke demo State-Based ---
w
--- Demo State-Based (DoorMachine) ---
Pintu terkunci
State awal pintu: Terkunci

> Membuka pintu...
Pintu tidak terkunci
State sekarang: Terbuka

> Mencoba membuka pintu lagi...
State sekarang: Terbuka

> Mengunci pintu...
Pintu terkunci
State sekarang: Terkunci

> Mencoba mengunci pintu lagi...
State sekarang: Terkunci

--- Simulasi Selesai ---
```