Muhammad Ikhsan Al Hakim

2311104064

Laporan Jurnal Modul 4

1. Program.cs

```
using System;

namespace modul4_2311104064 // Pastikan namespace ini sesuai dengan
project Anda
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("--- Demo Table-Driven (KodeBuah) ---");
            KodeBuah tabelBuah = new KodeBuah();

            // Contoh pemanggilan
            string namaBuah = "Apel";
            string kode = tabelBuah.getKodeBuah(namaBuah);
            Console.WriteLine($"Kode untuk buah {namaBuah} adalah:
{kode}");

            namaBuah = "Melon";
            kode = tabelBuah.getKodeBuah(namaBuah);
            Console.WriteLine($"Kode untuk buah {namaBuah} adalah:
{kode}");

            namaBuah = "Anggur";
            kode = tabelBuah.getKodeBuah(namaBuah);
            Console.WriteLine($"Kode untuk buah {namaBuah} 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
(PosisiKarakterGame) ---");
            PosisiKarakterGame gameChar = new PosisiKarakterGame();
            Console.WriteLine($"State awal: {gameChar.CurrentState}");

            // Simulasi untuk menghasilkan output NIM % 3 == 1
            Console.WriteLine("\nMenekan Tombol S (Berdiri ->
Jongkok):");

gameChar.ActivateTrigger(PosisiKarakterGame.Trigger.TombolS);

            Console.WriteLine("\nMenekan Tombol W (Jongkok ->
Berdiri):");

gameChar.ActivateTrigger(PosisiKarakterGame.Trigger.TombolW); // Akan
ada output "posisi standby"
```

```
            Console.WriteLine("\nMenekan Tombol S (Berdiri ->
Jongkok):");

gameChar.ActivateTrigger(PosisiKarakterGame.Trigger.TombolS);

            Console.WriteLine("\nMenekan Tombol S (Jongkok ->
Tengkurap):");

gameChar.ActivateTrigger(PosisiKarakterGame.Trigger.TombolS); // Akan
ada output "posisi istirahat"

            Console.WriteLine("\nMenekan Tombol W (Tengkurap ->
Jongkok):");

gameChar.ActivateTrigger(PosisiKarakterGame.Trigger.TombolW);

            Console.WriteLine("\nMenekan Tombol W (Jongkok ->
Berdiri):");

gameChar.ActivateTrigger(PosisiKarakterGame.Trigger.TombolW); // Akan
ada output "posisi standby" lagi

            Console.WriteLine("\nMenekan Tombol W (Berdiri ->
Terbang):");

gameChar.ActivateTrigger(PosisiKarakterGame.Trigger.TombolW);

            Console.WriteLine("\nMenekan Tombol S (Terbang ->
Berdiri):");

gameChar.ActivateTrigger(PosisiKarakterGame.Trigger.TombolS); // Akan
ada output "posisi standby" lagi

            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*.

Berikut adalah poin-poin penting dari kode tersebut:

1. **Demonstrasi Table-Driven (`KodeBuah`)**:
    - Bagian pertama program membuat sebuah objek dari kelas `KodeBuah`.
    - Kemudian, ia memanggil metode `getKodeBuah` beberapa kali dengan nama buah yang berbeda ("Apel", "Melon", "Anggur") untuk mengambil kode buah 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 (`PosisiKarakterGame`)**:
    - Bagian kedua membuat sebuah objek dari kelas `PosisiKarakterGame`.
    - Program kemudian mensimulasikan serangkaian "penekanan tombol" (`TombolS`, `TombolW`, `TombolX`) dengan memanggil metode `ActivateTrigger`.
    - Simulasi ini dirancang secara spesifik untuk menjalankan semua transisi state yang mungkin (Berdiri, Jongkok, Tengkurap, Terbang) dan untuk memicu output tambahan yang telah ditentukan berdasarkan sisa pembagian NIM (`NIM % 3 == 1`), yaitu "posisi standby" dan "posisi istirahat" pada saat state tertentu tercapai.

2. KodeBuah.cs

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

public class KodeBuah
{
    // Tabel data disimpan dalam Dictionary
    private Dictionary<string, string> kodeMapping = new
Dictionary<string, string>()
    {
        {"Apel",       "A00"},
        {"Aprikot",    "B00"},
        {"Alpukat",    "C00"},
        {"Pisang",     "D00"},
        {"Paprika",    "E00"},
        {"Blackberry", "F00"},
        {"Ceri",       "H00"},
        {"Kelapa",     "I00"}, // 'I' bukan '1'
        {"Jagung",     "J00"},
        {"Kurma",      "K00"},
        {"Durian",     "L00"},
        {"Anggur",     "M00"},
        {"Melon",      "N00"},
        {"Semangka",   "O00"}  // 'O' bukan '0'
    };

    public string getKodeBuah(string namaBuah)
    {
        if (kodeMapping.ContainsKey(namaBuah))
        {
            return kodeMapping[namaBuah];
        }
        return "Kode tidak ditemukan";
    }
}
```

Penjelasan:

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

Poin-Poin Utama:

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

3. PosisiKarakterGame.cs

```
using System;

public class PosisiKarakterGame
{
    // Mendefinisikan semua kemungkinan state
    public enum State { Berdiri, Jongkok, Terbang, Tengkurap }
    // Mendefinisikan semua kemungkinan trigger/perintah
    public enum Trigger { TombolW, TombolS, TombolX }

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

    // Representasi transisi state: Key = State Awal, Value = List dari
(Trigger, State Akhir)
    private Transition[,] transitions;

    public class Transition
    {
        public State StateAwal;
        public State StateAkhir;
        public Trigger Pemicu;

        public Transition(State awal, State akhir, Trigger pemicu)
        {
            StateAwal = awal;
            StateAkhir = akhir;
            Pemicu = pemicu;
        }
    }

    public PosisiKarakterGame()
    {
        // State awal adalah Berdiri
        CurrentState = State.Berdiri;

        // Inisialisasi transisi berdasarkan diagram
        Transition[] transisi = {
```

```
            new Transition(State.Berdiri, State.Jongkok,
Trigger.TombolS),
            new Transition(State.Berdiri, State.Terbang,
Trigger.TombolW),
            new Transition(State.Jongkok, State.Berdiri,
Trigger.TombolW),
            new Transition(State.Jongkok, State.Tengkurap,
Trigger.TombolS),
            new Transition(State.Tengkurap, State.Jongkok,
Trigger.TombolW),
            new Transition(State.Terbang, State.Berdiri,
Trigger.TombolS),
            new Transition(State.Terbang, State.Jongkok,
Trigger.TombolX) // Transisi unik dengan TombolX
        };

        // Untuk implementasi sederhana, kita bisa menggunakan list saja
        // atau jika ingin lebih cepat, bisa menggunakan struktur data
yang lebih kompleks.
        // Di sini kita akan buat sederhana dengan array of object.
        this.transitions = new Transition[transisi.Length, 1];
        for (int i = 0; i < transisi.Length; i++)
        {
            this.transitions[i, 0] = transisi[i];
        }
    }

    private State GetNextState(State stateAwal, Trigger pemicu)
    {
        State stateAkhir = stateAwal; // Default tidak berubah state
jika transisi tidak ditemukan
        for (int i = 0; i < this.transitions.Length /
this.transitions.Rank; i++)
        {
            if (this.transitions[i, 0] != null && this.transitions[i,
0].StateAwal == stateAwal && this.transitions[i, 0].Pemicu == pemicu)
            {
                stateAkhir = this.transitions[i, 0].StateAkhir;
                break;
            }
        }
        return stateAkhir;
    }

    public void ActivateTrigger(Trigger pemicu)
    {
        State stateBerikutnya = GetNextState(CurrentState, pemicu);

        // Kondisi tambahan berdasarkan NIM % 3 == 1
        if (stateBerikutnya == State.Berdiri)
        {
            Console.WriteLine("posisi standby");
        }
        if (stateBerikutnya == State.Tengkurap)
        {
            Console.WriteLine("posisi istirahat");
        }

        CurrentState = stateBerikutnya;
        Console.WriteLine($"State sekarang: {CurrentState}");
    }
}
```

Penjelasan:

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

Poin-Poin Utama:

1. **Definisi State dan Trigger**:
   - Kelas ini mendefinisikan semua kemungkinan keadaan karakter (`Berdiri`, `Jongkok`, `Terbang`, `Tengkurap`) menggunakan sebuah `enum` bernama `State`.
   - Aksi atau input yang dapat mengubah keadaan tersebut (misalnya, penekanan tombol) juga didefinisikan dalam `enum Trigger` (`TombolW`, `TombolS`, `TombolX`).
2. **Aturan Transisi (Transition Rules)**:
   - Semua aturan perubahan state (misalnya, dari `Berdiri` jika ditekan `TombolS` akan menjadi `Jongkok`) didefinisikan secara eksplisit di dalam konstruktor dan disimpan dalam sebuah array `transitions`. Ini memisahkan aturan dari logika eksekusi.
3. **Mekanisme Perubahan State**:
   - Metode `ActivateTrigger` adalah antarmuka utama untuk mengubah state karakter. Ketika dipanggil dengan sebuah `Trigger`, ia akan mencari aturan transisi yang sesuai untuk menentukan state berikutnya.
   - Jika tidak ada aturan yang cocok, karakter akan tetap pada state saat ini.
4. **Logika Tambahan Berdasarkan Kondisi**:
   - Kode ini memiliki logika tambahan yang spesifik (sesuai dengan hasil `NIM % 3 == 1`). Sebelum state diubah, program akan memeriksa apakah state berikutnya adalah `Berdiri` atau `Tengkurap`.
   - Jika ya, ia akan mencetak pesan khusus ("posisi standby" atau "posisi istirahat") ke konsol.

4. Hasil Run

```
--- Demo Table-Driven (KodeBuah) ---
Kode untuk buah Apel adalah: A00
Kode untuk buah Melon adalah: N00
Kode untuk buah Anggur adalah: M00

--- Tekan tombol apa saja untuk melanjutkan ke demo State-Based ---
1
--- Demo State-Based (PosisiKarakterGame) ---
State awal: Berdiri

Menekan Tombol S (Berdiri -> Jongkok):
State sekarang: Jongkok

Menekan Tombol W (Jongkok -> Berdiri):
posisi standby
State sekarang: Berdiri

Menekan Tombol S (Berdiri -> Jongkok):
State sekarang: Jongkok

Menekan Tombol S (Jongkok -> Tengkurap):
State sekarang: Jongkok

Menekan Tombol W (Tengkurap -> Jongkok):
posisi standby
State sekarang: Berdiri

Menekan Tombol W (Jongkok -> Berdiri):
State sekarang: Terbang

Menekan Tombol W (Berdiri -> Terbang):
State sekarang: Terbang

Menekan Tombol S (Terbang -> Berdiri):
State sekarang: Terbang

--- Simulasi Selesai ---
```